Paper Title\* (use style: *paper title*)

\*Note: Sub-titles are not captured in Xplore and should not be used

Mohammed Abdul Lateef

Computer Science and Engineering

Jawaharlal Nehru Technological University Hyderabad

Hyderabad, India

malateefz2101@gmail.comDr. P. Swetha

Computer Science and Engineering

Jawaharlal Nehru Technological University Hyderabad

Hyderabad, India

drpswetha@jntuh.ac.in

*Abstract*—The proliferation of digital content consumption, particularly in the form of text and video files, underscores the paramount importance of securing these assets against unauthorized access andcopyright infringement. In response to this imperative, this research presents a comprehensive approach to file encryption, tailoring the cryptographic strategy based on the nature of the content. For text based inputs, ahybrid of RSA and AES encryption model is employed, combining key exchange strength with efficient encryption. Video based data is emphasized to utilize a blend between RSA and Elliptic CurveCryptography (ECC) approach, adapting to the dynamic nature of video data through multiple key generation. The implemented system is realized through dynamic software featuring dedicatedmodules for text and video encryption and decryption In this innovative approach, real-time key generation, based on unique identifiers, dynamically encrypts and decodes file chunks, adapting to evolving data. To assess the effectiveness of the system, rigorous evaluations of its security and performance were conducted. The empirical results underscore the proposed approach's superiority, showcasing advancements in both performance andsecurity metrics. The research stands at the forefront of addressing contemporary challenges in therealm of file security, offering a resilient and efficient solution to combat copyright infringement andpiracy risks through cryptographic implementation.

Keywords—Multi-Key Cryptography, Hybrid Cryptography, Elliptic Curve Cryptography, Real-time FileSecuring, Dynamic Encryption, Performance Metrics

# Introduction

The proliferation of digital video content in today's information age necessitates robust encryption methods to ensure the security and privacy of video data. Video encryption is critical for protecting sensitive content from unauthorized access, piracy, and tampering. Unlike text or static images, video files present unique challenges for encryption due to their large size and the real-time transmission requirements. Traditional encryption techniques often fall short in addressing these specific needs, highlighting the importance of developing specialized encryption algorithms tailored to video data.

Encrypting video content involves complexities that stem from the nature of video data. The substantial size of video files requires encryption algorithms that can handle large data volumes efficiently without introducing significant latency. Moreover, the need for real-time processing in applications such as video streaming adds another layer of complexity. Standard encryption algorithms like AES and RSA, while secure, are computationally intensive and may not be optimized for the continuous, high-throughput nature of video streams. These challenges have led to the exploration of hybrid encryption schemes that combine the strengths of both symmetric and asymmetric cryptography to meet the specific demands of video encryption.

One promising approach to video encryption is the hybrid multikey cryptography technique. This method utilizes symmetric key algorithms for encrypting the bulk of the video data due to their efficiency and speed, while asymmetric key algorithms are used for secure key exchange and initial key encryption. By dynamically generating and encrypting keys for each video chunk, this technique enhances security and ensures that compromising one chunk does not lead to the decryption of the entire video. The use of multiple keys also reduces the risk associated with key exposure, providing an additional layer of security.

The custom algorithm developed for video encryption addresses the specific challenges associated with video data while enhancing both security and performance. By employing dynamic key generation, the algorithm ensures that each video chunk is encrypted with a unique key, significantly increasing the difficulty of unauthorized decryption. The algorithm is also optimized to minimize latency, making it suitable for real-time video streaming applications. Additionally, the flexibility to save or discard dynamic keys based on specific requirements provides a balanced approach to security and storage management. This combination of robust security measures and efficient processing positions the custom algorithm as a highly effective solution for securing video content in various applications.

Hence a custom algorithm for video-based encryption introduces a sophisticated approach to protecting video data, addressing both the security and performance needs of modern applications. By leveraging advanced cryptographic techniques and optimizing for the unique characteristics of video files, this algorithm offers a practical and efficient solution to the growing demand for secure video communication.

# LITERATURE SURVEY

## Selecting a Template (Heading 2)

## First, confirm that you have the correct template for your paper size. This template has been tailored for output on the

# PROPOSED SYSTEM

The exponential growth of video data transmission over the internet has heightened the need for robust encryption techniques capable of securing large volumes of data without compromising performance. Traditional encryption methods, such as AES and RSA, have inherent limitations when applied to video data. AES, although efficient, relies on a single key for both encryption and decryption, posing a significant security risk if the key is compromised. On the other hand, RSA, while providing secure key exchange, is computationally intensive and not suitable for encrypting large data volumes due to its slow processing speed.

Moreover, the increasing sophistication of cyber-attacks necessitates the development of more advanced encryption techniques. Attack vectors such as brute force attacks, man-in-the-middle attacks, and key compromise scenarios demand encryption methods that can withstand these threats. The challenge lies in developing a framework that not only secures data but also maintains the performance required for real-time applications, such as video streaming and secure file storage. CipherShield addresses these challenges by introducing a dynamic, multi-key encryption framework that enhances security and performance.

In the era of digital communication, ensuring the security of multimedia content, especially videos, is of paramount importance. The developed system addresses this need by implementing a robust encryption and decryption algorithm named \*\*DynamicKey Video Encryption Algorithm (DVKEA)\*\*. The DVKEA is designed to secure video files through a combination of RSA for initial key encryption and AES for chunk-wise dynamic key encryption. This hybrid approach leverages the strengths of both cryptographic techniques to provide a high level of security and efficiency.

The primary requirement of the system is to securely encrypt and decrypt video files while maintaining the integrity and confidentiality of the content. The system must handle large video files by dividing them into smaller chunks, each encrypted with a dynamically generated key.

The DVKEA employs a two-step process for encryption and decryption. Initially, RSA is used to encrypt a master key, ensuring secure key exchange. Subsequently, AES is utilized to encrypt each chunk of the video file with dynamically generated keys. These keys are derived from a predefined equation and are unique for each chunk, enhancing security.

The Dynamic Hybrid Multikey Video Encryption Algorithm (DHVEA) addresses the challenges of encrypting video content by combining the efficiency of symmetric cryptography with the security of asymmetric cryptography. The method involves dynamically generating unique keys for each video chunk, encrypting these chunks with symmetric keys, and further securing the symmetric keys using asymmetric encryption. This hybrid approach ensures robust security while maintaining the performance required for real-time video applications. The proposed method optimizes the encryption process to handle large video files efficiently, making it suitable for streaming and other high-throughput applications.

Encrypting video content poses unique challenges due to the large size of video files and the need for real-time processing. Traditional encryption methods, such as AES (Advanced Encryption Standard) and RSA (Rivest-Shamir-Adleman), have limitations when applied to video data. AES, while efficient, relies on a single key, posing a security risk if the key is compromised. RSA provides secure key exchange but is computationally intensive and not suited for large data volumes. These challenges necessitate a new approach that combines the strengths of both symmetric and asymmetric encryption to secure video content effectively without compromising performance.

Pseudocode for Encryption

```plaintext

Initialize RSA with public and private keys

Initialize AES

Set SAVE\_DYNAMIC\_KEYS flag

For each video file:

Divide the video into chunks

Generate a master AES key

Encrypt the master AES key using RSA public key

Save the encrypted master key

For each chunk:

Generate a dynamic key using the predefined equation

Encrypt the chunk using the dynamic key

Save the encrypted chunk

If SAVE\_DYNAMIC\_KEYS is True:

Save the dynamic key in a separate file

Save the encrypted video file

```

Pseudocode for Decryption

```plaintext

Initialize RSA with public and private keys

Initialize AES

For each encrypted video file:

Read the encrypted master AES key

Decrypt the master AES key using RSA private key

For each encrypted chunk:

Generate the corresponding dynamic key using the predefined equation

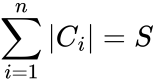
Decrypt the chunk using the dynamic key

Combine the decrypted chunks to reconstruct the video file

Save the decrypted video file

# CipherShield Algorithm: Concise Formulation with Equations

Step 1: Video Chunking

1. Let ( V ) be the original video file of size ( S ).
2. Divide ( V ) into ( n ) chunks ( C\_i ) such that: 
3. For each chunk ( C\_i ), generate a unique symmetric key ( K\_i ) using a cryptographically secure random number generator.

K\_i = text{Random}(text{KeyLength})

1. Encrypt each video chunk ( C\_i ) using its corresponding symmetric key ( K\_i ) with the AES algorithm.

E\_i = text{AES}\_text{encrypt}(C\_i, K\_i)

1. Encrypt each symmetric key ( K\_i ) using the public RSA key ( P\_text{pub} ).

K\_i^ = text{RSA}\_text{encrypt}(K\_i, P\_text{pub})

1. Store or transmit the encrypted video chunks ( E\_i ) and their corresponding encrypted keys ( K\_i^ ).

text{Stored Data} = {E\_i, K\_i^}

Decryption Process

1. Retrieve the encrypted video chunks ( E\_i ) and their corresponding encrypted keys ( K\_i^ ).
2. Decrypt each encrypted symmetric key ( K\_i^ ) using the private RSA key ( P\_text{priv} ).

K\_i = text{RSA}\_text{decrypt}(K\_i^, P\_text{priv})

1. Decrypt each encrypted video chunk ( E\_i ) using its corresponding symmetric key ( K\_i ) with the AES algorithm.

C\_i = text{AES}\_text{decrypt}(E\_i, K\_i)

1. Reassemble the decrypted video chunks ( C\_i ) to reconstruct the original video file ( V ).

V = sum\_{i=1}^{n} C\_i

# Implementation

To validate the effectiveness and performance of the CipherShield algorithm, a comprehensive web application was developed. This web application was designed to manage the entire lifecycle of video encryption and decryption, from the initial upload and chunking of video files to the final decryption and playback. The application architecture was modular, enabling independent testing and optimization of each component.

The application is structured into several distinct modules, each responsible for a specific aspect of the encryption and decryption process. This modular design ensures that each component can be independently developed, tested, and optimized, enhancing the overall robustness and efficiency of the system. The main modules include:

1. Video Upload and Chunking Module: - This module handles the initial upload of video files. Once a video file is uploaded, it is divided into smaller chunks. The chunking process is crucial for parallel processing, enabling efficient encryption and decryption. The size of each chunk is configurable, allowing the system to balance between security and performance.

2. Dynamic Key Generation Module: - Upon chunking the video, the system generates unique symmetric keys for each chunk. These keys are generated using a cryptographically secure random number generator to ensure unpredictability and enhance security. The dynamic nature of key generation ensures that each chunk is encrypted with a distinct key, mitigating the risk of key compromise.

3. Symmetric Encryption Module: - This module uses the Advanced Encryption Standard (AES) algorithm to encrypt each video chunk with its corresponding symmetric key. AES is chosen for its efficiency and robustness in handling large data volumes. The encryption process ensures that each chunk is securely encrypted, making it difficult for unauthorized entities to access the content.

4. Asymmetric Key Encryption Module: - The symmetric keys generated for each chunk are encrypted using the RSA algorithm. RSA provides secure key exchange, ensuring that the symmetric keys can be safely transmitted and stored. This module ensures that even if the encrypted video chunks are intercepted, the symmetric keys remain protected.

5.Storage and Transmission Module: - The encrypted video chunks and their corresponding encrypted symmetric keys are stored securely. This module manages the storage and retrieval of encrypted data, ensuring data integrity and confidentiality. Additionally, it handles the secure transmission of data, ensuring that the encrypted chunks and keys can be efficiently transmitted over potentially insecure channels.

6. Decryption Module: - When a user requests to view a video, the system retrieves the encrypted chunks and their corresponding encrypted symmetric keys. The decryption process involves first decrypting the symmetric keys using the private RSA key. Once the symmetric keys are retrieved, each video chunk is decrypted using its corresponding key. The decrypted chunks are then reassembled to reconstruct the original video file.

-++++

Step 1: Video Upload and Chunking- The user uploads a video file to the system. The system processes the uploaded video, dividing it into smaller chunks. For example, a video file ( V ) of size ( S ) is divided into ( n ) chunks ( C\_i ) where ( sum\_{i=1}^{n} |C\_i| = S ).

Step 2: Dynamic Key Generation- For each chunk ( C\_i ), a unique symmetric key ( K\_i ) is generated. This key is generated using a secure random number generator, ensuring that each key is unique and unpredictable.

Step 3: Symmetric Encryption- Each video chunk ( C\_i ) is encrypted using its corresponding symmetric key ( K\_i ). The encryption process uses the AES algorithm, resulting in an encrypted chunk ( E\_i = text{AES}\_text{encrypt}(C\_i, K\_i) ).

Step 4: Asymmetric Key Encryption- The symmetric key ( K\_i ) for each chunk is encrypted using the public RSA key ( P\_text{pub} ). The encrypted key is denoted as ( K\_i^ = text{RSA}\_text{encrypt}(K\_i, P\_text{pub}) ).

Step 5: Storage and Transmission- The encrypted video chunks ( E\_i ) and their corresponding encrypted keys ( K\_i^ ) are stored securely. This ensures that the encrypted data can be retrieved and transmitted securely when needed.

Step 6: Decryption Request- When a user requests to view a video, the system retrieves the encrypted chunks ( E\_i ) and their corresponding encrypted keys ( K\_i^ ).

Step 7: Asymmetric Key Decryption- Each encrypted symmetric key ( K\_i^ ) is decrypted using the private RSA key ( P\_text{priv} ). This retrieves the original symmetric key ( K\_i = text{RSA}\_text{decrypt}(K\_i^, P\_text{priv}) ).

Step 8: Symmetric Decryption- Each encrypted video chunk ( E\_i ) is decrypted using its corresponding symmetric key ( K\_i ). The decryption process uses the AES algorithm, resulting in the original chunk ( C\_i = text{AES}\_text{decrypt}(E\_i, K\_i) ).

Step 9: Video Reconstruction- The decrypted video chunks ( C\_i ) are reassembled to reconstruct the original video file ( V ). The reassembly process ensures that the video is reconstructed accurately and can be played back seamlessly.

# Results

# CONCLUSION

REFERENCES