VIT®

Vellore Institute of Technology

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

WEB-SECURITY

Project Report

TITLE

"FILE SHARING USING ACCESS CONTROL"

Under the guidance of

Prof. ARIVOLI A

Submitted by –

SAURABH AMBAR (17BCI0058)

KUMAR ABHISHEK (17BCI0145)

VISHAL SAINI (17BCI0188)

TABLE OF CONTENTS

Introduction: -

Access control is a fundamental component of security compliance programs that ensures security technology and access control policies are in place to protect confidential information, such as customer data.

The goal of access control is to minimize the risk of unauthorized access to physical and logical systems.

Aim of the Proposed Work: -

In present cloud service systems, the main role players are the user and the cloud administrator. At the user level the prime job of the user is to store its data on the cloud, provided by the cloud administrator. The security, encryption, authentication and all other security mechanisms are done directly by the cloud administrator.

But what if the data stored in the cloud is being misused, or if it being deleted, or being used as agent to destroy one's business. By taking this into consideration we have decided to generate a system to provide security for files in cloud. The aim is to secure the data by encrypting the data at the user level then encapsulated by the header for providing security. This data will then be sent to the cloud for storage.

Key Storage Problems: -

- **Confidentiality and Reliability of Data:**
    i. The information which we are storing can be secured by encryption which is after that many numbers of cipher text copies can be made.
- **How to transmit the key securely:**
    i. Encryption of the key may lead to again generating the key for sharing the secret key.
    ii. Replicating the secret key can lead to a problem of Duplication or compromised key.

Proposed Solution: -

The files should be encrypted before uploading/sharing into the cloud/in work space. If the file user is a single person, then it's not a big problem. Where if the file wants to be shared with the many number of people of the community then the decryption part will be more difficult where he cannot share the key with all persons. So, then we are going to use "Shamir's-Threshold Scheme" which came from threshold Cryptography.

Steps Involved: -

- The client will make the data ready to be stored in the cloud.
- The data will be protected by the help of some encryption technique, the key will only be known to the client.
- In case of it's a community or organization it's necessary to share the file at one point so, then Shamir's-Threshold Scheme Key exchange will be used.
- The encrypted data will be then transferred to the system and the system will provide a security header protected by some password which will only be known by the system.
- This encapsulated data will be sent to the cloud for storage.
- If someone tries to access the file, then the file will be blocked for a time session.

Shamir's-Threshold Scheme: -

It is a key generation and key Re-Generation technique. Firstly, it will generate the keys using the largest prime number and the number of shares and the minimum number of sub-keys required to regenerate the main secret key.

Algorithm:

- Let the minimum number of keys needed to generate the secret key be k.
- S be the secret key generated from the selected prime number p.
- Now the function for generating the points of sharing is f(x) = f0+f1x+f2x 2+· · · fk−1x k−1 where f(0) = s(secret key).
- The Key can be generated by the combining the minimum number of keys with the Lagrange Theorem.
- Formula for regenerating key:

$$f(x) = \sum_{i=1}^{k} y_i \prod_{j=1, j\neq i}^{k} \frac{x - x_j}{x_i - x_j} \pmod{p}$$

$$s = f(0) = \sum_{i=1}^{k} y_i \prod_{j=1, j\neq i}^{k} \frac{-x_j}{x_i - x_j} \pmod{p} .$$

Threshold cryptography Efficiency and Flexibility:

- Lagrange's regeneration requires O(klog^2k) Steps.
- But by using the threshold cryptography with Lagrange's theorem it requires O (k (log k -log j) ^2) where j=number of shared keys.
- Existing keys can be removed from the list without effecting the other key holders.

Literature Survey: -

Reference	Issue Addressed	Methodology/Algorithm	Advantages	Limitations
[1]	A secret key among a group of people in such a way that only well-defined combinations of people can recover the secret.	It is a key generation and key Re-Generation technique. It will generate the keys using the largest prime number and the number of shares and the minimum number of sub-keys required to regenerate the main secret key.	More than one people are required at receiver side to decrypt the data.	It is difficult for all the concerned people to be present.
[2]	A hybrid cryptography algorithm for cloud computing security.	1. Used the present hard breakable algorithms for their encrypting the file. 2. This method also includes the Secure Hash Algorithm -2 for data integrity.	1. Used the present hard breakable algorithms for their encrypting the file. 2. This method also includes the Secure Hash Algorithm -2 for data integrity.	Not proposed how to recover the file when it is shared with a community.
[3]	Enforcing Information Flow with Cryptography	1. Based on Bell-Lapadula or Biba model 2. Introducing and formally defining access control encryption (ACE)	2. By giving different rights to different users not only in terms of which messages they are allowed to receive, but also which messages they are allowed to send.	Difficult to implement.

[4]	Analysis of key-exchange protocols	1. Any key-exchange protocol that satisfies the security definition can be composed with symmetric encryption and authentication functions to provide provably secure communication channels.	1. By applying them to obtain the proof of two classes of key-exchange protocols, Diffie-Hellman and key-transport, authenticated via symmetric or asymmetric techniques.	All cryptographic techniques are not taken into consideration.
[5]	Secure file storage in cloud computing using hybrid cryptography algorithm	**1. Cryptography and steganography techniques** are more popular now a day's for data security. Use of a single algorithm is not effective for high level security to data in cloud computing **2. AES, blowfish, RC6** and BRA algorithms are used to provide block wise security to data. All algorithm key size is 128 bits.	1. LSB steganography technique is introduced for key information security 2. Each and every part of file are encrypted using different algorithm. All parts of file are encrypted simultaneously with the help of multithreading technique.	1.Splitting the file parts may lead to loss of data in the files. 2. No recovery technique
[6]	Secure File Sharing Mechanism and Key Management for Mobile Cloud Computing Environment	1. Propose a secure file storing and retrieving mechanism to avoid the limitations in existing systems like, file encryption, access rights and key management. 2. Asymmetric key cryptography is utilized to protect the data and retrieval of the data with minimal access rights.	Privacy of the mobile users are protected from the malicious insiders along with the preservation of confidentiality and integrity of the files being accessed.	RSA has limitations in handling large sizes files, whereas, ElGamal and Paillier algorithms are suitable for encryption and decryption of large file sizes.

[7]	A Hybrid Cloud Approach for Secure Authorized Deduplication	The convergent encryption technique has been proposed to encrypt the data before outsourcing.	1. Proposed process to reduce the storage and the bandwidth. 2. several new deduplication constructions supporting authorized duplicate check in hybrid cloud architecture.	Problem of authorized data deduplication

SOFTWARE AND HARDWARE: -

- Python language
- VS Code

Overview of proposed work: -

- Firstly, the admin of the file will be generating the keys that to be shared with the users.
- Admin will limit the minimum number of sub shared keys needed to generate the secret key.
- So, that if anyone wants to use it in wrong way, they need the minimum number of keys.
- Now if anyone wants to decrypt the file then he will be sending the request to the others.
- And they will be sharing their index value of the person and their sub secret key using RSA Algorithm.
- When the user got the minimum number of keys then the user can generate the secret key.
- Now the encrypted file is taken, and the encrypted data is retrieved into a variable and the alternative character are sent into the different stacks.
- Then the stacks will be getting together with their index values and added up with the key and divided by the index value of the alternate value.
- The decrypted file will be generated.

Code: Key Generation using Threshold's Algorithm:

```
from __future__ import division
from __future__ import print_function
import random
import time
import functools
#=======Code to key Generation using threahold's algorithmn==========#

_PRIME = int(input("Enter the Prime Number: "))
# 2**521 - 1
_RINT = functools.partial(random.SystemRandom().randint, 0)
def keygen():



    def _eval_at(poly, x, prime):
        accum = 0
        for coeff in reversed(poly):
            accum *= x
            accum += coeff
            accum %= prime
        return accum

    def make_random_shares(minimum, shares, prime=_PRIME):

        if minimum > shares:
            raise ValueError("pool secret would be irrecoverable")
        poly = [_RINT(prime) for i in range(minimum)]
```

```
    points = [(i, _eval_at(poly, i, prime))
              for i in range(1, shares + 1)]
    return poly[0], points

def _extended_gcd(a, b):
    x = 0
    last_x = 1
    y = 1
    last_y = 0
    while b != 0:
        quot = a // b
        a, b = b, a%b
        x, last_x = last_x - quot * x, x
        y, last_y = last_y - quot * y, y
    return last_x, last_y

def _divmod(num, den, p):

    inv, _ = _extended_gcd(den, p)
    return num * inv

def _lagrange_interpolate(x, x_s, y_s, p):

    k = len(x_s)
    assert k == len(set(x_s)), "points must be distinct"
    def PI(vals):
        accum = 1
        for v in vals:
```

```
            accum *= v
        return accum
    nums = []
    dens = []
    for i in range(k):
        others = list(x_s)
        cur = others.pop(i)
        nums.append(PI(x - o for o in others))
        dens.append(PI(cur - o for o in others))
    den = PI(dens)
    num = sum([_divmod(nums[i] * den * y_s[i] % p, dens[i], p)
               for i in range(k)])
    return (_divmod(num, den, p) + p) % p

def recover_secret(shares, prime=_PRIME):

    if len(shares) < 2:
        raise ValueError("need at least two shares")
    x_s, y_s = zip(*shares)
    return _lagrange_interpolate(0, x_s, y_s, prime)

def maingen():

    n=int(input("Enter the minimum keys required: "))
    s=int(input("Enter the no.of shares required: "))
    secret, shares = make_random_shares(minimum=n, shares=s)

    print('Generated secret Code is: ',secret)
```

```
        print('shares:')

        print(shares)
    maingen()


def secgen():
    def _extended_gcd(a, b):
        x = 0
        last_x = 1
        y = 1
        last_y = 0
        while b != 0:
            quot = a // b
            a, b = b, a%b
            x, last_x = last_x - quot * x, x
            y, last_y = last_y - quot * y, y
        return last_x, last_y
    def _divmod(num, den, p):

        inv, _ = _extended_gcd(den, p)
        return num * inv

    def _lagrange_interpolate(x, x_s, y_s, p):

        k = len(x_s)
        assert k == len(set(x_s)), "points must be distinct"
        def PI(vals):
```

```
        accum = 1
        for v in vals:
            accum *= v
        return accum
    nums = []
    dens = []
    for i in range(k):
        others = list(x_s)
        cur = others.pop(i)
        nums.append(PI(x - o for o in others))
        dens.append(PI(cur - o for o in others))
    den = PI(dens)
    num = sum([_divmod(nums[i] * den * y_s[i] % p, dens[i], p)
               for i in range(k)])
    return (_divmod(num, den, p) + p) % p

def recover_secret(shares, prime=_PRIME):

    if len(shares) < 2:
        raise ValueError("need at least two shares")
    x_s, y_s = zip(*shares)
    return _lagrange_interpolate(0, x_s, y_s, prime)

def mainsec():
    ns=int(input("Enter the no.of subset share keys u got: "))
    rs=[]
    for i in range(ns):
        m=int(input("Enter the i value of the person: "))
```

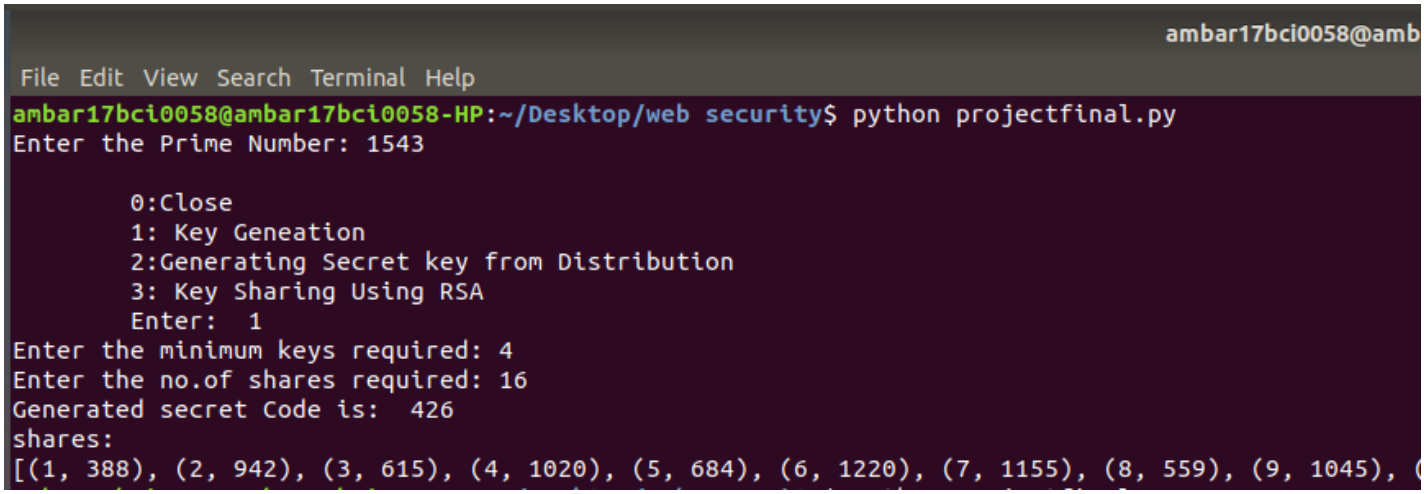
```
            s=int(input("Enter his secret key: "))
            p=(m,s)
            rs.append(p)
        print('secret recovered from  subset of share key:            ', recover_secret(
rs[:ns]))
    mainsec()


def rsa():
    alpha=["a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p","q"," r",
"s","t","u","v","w","x","y","z",",","0","1","2","3","4","5","6","7","8","9"]
    pt=str(input("Enter the data to be encypted:"))
    pl=len(pt)
    p1=int(input("Enter the first prime number: "))
    p2=int(input("Enter the second prime number: "))
    n=p1*p2
    relprime=[]
    enc=[]
    dt=[]
    dec=[]
    relprime.append(1)
    pin=(p1-1)*(p2-1)
    for i in range(2,pin):
        if(pin%i!=0):
            if(i!=pin):
                e=i
                break
    for a in range(2,pin+1):
        k=0
        for i in range(2,a//2+1):
```

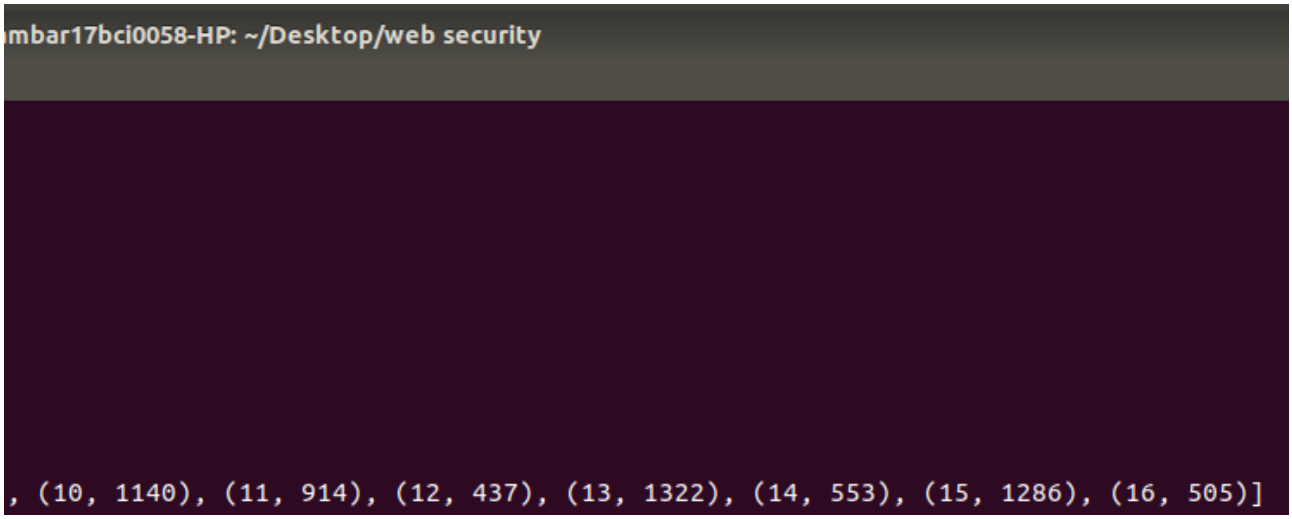
```
            if(a%i==0):
                k=k+1
        if(k<=0):
            relprime.append(a)
    l=len(relprime)
    for i in range(l):
        if((relprime[i]*e)%pin==1):
            d=relprime[i]
            break
    for i in range(pl):
        for j in range(37):
            if(pt[i]==alpha[j]):
                c=(pow(j,e))%n
                enc.append(c)
    e=''.join(map(str, enc))
    print("Encrypted data is: ",e)
    encl=len(enc)
    for i in range(encl):
        p=(pow(enc[i],d))%n
        dec.append(p)
    for i in range(len(dec)):
        dt.append(alpha[dec[i]])
        f=''.join(map(str, dt))

    print("Decypted data is:",f)
    main()

def main():
```

```
    p=int(input('''
    0:Close
    1: Key Geneation
    2:Generating Secret key from Distribution
    3: Key Sharing Using RSA
    Enter:  ''' ))

    if p==1:
        keygen()
    elif p==2:
        secgen()
    elif p==3:
        rsa()
    else:
        print("The program is closing!!!!")
        time.sleep(3)

main()
```

OUTPUT:

1. Key Generation:

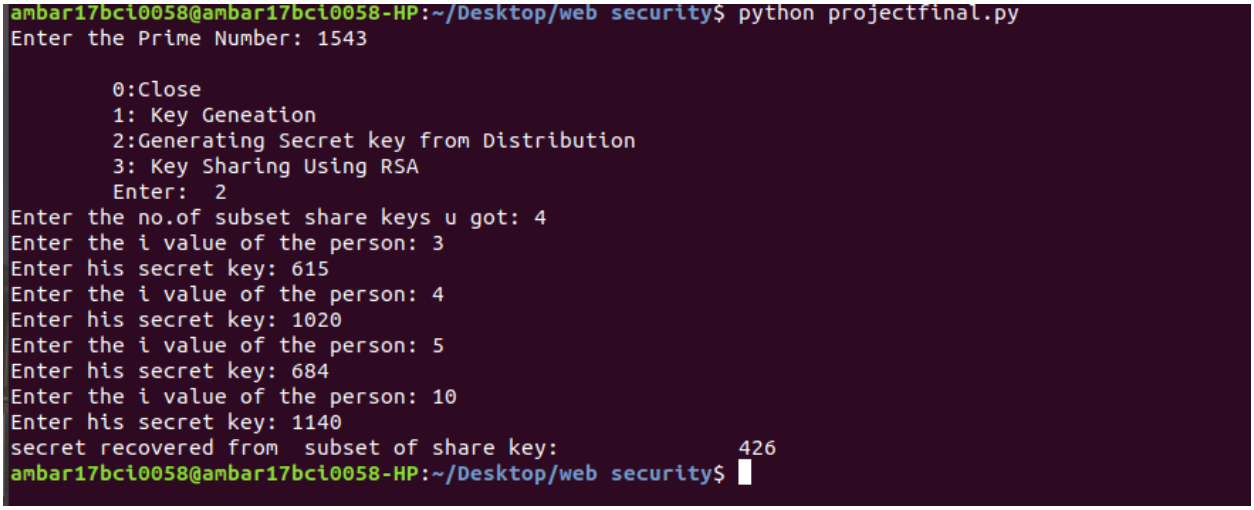




- Single Generated secret code is divided into **16 sub-keys**, in which **minimum 4 sub keys** are required to generate the secret code.

- We can observe that the secret key generated value: 426

2. Generating Secret Key From Distribution



- The value of the generated key from the distribution: 426
- It is same as the Generated Secret code Value.

Conclusion: -

This approach enables us to provide better authentication towards the shared data resources and no data breach. Organizations can easily approach this concept for file sharing among its client so that no one can misuse internal assets for false purposes.

References:

[1] Shamir's Threshold Scheme

https://link.springer.com/referenceworkentry/10.1007%2F978-1-4419-5906-5_390

[2] Divya Prathana Timothy ; Ajit Kumar Santra proposed their work on "A hybrid cryptography algorithm for cloud computing security" on Ieee xplore.

https://ieeexplore.ieee.org/document/8211728

[3] Access Control Encryption: "Enforcing Information Flow with Cryptography"

https://link.springer.com/chapter/10.1007/978-3-662-53644-5_21

[4] Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels

https://link.springer.com/chapter/10.1007/3-540-44987-6_28

[5] Punam V. Maitri ; Aruna Verma on "Secure file storage in cloud computing using hybrid cryptography algorithm"

https://ieeexplore.ieee.org/document/7566416

[6] Secure File Sharing Mechanism and Key Management for Mobile Cloud Computing Environment

https://www.researchgate.net/publication/313370436_Secure_File_Sharing_Mechanism_and_Key_Management_for_Mobile_Cloud_Computing_Environment

[7] Jin Li ; Yan Kit Li ; Xiaofeng Chen ; Patrick P.C. Lee ; Wenjing Lou on "A Hybrid Cloud Approach for Secure Authorized Deduplication"

https://ieeexplore.ieee.org/document/6802424