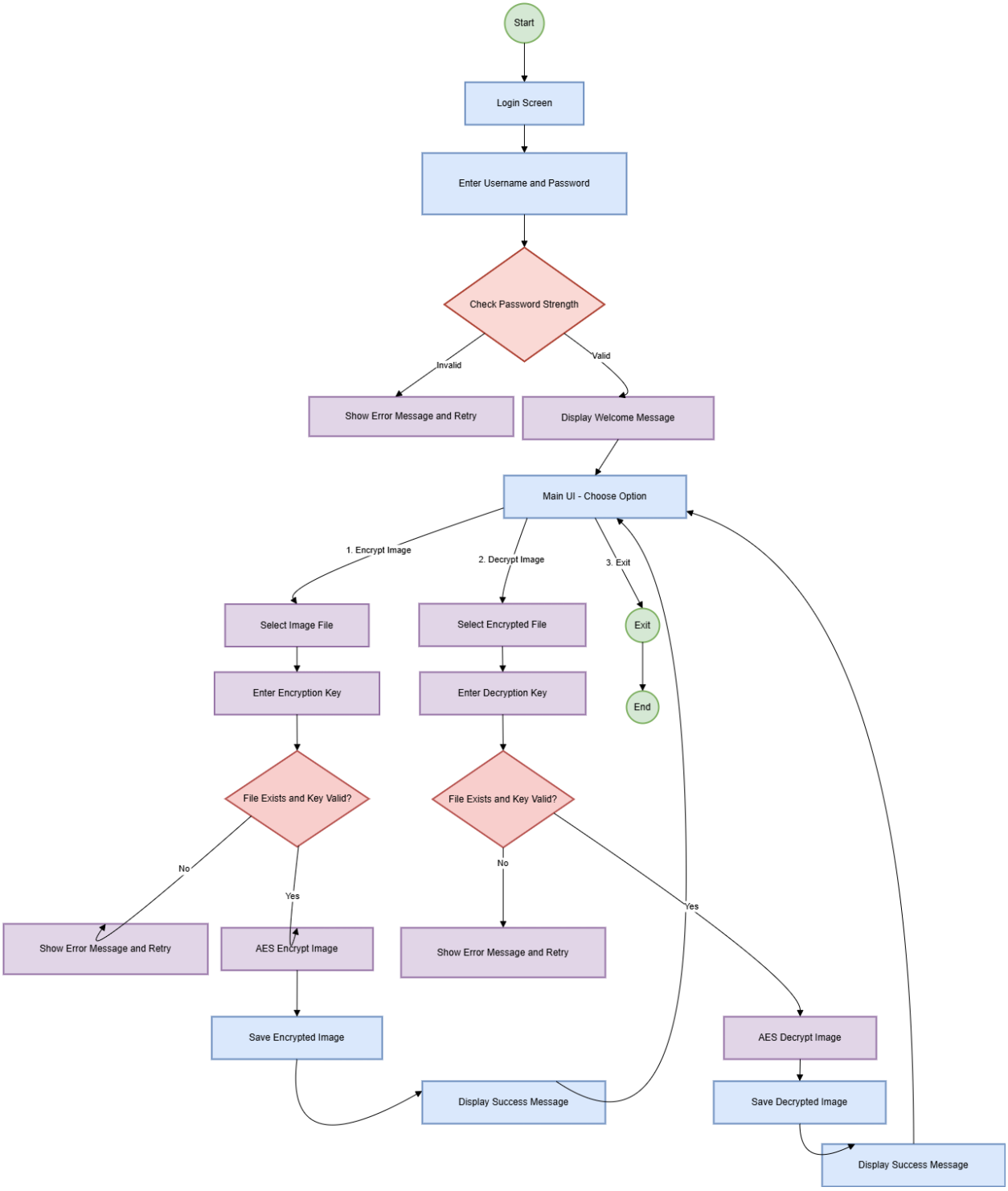
NIRMA UNIVERSITY
INSTITUTE OF TECHNOLOGY

Cyber Security Assignment

Topic: Image Encryption Using SHA-256 and AES Algorithms.

Hardini Dalwadi - 22BCE055

Priyanshu Joshi - 22BCE132


Start
Login Screen
Enter Username and Password
Check Password Strength
Invalid
Valid
Show Error Message and Retry
Display Welcome Message
Main UI - Choose Option
1. Encrypt Image
2. Decrypt Image
3. Exit
Select Image File
Select Encrypted File
Exit
Enter Encryption Key
Enter Decryption Key
End
File Exists and Key Valid?
File Exists and Key Valid?
No
Yes
No
Yes
Show Error Message and Retry
AES Encrypt Image
Show Error Message and Retry
Save Encrypted Image
AES Decrypt Image
Display Success Message
Save Decrypted Image
Display Success Message

Visual_Encryption.py

Encryption Techniques Used in the Python Script

The provided Python script employs **AES (Advanced Encryption Standard) encryption** to secure image files. AES is a symmetric encryption algorithm, which means it uses the same key for both encryption and decryption. The script utilizes AES in **CBC (Cipher Block Chaining) mode**, which enhances security by ensuring that identical plaintext blocks produce different ciphertext blocks.

Key Features of AES Encryption in the Script:

1. **SHA-256 Key Hashing**: The user-provided key is hashed using SHA-256 to generate a 256-bit encryption key.
2. **Padding Mechanism**: Since AES requires input data to be in fixed block sizes (16 bytes for AES-128), the script applies **PKCS#7 padding**.
3. **IV (Initialization Vector) Usage**: A random IV (Initialization Vector) is generated for each encryption to ensure uniqueness in ciphertext.
4. **Base64 Encoding**: The encrypted data, along with the IV, is Base64 encoded to be easily stored and shared.
5. **Secure Password Enforcement**: Before encryption or decryption, the script ensures users set a strong password.

Code Explanation

1. SHA-256 Hashing for Key Strengthening

```
def sha256(key):
    sha = hashlib.sha256()
    sha.update(key.encode("utf-8"))
```

```
    return sha.digest()
```

Detailed Explanation:

- The `sha256` function takes a user-provided key as input.
- It encodes the key into a UTF-8 byte string before hashing.
- The `hashlib.sha256()` function is used to compute a **256-bit cryptographic hash**.
- The resulting hash is used as the encryption key for AES.

---

2. Padding and Unpadding Mechanisms

```
def pad(plain_text, block):
    pad_len = block - (len(plain_text) % block)
    return plain_text + (chr(pad_len) * pad_len).encode("utf-8")
```

Detailed Explanation:

- AES requires input data to be in multiples of **16 bytes**.
- The function calculates how many padding bytes are needed (`pad_len`).
- It appends `pad_len` number of characters with the padding value.
- The result is an appropriately sized block of data.

```
def unpad(plain_text):
    return plain_text[:-ord(plain_text[-1:])]
```

Detailed Explanation:

- The last byte of the decrypted text indicates how much padding was added.
- The function removes that amount of padding to restore the original plaintext.

---

3. Image Encryption

```
def encrypt_image(file_path, key):
    with open(file_path, "rb") as f:
        image_data = f.read()

    key = sha256(key)
    iv = Random.new().read(BLOCK_SIZE)
    cipher = AES.new(key, AES.MODE_CBC, iv)
    encrypted_data = cipher.encrypt(pad(image_data, BLOCK_SIZE))

    enc_file_path = "encrypted_image.aes"
    with open(enc_file_path, "wb") as f:
        f.write(base64.b64encode(iv + encrypted_data))
```

Detailed Steps:

1. The function **reads the image file** in binary mode.
2. It **generates a 256-bit key** from the user-provided password.
3. A **random IV (Initialization Vector)** of 16 bytes is generated.

4. AES **encrypts the image data** using the CBC mode.
5. The encrypted content, along with the IV, is **Base64 encoded** and saved.

---

4. Image Decryption

```
def decrypt_image(encrypted_file_path, key):
    with open(encrypted_file_path, "rb") as f:
        encrypted_data = base64.b64decode(f.read())

    key = sha256(key)
    iv = encrypted_data[:BLOCK_SIZE]
    cipher = AES.new(key, AES.MODE_CBC, iv)
    decrypted_data = unpad(cipher.decrypt(encrypted_data[BLOCK_SIZE:]))

    dec_file_path = "decrypted_image.png"
    with open(dec_file_path, "wb") as f:
        f.write(decrypted_data)
```

Detailed Steps:

1. The function **reads and decodes the encrypted file** from Base64.
2. The **IV is extracted** from the first 16 bytes.
3. The AES cipher is initialized with the **same key and IV** used during encryption.
4. The **cipher decrypts the encrypted image data**.

5. **Padding is removed**, and the image is restored to its original state.
6. The decrypted image is **saved as a PNG file**.

---

5. Password Strength Validation

```
def check_password_strength(password):
    errors = []
    if len(password) < 8:
        errors.append("❌ Password must be at least 8 characters long.")
    if len(password) > 16:
        errors.append("❌ Password must not exceed 16 characters.")
    if not re.search(r"[A-Z]", password):
        errors.append("❌ Password must contain at least one uppercase letter.")
    if not re.search(r"[a-z]", password):
        errors.append("❌ Password must contain at least one lowercase letter.")
    if not re.search(r"[0-9]", password):
        errors.append("❌ Password must contain at least one number.")
    if not re.search(r"[!@#$%^&*()-_=+]", password):
        errors.append("❌ Password must contain at least one special character (!@#$%^&*()-_=+).")
```

Purpose:

- Ensures strong password criteria are met before allowing encryption or decryption.
- Prevents weak passwords from compromising security.

---

6. User Authentication & Menu System

```
def login():
    username = input("Enter your username: ").strip()
    if not username:
        print("❌ Username cannot be empty!")
        return False

    while True:
        password = getpass.getpass("Enter your password: ")
        if check_password_strength(password):
            print(f"\n☑ Welcome, {username}! You are now logged in.")
            return True
```

Purpose:

- Forces the user to enter valid credentials before proceeding.
- Loops until a strong password is entered.
- Once authenticated, access to encryption and decryption features is granted.

Conclusion

This Python script provides a **secure method** for encrypting and decrypting image files using AES encryption. It follows best cryptographic practices, including **SHA-256 hashing for key derivation, CBC mode for secure encryption, IVs for uniqueness, and Base64 encoding for storage**. Additionally, the script enforces strong password policies, ensuring robust protection for encrypted data.

VE_Front.py

1. Introduction

In this project, we have developed a **Secure Image Encryption Application** using Python's **Tkinter** library for the graphical user interface (GUI) and a separate module, `Visual_Encryption`, to handle the encryption and decryption of images. This application ensures that image files are securely encrypted and decrypted using a user-supplied key, with a built-in password strength checker to enhance security.

2. Features of the Application

The application provides the following functionalities:

- **User Authentication:** A login system that verifies a username and enforces password strength requirements.
- **Password Strength Checker:** Ensures the encryption key is strong enough before proceeding.
- **Image Encryption:** Securely encrypts an image file with the user-supplied key.
- **Image Decryption:** Restores an encrypted image file back to its original form.
- **Graphical User Interface (GUI):** Built using Tkinter with an enhanced theme from `ttkthemes`.

3. Modules and Libraries Used

- **tkinter:** Standard Python library for GUI applications.
- **filedialog, messagebox:** Components of Tkinter for file selection and displaying messages.

- **ttkthemes:** Provides a modern and aesthetic look for the GUI.
- **re (Regular Expressions):** Used for password validation.
- **Visual_Encryption:** A custom backend module handling encryption and decryption.

4. Code Explanation

4.1 Password Strength Checker

```
# Function to check password strength
def check_password_strength(password):
    errors = []
    if len(password) < 8:
        errors.append("Password must be at least 8 characters long.")
    if not re.search(r"[A-Z]", password):
        errors.append("Password must contain at least one uppercase letter.")
    if not re.search(r"[a-z]", password):
        errors.append("Password must contain at least one lowercase letter.")
    if not re.search(r"[0-9]", password):
        errors.append("Password must contain at least one number.")
    if not re.search(r"[!@#$%^&*()-_=+]", password):
        errors.append("Password must contain at least one special character (!@#$%^&*()-
_=+).")

    if errors:
        messagebox.showerror("Weak Password", "\n".join(errors))
        return False
    return True
```

This function enforces strong password criteria, ensuring security before allowing encryption operations.

4.2 Image Encryption

```
# Function to handle encryption
def encrypt_image():
    file_path = filedialog.askopenfilename(title="Select an Image")
    if not file_path:
        return
    key = key_entry.get()
    if not key:
        messagebox.showerror("Error", "Please enter an encryption key!")
        return

    Visual_Encryption.encrypt_image(file_path, key)
    messagebox.showinfo("Success", "Image Encrypted Successfully!")
```

This function:

1. Opens a file selection dialog to choose an image.
2. Ensures a valid encryption key is entered.
3. Calls `encrypt_image()` from the `Visual_Encryption` module.
4. Displays a success message.

4.3 Image Decryption

```
# Function to handle decryption
```

```
def decrypt_image():
    file_path = filedialog.askopenfilename(title="Select Encrypted File")
    if not file_path:
        return
    key = key_entry.get()
    if not key:
        messagebox.showerror("Error", "Please enter a decryption key!")
        return

    Visual_Encryption.decrypt_image(file_path, key)
    messagebox.showinfo("Success", "Image Decrypted Successfully!")
```

This function works similarly to encryption but calls `decrypt_image()` to reverse the process.

4.4 User Authentication System

```
# Function to handle login
def login():
    username = username_entry.get().strip()
    password = password_entry.get()
    if not username or not password:
        messagebox.showerror("Error", "Username and password cannot be empty!")
        return

    if check_password_strength(password):
        messagebox.showinfo("Success", f"Welcome, {username}! You are now logged in.")
        show_main_ui()
```

- Checks if username and password fields are not empty.
- Validates password strength before allowing access.
- Displays a success message upon successful login.

4.5 Graphical User Interface (GUI)

The GUI consists of two frames:

1. **Login Frame:** Takes user credentials.
2. **Main UI Frame:** Provides buttons for encryption and decryption.

Main Window

root = ThemedTk(theme="breeze")

root.title("🛡 Secure Image Encryptor")

root.geometry("500x600")

root.config(bg="#1f1f2e")

- Uses `ThemedTk` for a modern look.
- Configures window size and background color.

4.6 User Interface Components

tk.Label(main_frame, text="Secure Image Encryptor", font=("Arial", 18, "bold"), fg="white", bg="#1f1f2e").pack()

encrypt_btn = tk.Button(main_frame, text="🔒 Encrypt Image", font=("Arial", 12, "bold"), bg="#4CAF50", fg="white", padx=20, pady=10, command=encrypt_image)

encrypt_btn.pack(pady=10)

- Labels provide instructions.
- Buttons execute functions upon clicking.

5. Conclusion

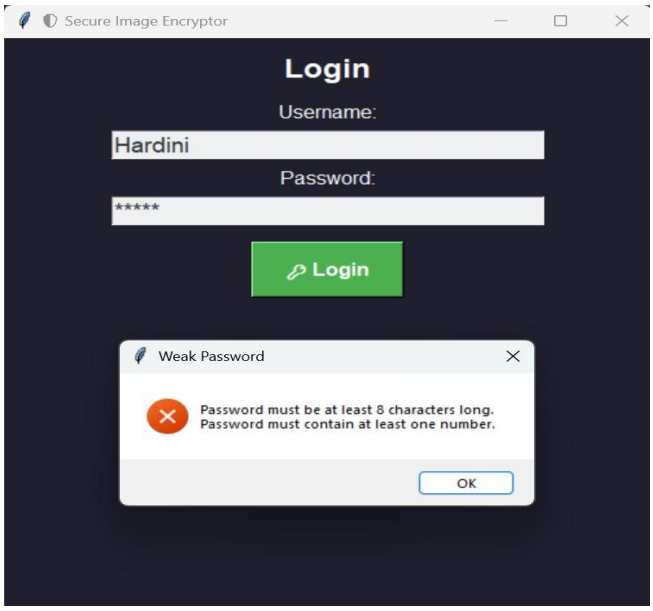
This application ensures secure encryption and decryption of image files using a user-friendly interface. It enforces password security, integrates with a backend encryption module, and provides an intuitive UI for users to interact with the system. By implementing strong encryption practices and user authentication, this project offers a practical approach to image security.

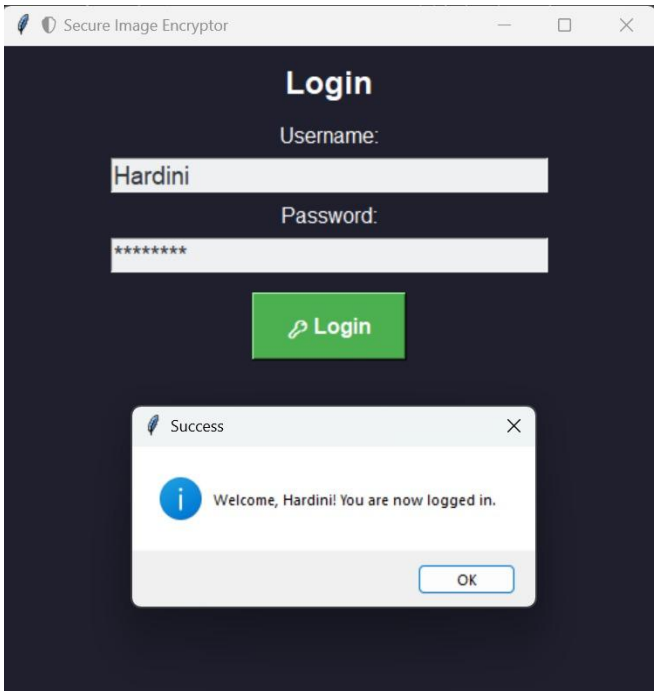
Future Enhancements:

- Implementing **AES encryption** within the script instead of a separate module.
- Adding **biometric authentication** for enhanced security.
- Introducing **cloud storage integration** for encrypted images.

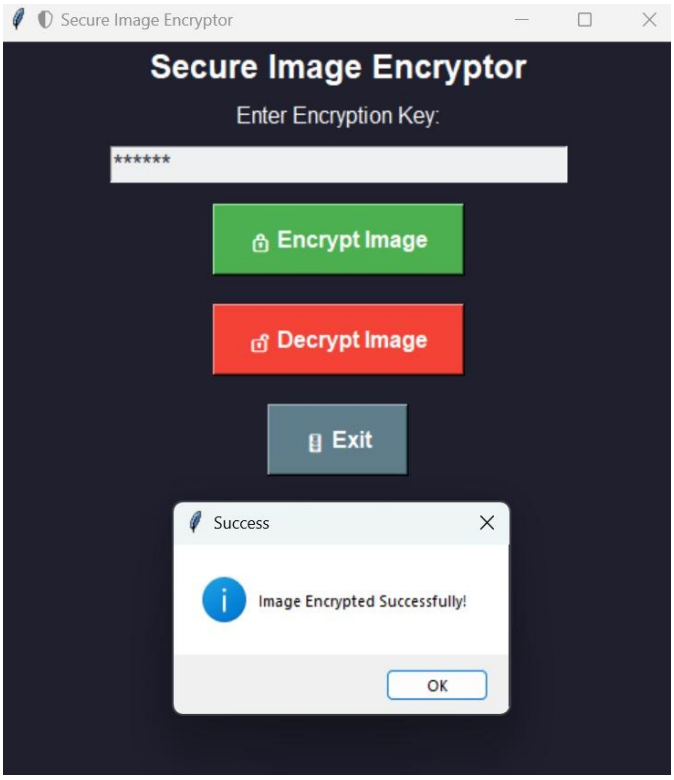
This project successfully demonstrates the **importance of security in data encryption**, making it a useful tool for protecting sensitive images.

OUTPUT


Secure Image Encryptor
Login
Username:
Hardini
Password:
*****
Login
Weak Password
Password must be at least 8 characters long.
Password must contain at least one number.
OK

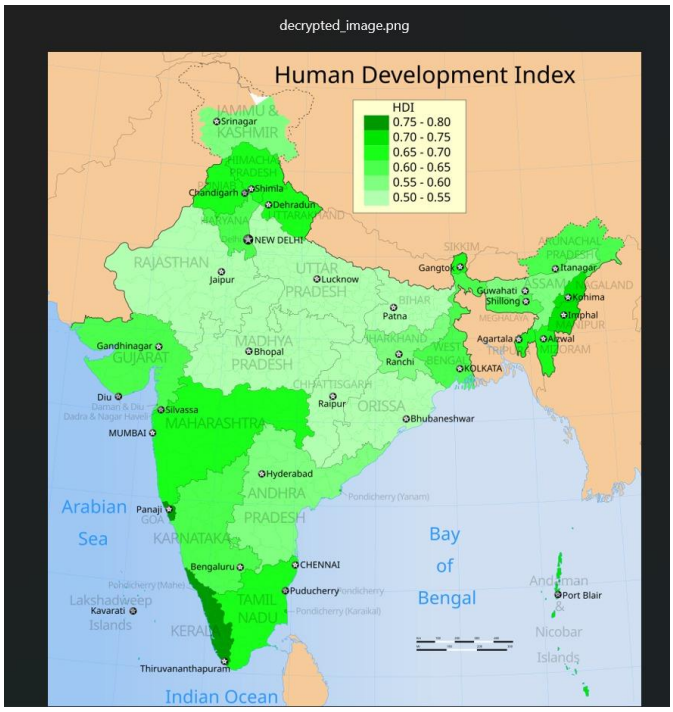

Secure Image Encryptor
Login
Username:
Hardini
Password:
****
Login
Success
Welcome, Hardini! You are now logged in.
OK


Secure Image Encryptor
Secure Image Encryptor
Enter Encryption Key:
Encrypt Image
Decrypt Image
Exit


Secure Image Encryptor
Secure Image Encryptor
Enter Encryption Key:
******
Encrypt Image
Decrypt Image
Exit
Success
Image Encrypted Successfully!
OK

encrypted_image.aes

```
1   NdFVeOovWYtun+pn7mGd/Xyrv
    +GEiTJHNGliZMWHOFaM0tFzZNqubG5wBREdGRcml9le7MQSREJ3WZuQb8jdy
    +440BOZqfUuAjxgHu6Eldc2E6dqE38y5pvy7u8E0fOhZ5VlNqdQ9XzEiOZv2Al5OB8M/
    +8OF00QQnDqOexwD2rMtvqvvMHnxvd4EzyzreMgZZy8ovMYWyze4t3fSj4SuaGTcbMl3nUeNNbkwzgNh
    ag96dMQ0YFRZ0VrPKmQfO7sJyUntACdMZsIPfVEEUj4+zxPKjrW805qSBcVyK3yZ/
    iHKyrjf61dTvi3dqIt30PMnZJA3bM8i8P8fGKdp/zV/
    1UYAcQeOpx719m27ME3xDX1FbKsZXnivdi1OM4SX8oUMBkBV6XJnoQquPjzj/Jj/4
    +OPFDoQBzznGU6SvBgxNbVarz6IM0KhdrRyI/
    hiVyfOmJvKXxrtG6NoL8TBvH4vP2p1xZvwXc23MCKKXIiXM5MJZcP4FySoImLDcUSOoar36BjqW0qI/
    aMyAW8VwlvNWfxaIKPNtS9UzsVxMmAcvxFpQKU/
    191fHIeqporkO4Iyo4cvd8GUQFZC2sCPq6BHmaUbldGgsaYutckCD84rjbMDsDEadphRwYni3/h
    +FhjFuirJuAv+lxiLDNmVUm04F8kmbomA5Ng4yRutjdW934oiPxCwlUujrhYVSvYbtraFpnF5c8FfHK
    +0RKwvsNq4xXrAJwKfhPREOoFqbQSoYRcXJOm6Z2wpjeUfXbxf9nDu4HNSuIEM8ffWzPF0kRvrNHf5AQ
    qwOkcR1drRyQBhMDxYP4YkhbWMSRZs00ftxVdJ628ZuiI2oWxV9oXguxGlDX0GsY2N5PIzaL0
    +APSAJjiyYr7Qwb8P2cGcx5ksMGC5rFrnxzf2efCKNc02C63qiv
    +S94ho444EseyeCaa52RPsnrMvhZXOdtRQtbV8I9CFr/rAWjZ2UidCvqV38A5
    +4qVmbP4fmBnFI2rNxqc4HJxMzHULXSnlflgs1Xax6ONKB0fwu5ERlu/Irl5Yx//
    IPUtqaHZSpPizQy1KYRQMnuSd1C52MC2zv4kZkoUHkbP6Lak/DtijmA1DBEw6iZK2GerxsfgBLIH+pA
    +17MakB6q2N6BZk+Y/GkgkZvTT8EPI25HPfnoyYlPCJY546mIQqI+TzMWygzv/
    os5tu0ASfwdv1qDHgQE5T0A8YMs29P5bjhFz5qCOM4EO906h6WLEa/5fVn+TDHmA7Ptce8Ey5HYK/
    vjDETsiZVm1Dr/
    YgarZM0yi9yGv58hZLomK8dyGf2cFcnJ9eqeDwOZ6mLeYO6g45Fr07SO9Q7xkDsSt1muuwXRXdI2
    +fGyOqV/V/dA8l92UWtDfBSQj1Ng0bD2lWMCvX+vkE7Tdwr9GOTTL1H/
```


decrypted_image.png
Human Development Index
HDI
0.75 - 0.80
0.70 - 0.75
0.65 - 0.70
0.60 - 0.65
0.55 - 0.60
0.50 - 0.55
Srinagar
Chandigarh
Shimla
Dehradun
NEW DELHI
Jaipur
Lucknow
Gangtok
Itanagar
Guwahati
Shillong
Kohima
Imphal
Patna
Agartala
Aizwal
Gandhinagar
Bhopal
Ranchi
KOLKATA
Diu
Silvassa
Raipur
Bhubaneshwar
MUMBAI
Hyderabad
Panaji
Bengaluru
CHENNAI
Puducherry
Kavarati
Port Blair
Thiruvananthapuram
Arabian Sea
Bay of Bengal
Indian Ocean
Lakshadweep Islands
Andaman & Nicobar Islands