Since I can't directly read the uploaded files, I'll provide you with a step-by-step guide on how to integrate the algorithms from your `onlyEncryption.py` and `onlyDecryption.py` files into your Flask application. Here’s how you can do it:

### 1. Encryption Process

1. \*\*Video Preparation and Initial Key (VID) Generation\*\*

```python

import os

from Crypto.Cipher import AES

from Crypto.PublicKey import RSA

from Crypto.Random import get\_random\_bytes

from ECC import ECC\_Generate\_Key # Assuming ECC\_Generate\_Key is defined in an ECC module

def prepare\_and\_encrypt\_video(video\_path, receiver\_public\_key, receiver\_mac):

with open(video\_path, 'rb') as video\_file:

video\_data = video\_file.read()

# Split the video into 1 MB chunks

chunk\_size = 1 \* 1024 \* 1024

chunks = [video\_data[i:i + chunk\_size] for i in range(0, len(video\_data), chunk\_size)]

# Extract the first 16 bytes from the first chunk as VID

VID = chunks[0][:16]

encrypted\_chunks = []

previous\_key = VID

# Load receiver's public key

rsa\_key = RSA.import\_key(receiver\_public\_key)

# Encrypt the first chunk using RSA

rsa\_cipher = PKCS1\_OAEP.new(rsa\_key)

encrypted\_chunks.append(rsa\_cipher.encrypt(chunks[0]))

# Encrypt subsequent chunks using AES with keys generated via ECC

for chunk in chunks[1:]:

new\_key = ECC\_Generate\_Key(previous\_key, receiver\_mac)

aes\_cipher = AES.new(new\_key, AES.MODE\_EAX)

nonce = aes\_cipher.nonce

ciphertext, tag = aes\_cipher.encrypt\_and\_digest(chunk)

encrypted\_chunks.append((nonce, ciphertext, tag))

previous\_key = new\_key

return encrypted\_chunks, VID

```

### 2. Decryption Process

```python

from Crypto.Cipher import AES, PKCS1\_OAEP

from Crypto.PublicKey import RSA

from ECC import ECC\_Generate\_Key # Assuming ECC\_Generate\_Key is defined in an ECC module

def decrypt\_video(encrypted\_chunks, VID, receiver\_private\_key, receiver\_mac):

rsa\_key = RSA.import\_key(receiver\_private\_key)

# Decrypt the first chunk using RSA

rsa\_cipher = PKCS1\_OAEP.new(rsa\_key)

first\_chunk = rsa\_cipher.decrypt(encrypted\_chunks[0])

decrypted\_video = first\_chunk

previous\_key = VID

# Decrypt subsequent chunks using AES with keys generated via ECC

for nonce, ciphertext, tag in encrypted\_chunks[1:]:

new\_key = ECC\_Generate\_Key(previous\_key, receiver\_mac)

aes\_cipher = AES.new(new\_key, AES.MODE\_EAX, nonce=nonce)

chunk = aes\_cipher.decrypt\_and\_verify(ciphertext, tag)

decrypted\_video += chunk

previous\_key = new\_key

return decrypted\_video

```

### 3. Integrate into Flask Application

Add endpoints for encryption and decryption in your Flask app:

```python

from flask import Flask, request, session, send\_file

import os

app = Flask(\_\_name\_\_)

app.secret\_key = 'your\_secret\_key'

@app.route('/upload', methods=['POST'])

def upload\_video():

if 'file' not in request.files:

return 'No file part', 400

file = request.files['file']

if file.filename == '':

return 'No selected file', 400

# Save the uploaded file

video\_path = os.path.join('uploads', file.filename)

file.save(video\_path)

# Get the receiver's public key and MAC address (assuming you have these stored somewhere)

receiver\_public\_key = get\_receiver\_public\_key(session['username'])

receiver\_mac = get\_receiver\_mac(session['username'])

# Encrypt the video

encrypted\_chunks, VID = prepare\_and\_encrypt\_video(video\_path, receiver\_public\_key, receiver\_mac)

# Save the encrypted video and VID (this is up to you how you want to store it)

save\_encrypted\_video(encrypted\_chunks, VID, video\_path)

return 'Video uploaded and encrypted successfully', 200

@app.route('/download/<video\_id>', methods=['GET'])

def download\_video(video\_id):

# Retrieve encrypted chunks and VID from storage

encrypted\_chunks, VID = get\_encrypted\_video(video\_id)

# Get the receiver's private key and MAC address (assuming you have these stored somewhere)

receiver\_private\_key = get\_receiver\_private\_key(session['username'])

receiver\_mac = get\_receiver\_mac(session['username'])

# Decrypt the video

decrypted\_video = decrypt\_video(encrypted\_chunks, VID, receiver\_private\_key, receiver\_mac)

# Save the decrypted video temporarily

decrypted\_video\_path = os.path.join('decrypted\_videos', f'{video\_id}.mp4')

with open(decrypted\_video\_path, 'wb') as f:

f.write(decrypted\_video)

return send\_file(decrypted\_video\_path, as\_attachment=True)

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

```

### 4. Utility Functions (Implement as needed)

```python

def get\_receiver\_public\_key(username):

# Fetch the public key for the user from the database

pass

def get\_receiver\_private\_key(username):

# Fetch the private key for the user from the database

pass

def get\_receiver\_mac(username):

# Fetch the MAC address for the user from the database

pass

def save\_encrypted\_video(encrypted\_chunks, VID, video\_path):

# Save the encrypted video chunks and VID to your storage (database, file system, etc.)

pass

def get\_encrypted\_video(video\_id):

# Retrieve the encrypted video chunks and VID from your storage

pass

```

This guide integrates the algorithms into your Flask app and outlines the necessary functions for encryption and decryption processes. Ensure you have the `ECC\_Generate\_Key` function defined correctly and make sure to handle your keys and MAC addresses securely.