Sipna College of Engineering & Technology, Amravati.
Department of Computer Science & Engineering
Session 2022-2023

Branch :- Computer Sci. & Engg.
Subject :-Block Chain Fundamentals Lab manual **Class :- Final Year**
Teacher Manual **Sem :- VII**

PRACTICAL NO 1

AIM: Understand basics of Block Chain and install various software's required to perform Blockchain practical's

INTRODUCTION TO BLOCK CHAIN:

What Is a Block chain?

A block chain is a distributed database or ledger that is shared among the nodes of a computer network. As a database, a block chain stores information electronically in digital format. Block chains are best known for their crucial role in cryptocurrency systems, such as Bitcoin, for maintaining a secure and decentralized record of transactions. The innovation with a block chain is that it guarantees the fidelity and security of a record of data and generates trust without the need for a trusted third party.

One key difference between a typical database and a block chain is how the data is structured. A block chain collects information together in groups, known as blocks, that hold sets of information. Blocks have certain storage capacities and, when filled, are closed and linked to the previously filled block, forming a chain of data known as the block chain. All new information that follows that freshly added block is compiled into a newly formed block that will then also be added to the chain once filled.

A database usually structures its data into tables, whereas a block chain, as its name implies, structures its data into chunks (blocks) that are strung together. This data structure inherently makes an irreversible timeline of data when implemented in a decentralized nature. When a block is filled, it is set in stone and becomes a part of this timeline. Each block in the chain is given an exact timestamp when it is added to the chain.

Some key features of Block Chain are:

- Blockchain is a type of shared database that differs from a typical database in the way that it stores information; blockchains store data in blocks that are then linked together via cryptography.
- As new data comes in, it is entered into a fresh block. Once the block is filled with data, it is chained onto the previous block, which makes the data chained together in chronological order.
- Different types of information can be stored on a blockchain, but the most common use so far has been as a ledger for transactions.
- In Bitcoin's case, blockchain is used in a decentralized way so that no single person or group has control—rather, all users collectively retain control.
- Decentralized blockchains are immutable, which means that the data entered is irreversible. For Bitcoin, this means that transactions are permanently recorded and viewable to anyone.

S/W REQUIRED:

We will be using various software for performing the practicals

1) **Phython:** Python is a powerful, high-level scripting language used by many developers around the globe. The language is ideal for a variety of real-world applications including web development, web scraping, and penetration testing.

How to intall Python in Ubuntu

Step 1: Setting Up Python 3

Ubuntu and other versions of Debian Linux ship with Python 3 pre-installed. To make sure that our versions are up-to-date, update your local package index:

sudo apt update

Then upgrade the packages installed on your system to ensure you have the latest versions:

sudo apt -y upgrade

The -y flag will confirm that we are agreeing for all items to be installed, but depending on your version of Linux, you may need to confirm additional prompts as your system updates and upgrades.

Once the process is complete, we can check the version of Python 3 that is installed in the system by typing:

python3 –V

You'll receive output in the terminal window that will let you know the version number. While this number may vary, the output will be similar to this:

Output
Python 3.8.10

To manage software packages for Python, let's install pip, a tool that will install and manage programming packages we may want to use in our development projects. You can learn more about modules or packages that you can install with pip by reading How To Import Modules in Python 3.

sudo apt install -y python3-pip

Python packages can be installed by typing:

pip3 install package_name

Here, package_name can refer to any Python package or library, such as Django for web development or NumPy for scientific computing. So if you would like to install NumPy, you can do so with the command pip3 install numpy.

There are a few more packages and development tools to install to ensure that we have a robust setup for our programming environment:

sudo apt install -y build-essential libssl-dev libffi-dev python3-dev

Once Python is set up, and pip and other tools are installed, we can set up a virtual environment for our development projects.

Step 2: Setting Up Python 3

Virtual environments enable you to have an isolated space on your server for Python projects, ensuring that each of your projects can have its own set of dependencies that won't disrupt any of your other projects.

While there are a few ways to achieve a programming environment in Python, we'll be using the **venv** module here, which is part of the standard Python 3 library. Let's install venv by typing:

```
sudo apt install -y python3-venv
```

With this installed, we are ready to create environments. Let's either choose which directory we would like to put our Python programming environments in, or create a new directory with mkdir, as in:

```
mkdir environments
```

then navigate to the directory where you'll store your programming environments:

```
cd environments
```

Once you are in the directory where you would like the environments to live, you can create an environment by running the following command:

```
python3 -m venv my_env
```

Essentially, pyvenv sets up a new directory that contains a few items which we can view with the ls command:

```
ls my_env
```

```
Output
bin include lib lib64 pyvenv.cfg share
```

To use this environment, you need to activate it, which you can achieve by typing the following command that calls the activate script:

```
source my_env/bin/activate
```

Our command prompt will now be prefixed with the name of your environment, in this case it is called my_env. Depending on what version of Debian Linux you are running, your prefix may appear somewhat differently, but the name of your environment in parentheses should be the first thing you see on your line:

```
(my_env) Sammy@ubuntu:~enviornment$
```

open up a command line text editor such as nano and create a new file:

nano hello.py

2) Etherum:

- Ethereum is a decentralized open-source platform based on blockchain domain, used to run smart contracts i.e. applications that execute the program exactly as it was programmed without the possibility of any fraud, interference from a third party, censorship, or downtime. It serves a platform for nearly 2,60,000 different cryptocurrencies. Ether is a cryptocurrency generated by ethereum miners, used to reward for the computations performed to secure the blockchain.

- **Ethereum Virtual Machine(EVM)**

Ethereum Virtual Machine abbreviated as EVM is a runtime environment for executing smart contracts in ethereum. It focuses widely on providing security and execution of untrusted code using an international network of public nodes. EVM is specialized to prevent Denial-of-service attack and confirms that the program does not have any access to each other's state, also ensures that the communication is established without any potential interference.

- **Solidity**

Solidity is a brand-new programming language created by the Ethereum which is the second-largest market of cryptocurrency by capitalization, released in the year 2015 led by Christian Reitwiessner.

Some key features of solidity are listed below:

- Solidity is a high-level programming language designed for implementing smart contracts.
- It is statically-typed object-oriented(contract-oriented) language.
- Solidity is highly influenced by Python, c++, and JavaScript which runs on the Ethereum Virtual Machine(EVM).
- Solidity supports complex user-defined programming, libraries and inheritance.
- Solidity is primary language for blockchains running platforms.
- Solidity can be used to creating contracts like voting, blind auctions, crowdfunding, multi-signature wallets, etc.

- **Smart Contract**

Smart contracts are high-level program codes that are compiled to EVM byte code and deployed to the ethereum blockchain for further execution. It allows us to perform credible transactions without any interference of the third party, these transactions are trackable and irreversible. Languages used to write smart contracts are Solidity (a language library with similarities to C and JavaScript), Serpent (similar to Python, but deprecated), LLL (a low-level Lisp-like language), and Mutan (Go-based, but deprecated).

- **REMIX IDE**

Remix IDE allows developing, deploying and administering smart contracts for Ethereum like blockchains. It can also be used as a learning platform.

The Remix is an Integrated Development Environment(IDE) for developing smart contracts in Solidity programming language. The IDE can be used to write, compile, and debug the Solidity code.

It was written in JavaScript and supports testing, debugging and deploying smart contracts, and much more.
Remix-IDE can be accessed in many different ways:

- You can use it online in any browser of your choice, by entering the URL: https://remix.ethereum.org/ in the browser.
- Or you can install it in your own system

Steps to install Remix IDE

Step 1:
INSTALLATION:

Remix-ide has been published as an npm module
Install npm and node.js , then do:

npm install -g @remix-project/remixd

Step 2:
Docker

- Run this command to download the current stable release of Docker Compose:

```
sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

- Apply executable permissions to the binary:

```
sudo chmod +x /usr/local/bin/docker-compose
```

Run with docker

```
docker pull remixproject/remix-ide:remix_live

docker run -p 8080:80 remixproject/remix-ide:remix_live
```

Step 3:
Then go to http://localhost:8080 and you can use your Remix instance.

CONCLUSION: Thus we have studied basics of Block Chain and installed various software's required to perform Blockchain practical's

Sipna College of Engineering & Technology, Amravati.
Department of Computer Science & Engineering
Session 2022-2023

Branch :- Computer Sci. & Engg.
Subject :-Block Chain Fundamentals Lab manual **Class :- Final Year**
Sem :- VII

Teacher Manual

PRACTICAL NO 2

AIM: Implement Diffie-Hellman Algorithm

S/W REQUIRED: Phython

Diffie-Hellman algorithm

The Diffie-Hellman algorithm is being used to establish a shared secret that can be used for secret communications while exchanging data over a public network using the elliptic curve to generate points and get the secret key using the parameters.

For the sake of simplicity and practical implementation of the algorithm, we will consider only 4 variables, one prime P and G (a primitive root of P) and two private values a and b.
P and G are both publicly available numbers. Users (say Alice and Bob) pick private values a and b and they generate a key and exchange it publicly. The opposite person receives the key and that generates a secret key, after which they have the same secret key to encrypt.

Step by Step Explanation

Alice	Bob
Public Keys available = P, G	Public Keys available = P, G
Private Key Selected = a	Private Key Selected = b
Key generated = $x=G^a \bmod P$	Key generated = $y=G^b \bmod P$
Exchange of generated keys takes place	
Key received = y	key received = x
Generated Secret Key = $K_a=y^a \bmod P$	Generated Secret Key = $K_b=x^b \bmod P$

Alice	Bob
Algebraically, it can be shown that $K_a = K_b$	
Users now have a symmetric secret key to encrypt	

Example:

Step 1: Alice and Bob get public numbers P = 23, G = 9

Step 2: Alice selected a private key a = 4 and
Bob selected a private key b = 3

Step 3: Alice and Bob compute public values
Alice: x =(9^4 mod 23) = (6561 mod 23) = 6
Bob: y = (9^3 mod 23) = (729 mod 23) = 16

Step 4: Alice and Bob exchange public numbers

Step 5: Alice receives public key y =16 and
Bob receives public key x = 6

Step 6: Alice and Bob compute symmetric keys
Alice: ka = y^a mod p = 65536 mod 23 = 9
Bob: kb = x^b mod p = 216 mod 23 = 9

Step 7: 9 is the shared secret.

Implementation:

```
from random import randint

if __name__ == '__main__':

    # Both the persons will be agreed upon the
    # public keys G and P
    # A prime number P is taken
    P = 23

    # A primitive root for P, G is taken
    G = 9


    print('The Value of P is :%d'%(P))
    print('The Value of G is :%d'%(G))
```

```
# Alice will choose the private key a
a = 4
print('The Private Key a for Alice is :%d'%(a))

# gets the generated key
x = int(pow(G,a,P))

# Bob will choose the private key b
b = 3
print('The Private Key b for Bob is :%d'%(b))

# gets the generated key
y = int(pow(G,b,P))


# Secret key for Alice
ka = int(pow(y,a,P))

# Secret key for Bob
kb = int(pow(x,b,P))

print('Secret key for the Alice is : %d'%(ka))
print('Secret Key for the Bob is : %d'%(kb))
```

Output:

```
The value of P : 23
The value of G : 9

The private key a for Alice : 4
The private key b for Bob : 3

Secret key for the Alice is : 9
Secret Key for the Bob is : 9
```

CONCLUSION: Thus we have implemented a Diffie-Hellman Algorithm.

Sipna College of Engineering & Technology, Amravati.
Department of Computer Science & Engineering
Session 2022-2023

Branch :- Computer Sci. & Engg. **Class :- Final Year**
Subject :-Block Chain Fundamentals Lab manual **Sem :- VII**
Teacher Manual

PRACTICAL NO 3

AIM: Implement various SHA algorithms

S/W REQUIRED: Phython

SHA (Secure Hash Algorithms)

SHA stands for secure hashing algorithm. SHA is a modified version of MD5 and used for hashing information and certificates. A hashing algorithm shortens the input information into a smaller form that cannot be learned by utilizing bitwise operations, modular additions, and compression functions.

SHAs also help in revealing if an original message was transformed in any way. By imputing the original hash digest, a user can tell if even an individual letter has been shifted, as the hash digests will be effectively different.

The important element of SHAs are that they are deterministic. This define that consider the hash function used is known, any computer or user can regenerate the hash digest. The determinism of SHAs is one of main reasons that each SSL certificate on the Internet is needed to have been hashed with a SHA-2 function.

SHA, (Secure Hash Algorithms) are set of cryptographic hash functions defined by the language to be used for various applications such as password security etc. Some variants of it are supported by Python in the "hashlib" library. These can be found using "algorithms_guaranteed" function of hashlib.

SHA Hash

The different SHA hash functions are explained below.

1. SHA256 : This hash function belong to hash class SHA-2, the internal block size of it is 32 bits.
2. SHA384 : This hash function belong to hash class SHA-2, the internal block size of it is 32 bits. This is one of the truncated version.
3. SHA224 : This hash function belong to hash class SHA-2, the internal block size of it is 32 bits. This is one of the truncated version.
4. SHA512 : This hash function belong to hash class SHA-2, the internal block size of it is 64 bits.
5. SHA1 : The 160 bit hash function that resembles MD5 hash in working and was discontinued to be used seeing its security vulnerabilities.

Implementation:

```
# Python 3 code to demonstrate
# SHA hash algorithms.

import hashlib

# initializing string
str = "SIPNA COET"

# encoding SIPNA COETusing encode()
# then sending to SHA256()
result = hashlib.sha256(str.encode())

# printing the equivalent hexadecimal value.
print("The hexadecimal equivalent of SHA256 is : ")
print(result.hexdigest())

print ("\r")

# initializing string
str = "SIPNA COET"

# encoding SIPNA COETusing encode()
# then sending to SHA384()
result = hashlib.sha384(str.encode())

# printing the equivalent hexadecimal value.
print("The hexadecimal equivalent of SHA384 is : ")
print(result.hexdigest())

print ("\r")

# initializing string
str = "SIPNA COET"

# encoding SIPNA COETusing encode()
# then sending to SHA224()
result = hashlib.sha224(str.encode())

# printing the equivalent hexadecimal value.
print("The hexadecimal equivalent of SHA224 is : ")
print(result.hexdigest())

print ("\r")

# initializing string
str = "SIPNA COET"

# encoding SIPNA COETusing encode()
# then sending to SHA512()
result = hashlib.sha512(str.encode())
```

```
# printing the equivalent hexadecimal value.
print("The hexadecimal equivalent of SHA512 is : ")
print(result.hexdigest())

print ("\r")

# initializing string
str = "SIPNA COET"

# encoding SIPNA COETusing encode()
# then sending to SHA1()
result = hashlib.sha1(str.encode())

# printing the equivalent hexadecimal value.
print("The hexadecimal equivalent of SHA1 is : ")
print(result.hexdigest())
```

Output:

```
59\hello\SHA algorithms.py'
The hexadecimal equivalent of SHA256 is :
ebe3f7219907ff819e1fbae2327f25f85158114309f6cff1f48fcf3259c30784

The hexadecimal equivalent of SHA384 is :
d1e67b8819b089ec7929933b6fc1928dd64b5df31bcde6381b9d3f90488d253240490460c0a5a1a873da8236c12ef9b3

The hexadecimal equivalent of SHA224 is :
173994f389f727ca939bb185086cd7b36e66141c9e52ba0bdcfd145d

The hexadecimal equivalent of SHA512 is :
0d8fb9370a5bf7b892be4865cdf8b658a82209624e33ed71cae353b0df254a75db63d1baa35ad99f26f1b399c31f3c666a7fc67ecef3bdcdb7d60e8ada
90b722

The hexadecimal equivalent of SHA1 is :
4175a37afd561152fb60c305d4fa6026b7e79856
```

CONCLUSION: Thus we have implemented various SHA algorithms.

Key(d = 2011)

Sipna College of Engineering & Technology, Amravati.
Department of Computer Science & Engineering
Session 2022-2023

Branch :- Computer Sci. & Engg. **Class :- Final Year**
Subject :-Block Chain Fundamentals Lab manual **Sem :- VII**
Teacher Manual

PRACTICAL NO 4

AIM: Implement RSA Encryption and Decryption

S/W REQUIRED: Phython

Rivest–Shamir–Adleman(RSA)

RSA abbreviation is Rivest–Shamir–Adleman. This algorithm is used by many companies to encrypt and decrypt messages. It is an asymmetric cryptographic algorithm which means that there are two different keys i.e., the public key and the private key. This is also known as public-key cryptography because one of the keys can be given to anyone. Companies such as Acer, Asus, HP, Lenovo, etc., use encryption techniques in their products.

How does RSA algorithm work?
let us learn the mechanism behind RSA algorithm by considering one example :

1. **Generating Public Key :**

   Select two prime no's. Suppose P = 53 and Q = 59.
   Now First part of the Public key : n = P*Q = 3127.

   We also need a small exponent say e :
   But e Must be

   An integer.

   Not be a factor of n.

   $1 < e < \Phi(n)$ [$\Phi(n)$ is discussed below],
   Let us now consider it to be equal to 3.

   Our Public Key is made of n and e

2. **Generating Private Key :**

   We need to calculate $\Phi(n)$:
   Such that $\Phi(n) = (P-1)(Q-1)$
   so, $\Phi(n) = 3016$

   Now calculate Private Key, d :
   $d = (k*\Phi(n) + 1) / e$ for some integer k
   For k = 2, value of d is 2011.

3. **Now we are ready with our – Public Key (n = 3127 and e = 3) and Private Key(d = 2011)**

4. **Now we will encrypt "HI" :**

Convert letters to numbers : H = 8 and I = 9

Thus Encrypted Data c = 89e mod n.
Thus our Encrypted Data comes out to be 1394

5. **Now we will decrypt 1394 :**

Decrypted Data = cd mod n.
Thus our Encrypted Data comes out to be 89

8 = H and I = 9 i.e. "HI".

Implementation:

```
import math

print("RSA ENCRYPTOR/DECRYPTOR")
print("***************************************************")

#Input Prime Numbers
print("PLEASE ENTER THE 'p' AND 'q' VALUES BELOW:")
p = int(input("Enter a prime number for p: "))
q = int(input("Enter a prime number for q: "))
print("***************************************************")

#Check if Input's are Prime
'''THIS FUNCTION AND THE CODE IMMEDIATELY BELOW THE FUNCTION CHECKS WHETHER
THE INPUTS ARE PRIME OR NOT.'''
def prime_check(a):
    if(a==2):
        return True
    elif((a<2) or ((a%2)==0)):
        return False
    elif(a>2):
        for i in range(2,a):
            if not(a%i):
                return false
    return True

check_p = prime_check(p)
check_q = prime_check(q)
while(((check_p==False)or(check_q==False))):
    p = int(input("Enter a prime number for p: "))
    q = int(input("Enter a prime number for q: "))
    check_p = prime_check(p)
    check_q = prime_check(q)
```

```
#RSA Modulus
'''CALCULATION OF RSA MODULUS 'n'.'''
n = p * q
print("RSA Modulus(n) is:",n)

#Eulers Toitent
'''CALCULATION OF EULERS TOITENT 'r'.'''
r= (p-1)*(q-1)
print("Eulers Toitent(r) is:",r)
print("**************************************************")

#GCD
'''CALCULATION OF GCD FOR 'e' CALCULATION.'''
def egcd(e,r):
  while(r!=0):
    e,r=r,e%r
  return e

#Euclid's Algorithm
def eugcd(e,r):
  for i in range(1,r):
    while(e!=0):
      a,b=r//e,r%e
      if(b!=0):
        print("%d = %d*(%d) + %d"%(r,a,e,b))
      r=e
      e=b

#Extended Euclidean Algorithm
def eea(a,b):
  if(a%b==0):
    return(b,0,1)
  else:
    gcd,s,t = eea(b,a%b)
    s = s-((a//b) * t)
    print("%d = %d*(%d) + (%d)*(%d)"%(gcd,a,t,s,b))
    return(gcd,t,s)

#Multiplicative Inverse
def mult_inv(e,r):
  gcd,s,_=eea(e,r)
  if(gcd!=1):
    return None
  else:
    if(s<0):
      print("s=%d. Since %d is less than 0, s = s(modr), i.e., s=%d."%(s,s,s%r))
    elif(s>0):
      print("s=%d."%(s))
    return s%r

#e Value Calculation
'''FINDS THE HIGHEST POSSIBLE VALUE OF 'e' BETWEEN 1 and 1000 THAT MAKES (e,r)
COPRIME.'''
```

```
for i in range(1,1000):
    if(egcd(i,r)==1):
        e=i
print("The value of e is:",e)
print("**********************************************************")

#d, Private and Public Keys
'''CALCULATION OF 'd', PRIVATE KEY, AND PUBLIC KEY.'''
print("EUCLID'S ALGORITHM:")
eugcd(e,r)
print("END OF THE STEPS USED TO ACHIEVE EUCLID'S ALGORITHM.")
print("**********************************************************")
print("EUCLID'S EXTENDED ALGORITHM:")
d = mult_inv(e,r)
print("END OF THE STEPS USED TO ACHIEVE THE VALUE OF 'd'.")
print("The value of d is:",d)
print("**********************************************************")
public = (e,n)
private = (d,n)
print("Private Key is:",private)
print("Public Key is:",public)
print("**********************************************************")

#Encryption
'''ENCRYPTION ALGORITHM.'''
def encrypt(pub_key,n_text):
    e,n=pub_key
    x=[]
    m=0
    for i in n_text:
        if(i.isupper()):
            m = ord(i)-65
            c=(m**e)%n
            x.append(c)
        elif(i.islower()):
            m= ord(i)-97
            c=(m**e)%n
            x.append(c)
        elif(i.isspace()):
            spc=400
            x.append(400)
    return x


#Decryption
'''DECRYPTION ALGORITHM'''
def decrypt(priv_key,c_text):
    d,n=priv_key
    txt=c_text.split(',')
    x=''
    m=0
    for i in txt:
        if(i=='400'):
```

```
        x+=' '
    else:
        m=(int(i)**d)%n
        m+=65
        c=chr(m)
        x+=c
    return x

#Message
message = input("What would you like encrypted or decrypted?(Separate numbers with ',' for decryption):")
print("Your message is:",message)

#Choose Encrypt or Decrypt and Print
choose = input("Type '1' for encryption and '2' for decryption.")
if(choose=='1'):
    enc_msg=encrypt(public,message)
    print("Your encrypted message is:",enc_msg)
    print("Thank you for using the RSA Encryptor. Goodbye!")
elif(choose=='2'):
    print("Your decrypted message is:",decrypt(private,message))
    print("Thank you for using the RSA Encryptor. Goodbye!")
else:
    print("You entered the wrong option.")
    print("Thank you for using the RSA Encryptor. Goodbye!")
```

Output:

```
RSA ENCRYPTOR/DECRYPTOR
**************************************************
PLEASE ENTER THE 'p' AND 'q' VALUES BELOW:
Enter a prime number for p: 3
Enter a prime number for q: 5
**************************************************
RSA Modulus(n) is: 15
Eulers Toitent(r) is: 8
**************************************************
The value of e is: 999
**************************************************
EUCLID'S ALGORITHM:
8 = 0*(999) + 8
999 = 124*(8) + 7
8 = 1*(7) + 1
END OF THE STEPS USED TO ACHIEVE EUCLID'S ALGORITHM.
**************************************************
EUCLID'S EXTENDED ALGORITHM:
1 = 8*(1) + (-1)*(7)
1 = 999*(-1) + (125)*(8)
s=-1. Since -1 is less than 0, s = s(modr), i.e., s=7.
END OF THE STEPS USED TO ACHIEVE THE VALUE OF 'd'.
The value of d is: 7
**************************************************
Private Key is: (7, 15)
Public Key is: (999, 15)
**************************************************
```

CONCLUSION: Thus we have implemented RSA Encryption and Decryption.