Somaiya Vidyavihar University
K. J. Somaiya College of Engineering
Department of Computer Engineering

Batch: B1 Roll No.: 16010121045

Experiment No. 4

Title: RSA applications using OPENSSL

Objective: Strengths and applications of RSA.

Expected Outcome of Experiment: To implement Cryptanalysis Tools .

CO	Outcome
CO4	Illustrate and Compare network security mechanisms

Books/ Journals/ Websites referred:

https://www.youtube.com/watch?v=bfKwizfuuOU

https://deepaksharma.moodlecloud.com/login/index.php

Abstract:-

CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) is a type of security measure known as challenge-response authentication. CAPTCHA helps protect you from spam and password decryption by asking you to complete a simple test that proves you are human and not a computer trying to break into a password protected account.

A CAPTCHA test is made up of two simple parts: a randomly generated sequence of letters and/or numbers that appear as a distorted image, and a text box. To pass a the test and prove your human identity, simply type the characters you see in the image into the text box.

In this experiment we understand the algorithm to generate CAPTCHA and implement it in any programming language. After implementation, we give a quiz to test our understanding of CAPTCHA

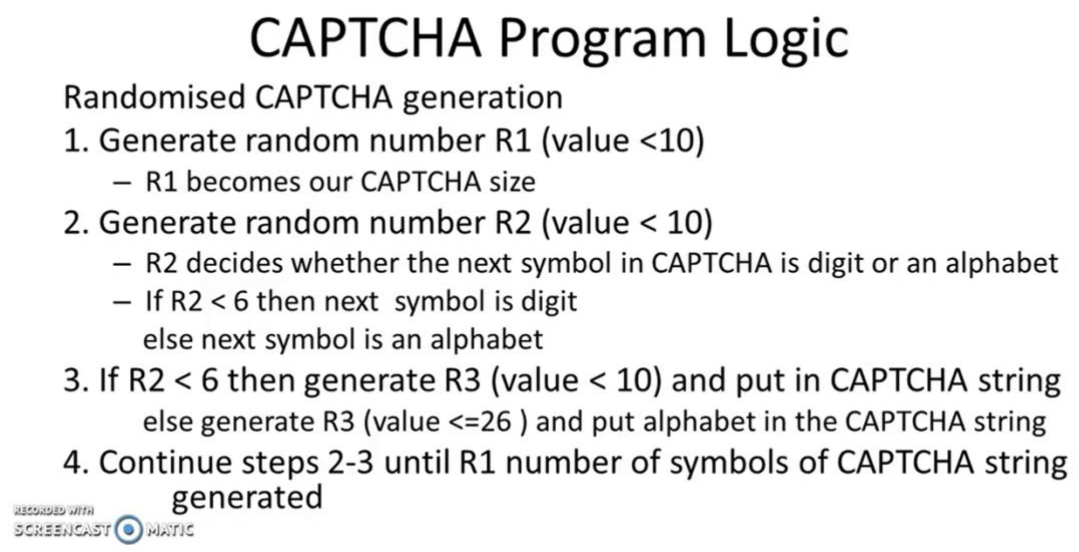
Related Theory: -

CAPTCHA stands for the Completely Automated Public Turing test to tell Computers and Humans Apart. CAPTCHAs are tools you can use to differentiate between real users and automated users, such as bots. CAPTCHAs provide challenges that are difficult for computers to perform but relatively easy for humans. For example, identifying stretched letters or numbers, or clicking in a specific area.

CAPTCHAs are used by any website that wishes to restrict usage by bots. Specific uses include:

- Maintaining poll accuracy—CAPTCHAs can prevent poll skewing by ensuring that each vote is entered by a human. Although this does not limit the overall number of votes that can be made, it makes the time required for each vote longer, discouraging multiple votes.
- Limiting registration for services—services can use CAPTCHAs to prevent bots from spamming registration systems to create fake accounts. Restricting account creation prevents waste of a service’s resources and reduces opportunities for fraud.
- Preventing ticket inflation—ticketing systems can use CAPTCHA to limit scalpers from purchasing large numbers of tickets for resale. It can also be used to prevent false registrations to free events.
- Preventing false comments—CAPTCHAs can prevent bots from spamming message boards, contact forms, or review sites. The extra step required by a CAPTCHA can also play a role in reducing online harassment through inconvenience.

The algorithm to generate a CAPTCHA:



CAPTCHA Program Logic

Randomised CAPTCHA generation

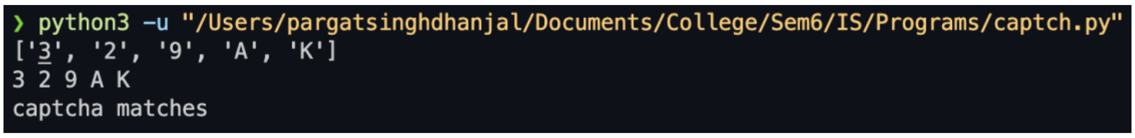
1. Generate random number R1 (value <10)
   - R1 becomes our CAPTCHA size
2. Generate random number R2 (value < 10)
   - R2 decides whether the next symbol in CAPTCHA is digit or an alphabet
   - If R2 < 6 then next symbol is digit
     else next symbol is an alphabet
3. If R2 < 6 then generate R3 (value < 10) and put in CAPTCHA string
   else generate R3 (value <=26) and put alphabet in the CAPTCHA string
4. Continue steps 2-3 until R1 number of symbols of CAPTCHA string generated

RECORDED WITH SCREENCAST MATIC

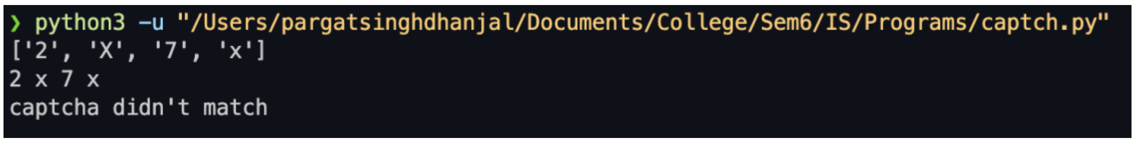
Captcha Program (Simple text based)

```
import random

import string


def generateCaptcha():

    R1=random.randint(0,9)

    captcha=[]

    for i in range(0,R1,1):

        R2=random.randint(0,9)

        if R2<6:

            R3=random.randint(0,9)

            captcha.append(str(R3))

        else:

            R3 = random.choice(string.ascii_letters)

            captcha.append(R3)


    return captcha
```

```
captcha=generateCaptcha()
print(captcha)

x=[]

x=input().split( )
if(x==captcha):
    print("captcha matches")
else:
    print("captcha didn't match")
```



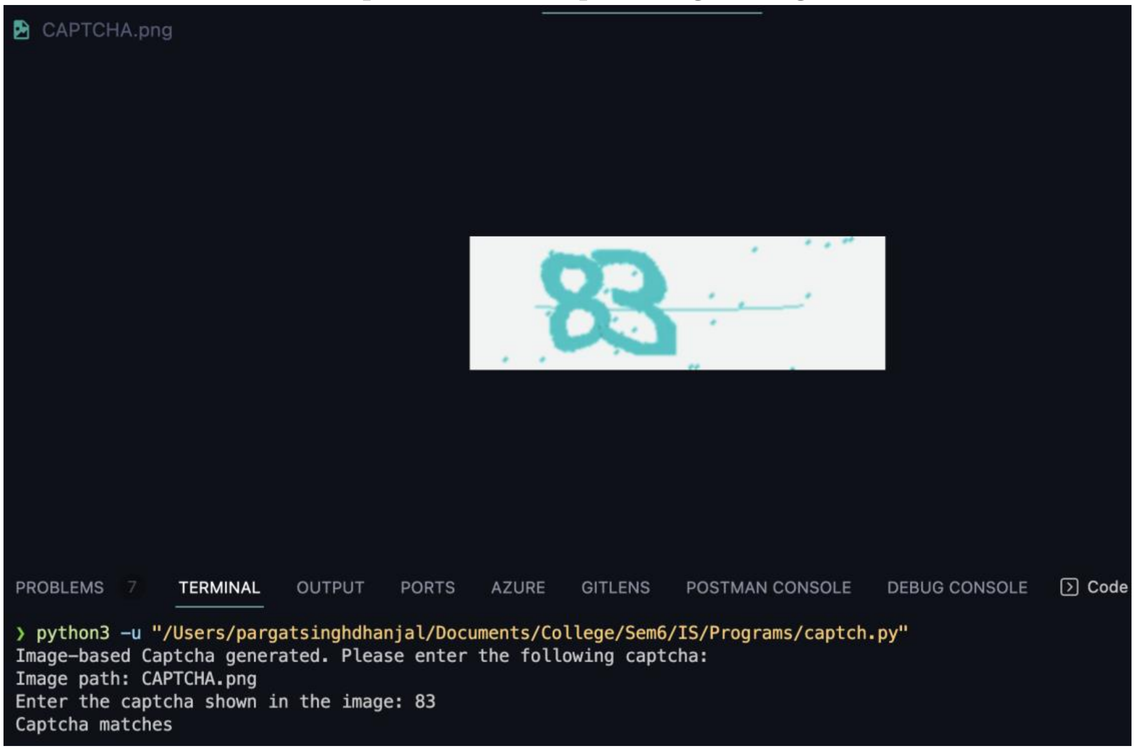
```
❯ python3 -u "/Users/pargatsinghdhanjal/Documents/College/Sem6/IS/Programs/captch.py"
['3', '2', '9', 'A', 'K']
3 2 9 A K
captcha matches
```



```
❯ python3 -u "/Users/pargatsinghdhanjal/Documents/College/Sem6/IS/Programs/captch.py"
['2', 'X', '7', 'x']
2 x 7 x
captcha didn't match
```

Image based Captcha:

```
from captcha.image import ImageCaptcha
import random
import string

def generateCaptcha():
    R1 = random.randint(0, 9)
    captcha = []
    for i in range(R1):
        R2 = random.randint(0, 9)
        if R2 < 6:
            R3 = random.randint(0, 9)
            captcha.append(str(R3))
        else:
            R3 = random.choice(string.ascii_letters)
            captcha.append(R3)

    return captcha

def createImageCaptcha(captcha_text, image_path='CAPTCHA.png'):
    image = ImageCaptcha(width=280, height=90)
    data = image.generate(captcha_text)
    image.write(captcha_text, image_path)
    return image_path

captcha = generateCaptcha()
captcha_text = ''.join(captcha)
image_path = createImageCaptcha(captcha_text)

print("Image-based Captcha generated. Please enter the following captcha:")
print(f"Image path: {image_path}")

user_input = input("Enter the captcha shown in the image: ")

if user_input == captcha_text:
    print("Captcha matches")
else:
    print("Captcha didn't match")
```



```
❯ python3 -u "/Users/pargatsinghdhanjal/Documents/College/Sem6/IS/Programs/captch.py"
Image-based Captcha generated. Please enter the following captcha:
Image path: CAPTCHA.png
Enter the captcha shown in the image: 83
Captcha matches
```

Conclusion:- Learnt to implement captcha and learnt it's working.

Postlab questions:
9.1 Discuss how CAPTCHA helps in improving web security.

CAPTCHA, which stands for Completely Automated Public Turing test to tell Computers and Humans Apart, is a security measure designed to distinguish between human users and automated bots on the internet. Its primary purpose is to enhance web security by preventing automated programs from performing certain actions that could be harmful or disruptive. Here are some ways CAPTCHA contributes to improving web security:

- **Bot Prevention:** CAPTCHA challenges are designed to be easy for humans to solve but difficult for automated scripts to decipher. This helps in preventing bots from carrying out actions such as submitting forms, creating accounts, or conducting activities that could lead to spam or unauthorized access.
- **Authentication:** CAPTCHA is often used as an additional layer of authentication during login processes. By requiring users to complete a CAPTCHA, websites can ensure that the entity trying to log in is likely a human rather than a malicious script attempting to gain unauthorized access.

- **Protection against Brute Force Attacks:** CAPTCHA can be employed to mitigate brute force attacks where an automated script tries to guess passwords or access codes. By introducing a CAPTCHA challenge after a certain number of failed login attempts, the system can thwart these automated attacks.
- **Prevention of Scraping:** CAPTCHA helps in preventing data scraping by requiring user interaction to access or retrieve information from a website. This makes it more challenging for automated bots to collect data at scale.

9.2 What are the different types of CAPTCHA? Is re-CAPTCHA different from CAPTCHA? Justify your answer.

There are various types of CAPTCHA, each with its own set of challenges for distinguishing humans from bots. Some common types include:

- **Image-based CAPTCHA:** Users are required to identify objects, characters, or patterns within an image.
- **Text-based CAPTCHA:** Users must enter characters displayed in a distorted or obscured way to prove they are human.
- **Mathematical CAPTCHA:** Users solve a simple mathematical problem to prove their human identity.
- **Checkbox CAPTCHA:** Users click a checkbox to confirm they are not a robot.

reCAPTCHA, on the other hand, is a specific implementation of CAPTCHA developed by Google. It is designed to be more user-friendly and efficient in distinguishing humans from bots. The key difference lies in the way reCAPTCHA employs advanced algorithms and machine learning to analyze user behavior and make determinations. While traditional CAPTCHAs may rely solely on distorted characters or images, reCAPTCHA adapts its challenges based on user behavior, making it more dynamic and often less intrusive for legitimate users.

9.3 List the limitations of CAPTCHA. Mention the alternatives to CAPTCHA which can overcome these limitations.

CAPTCHA, while effective, has its limitations:

- **Accessibility:** CAPTCHAs can be challenging for users with visual or cognitive impairments, potentially excluding a portion of the population.
- **User Experience:** Some CAPTCHAs, especially those that are too complex or difficult, can frustrate users and result in a negative experience.
- **Advancements in Automation:** As AI and machine learning technologies advance, there is an ongoing cat-and-mouse game where more sophisticated bots can sometimes bypass traditional CAPTCHAs.

Alternatives to CAPTCHA that address these limitations include:

- **Biometric Authentication:** Leveraging fingerprint, face recognition, or other biometric data can provide a more user-friendly and secure alternative.
- **Behavioral Analysis:** Analyzing user behavior and interaction patterns on a website can help identify bots without requiring explicit user input.
- **Honeypots:** Including hidden form fields that only bots are likely to fill out can be an effective way to detect automated submissions.

- **Time-based Challenges:** Introducing time-based challenges or delays can thwart automated scripts that aim to quickly perform actions on a website.
- **Two-Factor Authentication (2FA):** Requiring users to verify their identity through a secondary method, such as a mobile app or SMS, adds an extra layer of security.

Choosing the right solution often involves a trade-off between security and user experience, and combining multiple methods can provide a more robust defense against automated threats.