Project update

Work Completed:

1. Backend functionalities/logic for video encryption and decryption is ready.
2. Implemented video chunking and combination processes.

[in short algorithm implementation works fine without much user interface]

Work in Progress:

3. Creating a basic UI

Next Steps:

4. Integrating user authentication
5. Complete User Interface/dashboard
6. Generate performance metrics

In our last discussion below was the question for our project

- Why are we implementing this kind of algorithm?…. what is good in our algorithm for selecting it rather than others.

I see our algorithm standing out for reasons like below:

1. **Versatility**: Unlike many encryption algorithms that are designed for specific use cases, our algorithm offers versatility by accommodating both text and video files. Its adaptive encryption approach(different for video and text) ensures that the most suitable encryption method is applied based on the characteristics of the input file, optimizing both security and efficiency.

2. **Hybrid Encryption**: By combining RSA for key exchange and AES for symmetric encryption, our algorithm leverages the strengths of both asymmetric and symmetric encryption techniques. This hybrid approach strikes a balance between security and performance, making it suitable for a wide range of encryption scenarios.

3. **Enhanced Security Measures**: our algorithm incorporates advanced security measures such as *attribute-based encryption* for video files. By encrypting video chunks based on attributes like device ID, system timestamp, and process ID, it adds an extra layer of protection against unauthorized access and tampering.

4. **Realtime Key Generation**: While the complexity of key management may seem a concern, it reflects the robustness of our algorithm's framework. By employing RSA for secure key exchange and ECC for attribute-based encryption, our algorithm ensures that encryption keys are securely generated in realtime throughout the encryption process.

5. **Scalability and Efficiency**: Despite the complexity involved in key management, our algorithm remains scalable and efficient, capable of handling encryption tasks of varying complexities. Its ability to adapt to different encryption scenarios while maintaining performance and security sets it apart from other algorithms in the market.

6. **Decryption Time**: Measuring Decryption time can provide insights into the overall efficiency of your algorithm. Low decryption time indicates that your algorithm can quickly process and decrypt data, which is crucial for applications that require real-time or high-throughput decryption.

7. **Memory Usage**: Monitoring memory usage during encryption and decryption operations can help assess the resource efficiency of your algorithm. Lower memory usage suggests that our algorithm is optimized for memory-constrained environments, which is advantageous for devices with limited memory resources, such as mobile devices or embedded systems.
-- lower memory because the keys generated are not stored rather they are used on the go

8. **CPU Utilization**: Tracking CPU utilization during encryption and decryption tasks can provide insights into the computational efficiency of your algorithm. Low CPU utilization indicates that our algorithm effectively utilizes available CPU resources, minimizing resource contention and allowing for concurrent processing of multiple encryption or decryption tasks.