Course Name:	**Applied Cryptography**	**Semester:**	**V**
Date of Performance:	**__11_ / __10_ / ___25___**	**DIV/ Batch No:**	**D-2**
Student Name:	**Shreyans Tatiya**	**Roll No:**	**16010123325**

Experiment No: 7

Title: Implementation of authentication algorithms

Aim and Objective of the Experiment:

Students should generate Captcha and one of the following authentication mechanisms. Students may use libraries to implement the experiment.

- **CAPTCHA Integration**: Create a CAPTCHA system to differentiate between human users and bots. Students can generate simple text based or math based captchas.
  Use video tutorial: https://www.youtube.com/watch?v=bfKwizfuuOU
- **One-Time Pads (OTP)**: Implement a one-time pad system that generates a series of unique codes for users during login. When executed on two diff systems, the onetime pad sequence has to be the same.
- **Challenge-Response Systems**: Design a challenge-response mechanism where the server issues a challenge (e.g., a random number or phrase), and the user must respond correctly based on a shared secret key.
- **Password-Based Authentication with Salt**: Accept password in plain, generate a random salt. Store salt and hash (encrypted password + salt) in the table. When a password is entered, use the same encryption algorithm, use the corresponding salt stored, compute hash function and allow only when there's a match. Students may use any encryption algorithm.

COs to be achieved:

CO4: Understand Authentication Mechanisms and Evaluate Cryptographic Hash Functions

Books/ Journals/ Websites referred:

1. https://www.youtube.com/watch?v=bfKwizfuuOU
2. **https://rajeevkdave.medium.com/how-to-generate-captcha-in-python-72fc5c0a73de**

Theory:

-Need of authentication

Authentication is needed to **verify the identity of a user or system** before allowing access to data or resources.

It ensures that only **authorized users** can perform actions, protecting systems from **unauthorized access, data theft, and misuse**.
In simple terms, it confirms *"you are who you claim to be."*

- various means of authentication3

- **Password-based Authentication:**
User provides a username and password; simplest and most common method.

- **OTP (One-Time Password):**
A temporary code sent to a user (via app, SMS, or email) for extra security.

- **CAPTCHA-based Authentication:**
Ensures the user is human by solving an image or text challenge.

- **Biometric Authentication:**
Uses fingerprints, face, or voice for unique identity verification.

- **Token-based Authentication:**
Generates a secure token (like in 2FA or API systems) to confirm identity.

- **Challenge-Response Authentication:**
The system sends a challenge, and the user responds correctly using a secret key.

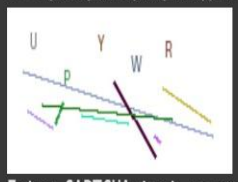
Code :

```
from PIL import Image, ImageDraw, ImageFont
import random, string
from IPython.display import display

# Step 1: Generate random CAPTCHA text
def generate_captcha_text(length=5):
    letters = string.ascii_uppercase + string.digits
    return ''.join(random.choice(letters) for _ in range(length))

# Step 2: Create CAPTCHA image
def create_captcha_image(captcha_text):
    width, height = 200, 80
    image = Image.new('RGB', (width, height), color=(255, 255, 255))
    draw = ImageDraw.Draw(image)
```

```
    # Colab doesn't have system fonts
    try:
        font = ImageFont.truetype("arial.ttf", 40)
    except:
        font = ImageFont.load_default()

    # Draw characters with slight random jitter
    x = 20
    for char in captcha_text:
        y = random.randint(10, 30)
        draw.text((x, y), char, font=font,
                  fill=(random.randint(0, 150),
                        random.randint(0, 150),
                        random.randint(0, 150)))
        x += 30

    # Add random noise lines
    for i in range(8):
        x1, y1 = random.randint(0, width), random.randint(0, height)
        x2, y2 = random.randint(0, width), random.randint(0, height)
        draw.line(((x1, y1), (x2, y2)),
                  fill=(random.randint(0, 255),
                        random.randint(0, 255),
                        random.randint(0, 255)), width=2)

    display(image)    # show directly in notebook
    return image

# Step 3: Verification
def captcha_authentication():
    print("=== CAPTCHA Authentication Demo ===")
    captcha_text = generate_captcha_text()
    create_captcha_image(captcha_text)

    user_input = input("Enter CAPTCHA text as seen above:
").strip().upper()
    if user_input == captcha_text:
        print("CAPTCHA verified successfully! You are human.")
    else:
        print(f"CAPTCHA failed! Correct text was: {captcha_text}")

captcha_authentication()
```

Output:

```
=== CAPTCHA Authentication Demo ===
```




```
Enter CAPTCHA text as seen above: UPYWR
CAPTCHA verified successfully! You are human.
```

Post Lab Subjective/Objective type Questions:

Compare and contrast various authentication methods for their vulnerabilities and strengths

Authentication Method	Description	Strengths	Vulnerabilities / Weaknesses
Password-based	User enters a secret password to log in.	Simple and widely used; easy to implement.	Weak passwords can be guessed or stolen (phishing, brute force).
CAPTCHA-based	Distinguishes humans from bots using images or puzzles.	Protects websites from automated attacks and spam.	Can be bypassed by AI/bot scripts or CAPTCHA-solving services.
OTP (One-Time Password)	Generates a unique code for each login (via app or SMS).	Strong protection even if password is stolen; short-lived codes.	Vulnerable if phone/app is compromised; phishing can still trick users.
Biometric Authentication	Uses physical traits (fingerprint, face, iris).	Very secure and user-friendly; cannot be easily shared or guessed.	Privacy concerns; spoofing possible with fake biometrics; expensive hardware.
Token-based (e.g., 2FA apps, security keys)	Uses a physical or digital token to verify user.	High security; resistant to phishing.	Tokens can be lost, stolen, or not always available.
Challenge-Response	System sends a random challenge, user responds using a secret key.	Prevents replay attacks; strong cryptographic protection.	More complex to implement; requires secure key sharing.

Conclusion:

Authentication is essential to ensure that only legitimate users can access a system. Different methods like passwords, OTPs, biometrics, and CAPTCHA offer varying levels of security and convenience.

Choosing the right authentication method depends on the required balance between **security, usability, and system needs**.