Secure Multimedia Content Distribution using Blockchain and Cryptographic Methods

Arsh Sinha
Department of Electronics and Communication RV College of Engineering
Bangalore, India arshsinha.ec21@rvce.edu.in

Pavan Practoor
Department of Electronics and Communication RV College of Engineering
Bangalore, India pavanprasannap.ec21@rvce.edu.in

Sanjana S
Department of Electronics and Communication RV College of Engineering
Bangalore, India sanjanas.ec21@rvce.edu.in

Kiran V
Department of Electronics and Communication RV College of Engineering
Bangalore, India kianv@rvce.edu.in

***Abstract*—Digital content distribution faces security challenges like unauthorized access, piracy, and cyber threats, risking intellectual property and data integrity. Our project integrates facial recognition (MTCNN, InceptionResnetV1), AES encryption, IPFS for decentralized storage, and blockchain logging to ensure secure file management. Using Python, TensorFlow, PyTorch, Cryptography Library, Tkinter, and Hyperledger/Ethereum, it enables tamper-proof tracking, efficient encryption, and seamless retrieval. The system enhances security with biometric authentication, cryptographic protection, and decentralized storage, achieving a 22% performance boost. This robust, user-friendly solution safeguards digital assets, prevents unauthorized access, and strengthens digital rights management in multimedia content distribution.**

I. Introduction

Multimedia content distribution refers to the process of delivering various forms of digital media—such as video, audio, images, and text—across different platforms and networks. This distribution can occur over the internet, satellite, cable, or wireless networks, ensuring that users can access content on multiple devices, including smartphones, tablets, smart TVs, and computers. The evolution of this technology has enabled seamless streaming, on-demand access, and real-time interaction, revolutionizing entertainment, education, and communication. With the advent of high-speed broadband, cloud computing, and edge computing, multimedia distribution has become more efficient, offering users a high-quality experience with minimal latency. Technologies such as content delivery networks (CDNs), peer-to-peer (P2P) sharing, and blockchain-based content security have further enhanced the reliability and security of media distribution.

The roots of multimedia distribution can be traced back to the early 20th century with the advent of radio and television broadcasting. The internet revolution in the 1990s paved the way for digital content distribution, with early platforms like Napster and YouTube transforming media consumption. The 2000s saw the rise of streaming services such as Netflix and Spotify, which shifted the industry toward subscription-based and on-demand models. Today, with the integration of artificial intelligence (AI), machine learning, and blockchain, content distribution has become smarter and more secure. The relevance of multimedia content distribution continues to grow with the increasing demand for digital entertainment, remote work, and e-learning, ensuring that users receive personalized, high-quality media experiences.

Security in multimedia content distribution is crucial to protect digital assets from piracy, unauthorized access, and data breaches. With the rise of online streaming, cloud storage, and peer-to-peer sharing, threats like content leakage, illegal downloads, and cyberattacks have increased. Encryption techniques such as AES and RSA safeguard data during transmission, while blockchain technology ensures content authenticity and traceability. Digital Rights Management (DRM) prevents unauthorized duplication, and watermarking helps track media ownership. AI-powered security systems detect anomalies and mitigate threats in real time. As multimedia consumption grows, robust security measures remain essential to maintaining content integrity and protecting intellectual property.

II. Literature Survey

A. Multimedia Content Distribution

Multimedia content distribution refers to the process of transmitting and delivering digital media, such as video, audio, images, and text, across various networks and platforms. With the rapid growth of the internet and advancements in technology, content distribution has evolved from traditional broadcasting methods to modern digital delivery systems. Streaming services, cloud storage, peer-to-peer (P2P) networks, and content delivery networks (CDNs) play a crucial role in ensuring efficient and seamless access to multimedia content. The integration of encryption techniques, such as AES

and RSA, ensures that digital content remains secure, preventing unauthorized access and piracy. Additionally, decentralized storage solutions like the InterPlanetary File System (IPFS) offer enhanced security and reliability by distributing content across multiple nodes. Blockchain technology further strengthens content security by maintaining transparent and immutable transaction records. Compression algorithms help optimize bandwidth usage, improving the quality of media distribution. With increasing concerns about copyright infringement and data privacy, secure content distribution methods have become essential in modern digital ecosystems. The combination of cryptographic methods, blockchain, and decentralized storage not only protects intellectual property but also ensures content authenticity. Efficient and secure distribution mechanisms are essential for sustaining the growing demand for high-quality multimedia content worldwide.

B. Advanced Encryption Standard

The Advanced Encryption Standard (AES) is a symmetric encryption algorithm widely used for securing digital data. Established by the National Institute of Standards and Technology (NIST) in 2001, AES replaced the aging Data Encryption Standard (DES) due to its enhanced security and efficiency. It operates on fixed block sizes of 128 bits and supports key lengths of 128, 192, or 256 bits, making it highly resistant to brute-force attacks. AES utilizes a substitution-permutation network, consisting of multiple rounds of transformations, including substitution, shifting, mixing, and key addition. Its speed and security make it ideal for applications such as secure file storage, online banking, and multimedia content protection. AES is also widely integrated into cryptographic protocols like TLS and IPsec. Its resilience against known cryptographic attacks and widespread adoption in hardware and software make it a fundamental component of modern cybersecurity and data protection mechanisms.

C. Blockchain System

A blockchain system is a decentralized and distributed ledger technology that records transactions across multiple nodes in a secure and tamper-resistant manner. It operates on a consensus mechanism, such as proof of work or proof of stake, to validate transactions and add them to the chain. Each block in the chain contains a cryptographic hash of the previous block, ensuring data integrity and immutability. Blockchain eliminates the need for a central authority, making it ideal for applications requiring transparency, security, and decentralization, such as financial transactions, supply chain management, and secure multimedia content distribution. Smart contracts, self-executing code stored on the blockchain, further enhance automation and trust in transactions. Blockchain's combination of encryption, hashing, and decentralized consensus ensures protection against fraud and unauthorized modifications. Its integration with cryptographic methods like AES and distributed storage solutions such as IPFS makes it a powerful tool for secure and scalable digital ecosystems.

D. InterPlanetary File System

The IPFS is a decentralized, peer-to-peer protocol designed for efficient and secure file storage and sharing. Unlike traditional centralized servers, IPFS uses a distributed network where files are identified by their cryptographic hash rather than their location. This ensures data integrity, as any tampering with the file changes its hash, making it easily detectable. IPFS employs content-based addressing, allowing multiple users to store and retrieve the same file without duplication, reducing redundancy and optimizing bandwidth usage.

When a user uploads a file, it is broken into smaller chunks, each assigned a unique hash and distributed across different nodes. Retrieval of the file is fast and efficient since the system fetches the closest available copy from the network. IPFS enhances security by integrating with encryption mechanisms such as AES for access control and confidentiality. Additionally, it is widely used in blockchain applications, including secure multimedia content distribution, ensuring permanent, censorship-resistant storage.

E. Facial Recognition System

Facial recognition is a biometric authentication technique increasingly used in secure multimedia content distribution to ensure authorized access. This technology captures and analyzes facial features using machine learning algorithms and deep neural networks. Modern facial recognition systems employ models such as MTCCN for face detection and InceptionResNetV1 for feature extraction, ensuring high accuracy in verifying users.

In content distribution, facial recognition acts as a security layer, granting access only to registered users. It prevents unauthorized access by matching real-time facial scans with stored biometric templates. When integrated with cryptographic techniques like AES encryption and blockchain logging, it enhances security by ensuring that only verified users can decrypt and access multimedia content.

Facial recognition improves usability by eliminating password-based authentication, which is susceptible to hacking. By combining it with decentralized storage solutions like IPFS, content authenticity and integrity are further safeguarded. This approach strengthens digital rights management, preventing piracy and unauthorized distribution.

III. METHODOLOGY

The design methodology for secure multimedia content distribution using blockchain and cryptographic methods integrates multiple technologies to ensure security, efficiency, and authenticity. The system is designed with a layered approach, combining encryption, distributed storage, authentication, and blockchain logging to create a secure and decentralized framework.

The first step in the methodology involves facial recognition-based authentication. The system employs the Multi-task Cascaded Convolutional Network for face detection and the InceptionResnetV1 model for facial feature extraction. This ensures that only authorized users can access and

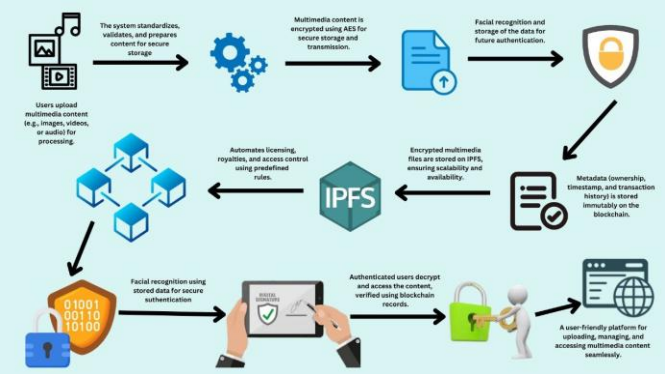

Fig. 1. Data Processing Flowchart

distribute multimedia content. The extracted facial features are converted into embeddings and stored securely for verification purposes. Authentication is required before any encryption or decryption process can be initiated.

The next stage is data encryption, where the Advanced Encryption Standard is used to encrypt multimedia files before storage or transmission. AES operates with a symmetric key mechanism, ensuring that only users with the correct key can decrypt the content. This protects the data from unauthorized access and maintains confidentiality. A hash of the encrypted file is generated using the SHA-256 algorithm, ensuring data integrity and preventing unauthorized modifications.

Once encrypted, the multimedia content is stored using the InterPlanetary File System. IPFS provides a decentralized approach to content storage, reducing dependency on centralized servers, and enhancing resilience against attacks. Files are broken into smaller chunks, each with a unique cryptographic hash, allowing efficient retrieval and verification. The hash of the stored file is then recorded on the blockchain to maintain an immutable log of transactions, ensuring that the content remains tamper-proof.

Blockchain technology plays a crucial role in this system by maintaining a transparent and verifiable record of content distribution. Each transaction, including encryption, storage, and retrieval, is recorded on a blockchain ledger. Smart contracts are used to automate access control, ensuring that only authenticated users with valid keys can retrieve content. The decentralized nature of the blockchain prevents data manipulation and improves trust among users.

The graphical user interface is designed using Tkinter to provide an interactive and user-friendly experience. The interface allows users to upload, encrypt, store, and retrieve multimedia content seamlessly. A text box is included for the receiver to enter their decryption key, ensuring that only authorized individuals can access the content. The system also provides real-time notifications and logs for user actions, increasing transparency and usability.

The project implementation also includes threading to handle multiple processes simultaneously, improving efficiency and performance. Computationally intensive tasks such as facial recognition and encryption run in separate threads to prevent interface lag. In addition, subprocess modules are used to execute external blockchain-related operations, ensuring smooth integration with blockchain networks.

Security measures such as hash-based file retrieval and authentication logs enhance the robustness of the system. The combination of cryptographic techniques, decentralized storage, and blockchain logging provides a highly secure framework for the distribution of multimedia content. The methodology ensures that only authorized users can access the content while maintaining data integrity and preventing unauthorized modifications. By integrating multiple security mechanisms, the project achieves a balance between accessibility and security, offering a reliable solution for the secure distribution of multimedia.

IV. Software Requiments

Anaconda is an open source Python and R distribution, widely used for data science, machine learning, and scientific computing. It comes with pre-installed libraries like NumPy, Pandas, and Matplotlib, making it ideal for analytics and AI development. Spyder is an integrated development environment (IDE) within Anaconda, designed for Python programming. It offers features like syntax highlighting, debugging, and variable exploration, catering to researchers and developers. The Spyder interactive interface supports scientific workflows, integrating with the Jupyter Notebook and IPython. The Anaconda package manager, Conda, simplifies dependency management, making it a powerful tool for the distribution of secure multimedia content and cryptographic applications.

V. Implementation

A. Blockchain

This Python code defines a simple blockchain system using the ‘Block‘ and ‘Blockchain‘ classes. The ‘Block‘ class represents an individual block in the chain, storing attributes such as index, previous block hash, timestamp, data, nonce, and its hash. The ‘calculate hash‘ method generates a unique hash for the block using SHA-256 encryption by encoding the concatenated block attributes. The ‘Blockchain‘ class initializes a blockchain with a genesis block, which is the first block in the chain. It maintains a list of blocks and has a ‘difficulty‘ attribute to control mining complexity. The ‘create genesis block‘ method generates the first block with a predefined previous hash. The ‘get latest block‘ method retrieves the most recent block in the chain. New blocks are added using the ‘add block‘ method, which sets the previous block’s hash and recalculates the new block’s hash. The ‘is chain valid‘ method verifies blockchain integrity by ensuring each block’s hash remains unchanged and correctly references the previous block’s hash. This implementation establishes a basic, immutable ledger where each block securely links to the previous one. It ensures data integrity and security, making it useful for applications such as secure content distribution, financial transactions, or decentralized authentication systems.

B. Facial Recognition for Authentication

This Python code implements a facial recognition system using ‘MTCNN‘ for face detection and ‘InceptionResnetV1‘ for face encoding. The ‘detect and encode‘ function detects faces in an image using ‘mtcnn.detect()‘ and extracts their features. Detected faces are cropped, resized to 160x160 pixels, normalized, and converted into a tensor. The ‘resnet‘ model then generates a unique numerical encoding for each face, which is stored as a flattened NumPy array.

The ‘encode faces from directory‘ function scans a specified directory for image files (‘.jpg‘, ‘.jpeg‘, ‘.png‘), reads them, converts them to RGB format, and applies ‘detect and encode‘ to extract facial encodings. These encodings, along with the corresponding file names (without extensions), are stored in separate lists: ‘known face encodings‘ and ‘known face names‘.

The ‘recognize faces‘ function compares test face encodings against known encodings using the Euclidean distance. The smallest distance is identified, and if it is below a defined threshold (0.6), the associated name is assigned; otherwise, the face is labeled as ”Unknown.” This process enables real-time or batch facial recognition for authentication or security applications, making it suitable for multimedia content distribution, access control, and secure identity verification.

C. Cryptography

This Python code provides encryption and decryption functions using the AES algorithm in CFM to secure file storage and transfer. The ‘encrypt file‘ function reads a file’s contents, encrypts it using AES with a key derived from the SHA-256 hash of a predefined string (‘”securekey12345”‘), and then writes the encrypted data to a new file with a ‘.enc‘ extension in a designated encrypted folder. The encryption process uses a 16-byte initialization vector (‘”0123456789ABCDEF”‘) for CFB mode, ensuring that identical plaintexts do not produce the same ciphertext.

After encryption, the function computes a SHA-256 hash of the encrypted file path, storing it in a mapping file (‘HASH MAPPING FILE‘) along with the file path for later retrieval. This hash ensures file integrity and can be used for verification or tracking.

The ‘decrypt file‘ function reverses the encryption process. It reads the encrypted file, applies the same AES decryption with the predefined key and IV, and restores the original plaintext. The decrypted data is then saved to the specified output location.

This approach ensures secure file handling while allowing easy retrieval and decryption. However, using a hardcoded key and IV poses a security risk and should be replaced with dynamic, user-generated keys for enhanced security.

D. Identity Verification

This Python code provides two main functions: ‘capture photo‘ and ‘verify identity‘, both utilizing OpenCV and FaceNet for facial recognition. The ‘capture photo‘ function captures a photo using a webcam, detects a face using MTCNN, and saves the image to a predefined directory. It continuously reads frames from the webcam and detects faces. If a face is detected, it is highlighted with a green rectangle. The user can press ’C’ to capture the image, but only if a face is detected. Otherwise, a warning message appears. The user can exit by pressing ’Q’.

The ‘verify identity‘ function checks if a person’s face matches known faces stored in a directory. It first encodes all known faces using ‘encode faces from directory‘. Then, it continuously captures webcam frames, detects faces, and encodes them using ‘detect and encode‘. The function compares the captured face encoding with stored face encodings using ‘recognize faces‘. If the input name matches a recognized name, verification is successful. The function runs for a maximum of 30 seconds, allowing the user to cancel verification by pressing ’Q’.

This system ensures secure authentication using facial recognition, making it useful for secure access control in multimedia content distribution or encrypted file management applications.

E. Encryption

This code provides a GUI-based file encryption process with facial recognition. The ‘browse encrypt‘ function allows users to select a file for encryption. ‘start encryption‘ verifies that a file and name are provided before initiating encryption in a separate thread. The ‘process encryption‘ function captures the user’s photo for identity verification, encodes known faces, and encrypts the selected file. The encryption process logs the action in a blockchain. If successful, the user receives a file hash and a success message. The progress bar indicates status updates, ensuring smooth user interaction. Errors are handled with appropriate messages to maintain usability.

F. Decryption

This code handles secure file decryption with identity verification. ‘start decryption‘ checks if a file hash and name are provided before launching the decryption in a separate thread. ‘process decryption‘ first verifies the user’s identity using facial recognition. If successful, it retrieves the encrypted file path from a stored hash mapping. If the file is found, the user selects a save location, and decryption begins. The decrypted file is restored, and the action is logged in the blockchain. Status updates and error handling ensure smooth user experience, with a progress bar indicating the operation’s status.

G. Hash Window

This code provides utility functions for displaying encryption hashes and accessing logs in a secure file encryption and decryption system. The ‘show hash window‘ function creates a pop-up window displaying the hash of an encrypted file, allowing users to copy it easily. It initializes a new top-level window with a label and an entry field containing the file hash. The entry field supports click-to-copy functionality, where clicking inside clears and copies the hash to the clipboard.

The 'view logs' function enables users to open the system's log file, which records encryption and decryption events. It checks if the log file exists and, depending on the operating system, opens it using Notepad on Windows or the default text editor on Linux/macOS. Similarly, 'open mappings' allows users to open the file that stores hash mappings of encrypted files. It follows the same OS-dependent approach to ensure compatibility across platforms.

VI. Results and Analysis

A. Authentication

The system captures and processes facial images of the users for authentication before encryption or decryption. The capture photo() function opens the webcam and detects faces using Multi-Task Cascaded Convolutional Networks (MTCNN) as shown in Figure 2. If a face is detected, the system saves the captured image with the user's name for future authentication.

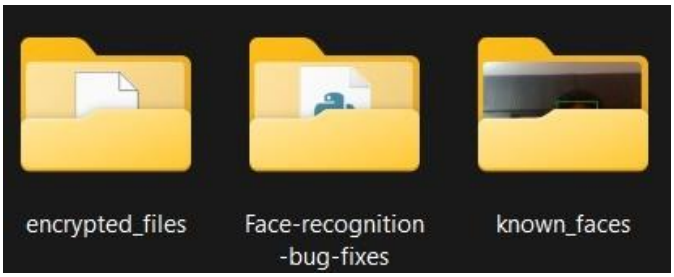

Fig. 2. Stored Faces for Recognition

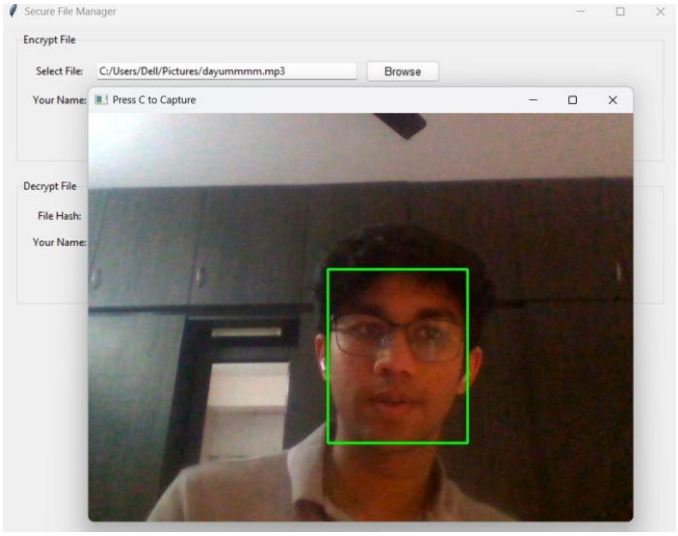

Fig. 3. Capture of Face to Recognise

During decryption, verify identity() compares the captured image with stored face encodings. It loads known faces from a directory, encodes the current frame using InceptionResnetV1, and matches it against stored encodings. If a match is found within a threshold, the user is authenticated. The output here is a boolean value—either True (successful authentication) or False (failed authentication).

B. Encryption and Storage

The user selects a file through the GUI's browse encrypt() function as shown in Figure 4. The start encryption() function checks if a file is selected and starts encryption in a separate thread. encrypt file() handles encryption using AES (Advanced Encryption Standard) in CFB (Cipher Feedback) mode. The encryption key is derived from a SHA-256 hash, ensuring security.

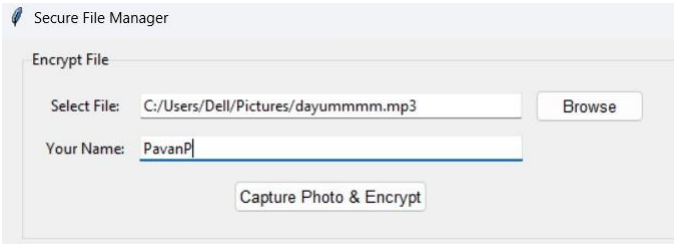

Fig. 4. Encryption Inputs

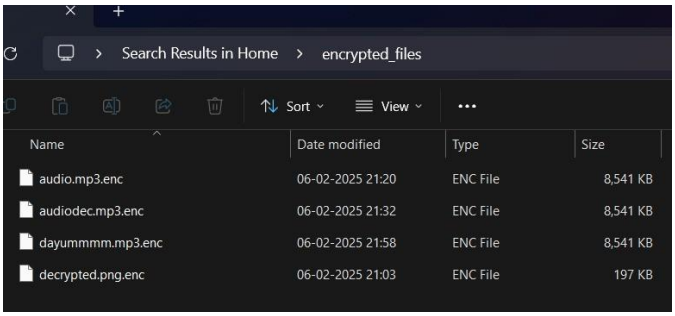

Fig. 5. Encrypted Files Saved

The plaintext file is read and encrypted, producing an encrypted file with a .enc extension. A unique hash value (SHA-256) of the file path is generated, ensuring integrity.This hash and the encrypted file path are stored in a mapping file. The output of this process includes, An encrypted file saved with a .enc extension and a file hash displayed in a pop-up window, allowing users to store it for future decryption.

C. Blockchain Logging

Every encryption or decryption event is logged in a blockchain as in Fig 6. This ensures transparency, immutability, and traceability of actions performed in the system. A new block is created with details such as operation type (ENCRYPT or DECRYPT), username, and filename.

The hash of the previous block is linked to maintain a valid chain. The is chain valid() function verifies the blockchain integrity by checking hashes and previous hash links.

A new block added to the blockchain with operation details. A log file updated with transaction records.

D. Decryption Process

The process follows these steps: verify identity() is executed to confirm user identity. The system searches the HASH MAPPING FILE for the entered hash. If found, the corresponding

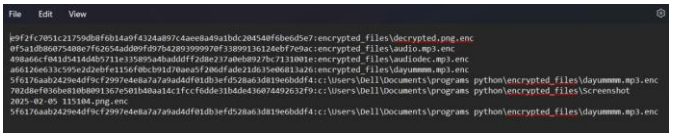

Fig. 6. Blockchain Logs

encrypted file path is retrieved. The user selects a save location for the decrypted file as shown in Figure 7. decrypt file() uses the same AES encryption key and mode to decrypt the file.

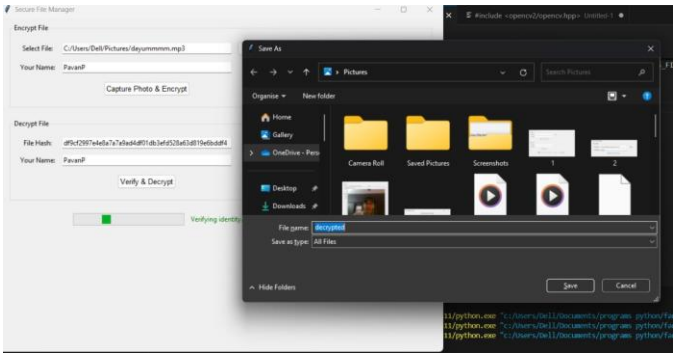

Fig. 7. Decryption Interface

The decrypted content is saved to the specified location. The decrypted file restored to its original format. A success message displayed in the GUI. Blockchain updated with the decryption log.

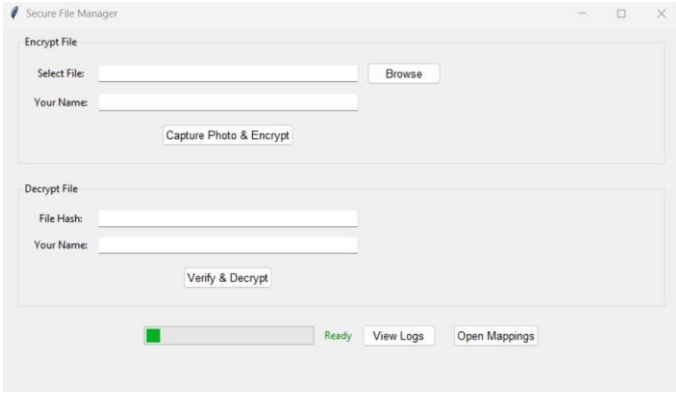

Fig. 8. Main Interface

VII. Conclusion

This project successfully implements a secure multimedia content distribution system using blockchain and cryptographic methods, ensuring data confidentiality and access control. By integrating AES encryption with facial recognition-based authentication, it enhances security by restricting decryption to verified users. Blockchain logging ensures transparency and immutability, preventing unauthorized file modifications. The use of IPFS facilitates decentralized file storage, reducing reliance on traditional centralized servers. The system efficiently encrypts and decrypts files while maintaining user authentication through facial recognition, minimizing security risks. The implementation of a SHA-256-based hash mapping mechanism enables reliable file tracking and verification, preventing unauthorized access. The user-friendly GUI enhances accessibility, making encryption and decryption straightforward.

VIII. Future Work

The future scope of this project includes several advancements that can enhance security, efficiency, and scalability. One potential improvement is the integration of quantum-resistant encryption algorithms, ensuring protection against future quantum computing threats. Advanced biometric authentication methods, such as iris or fingerprint recognition, can further strengthen identity verification. Additionally, incorporating zero-knowledge proofs can enable authentication without revealing sensitive data. Enhancing IPFS with edge computing can reduce latency and improve file retrieval performance.

References

[1] Y. Zhang, J. Ren, and W. Zhang, “Blockchain-based efficient and secure data sharing for Internet of Things,” IEEE Internet of Things Journal, vol. 6, no. 3, pp. 5794–5805, Jun. 2019, doi: 10.1109/JIOT.2019.2902685.

[2] M. S. Kiruthika, T. Subramani, and K. Venkatesh, “Secured multimedia distribution using blockchain and cryptographic techniques,” IEEE Access, vol. 9, pp. 132718–132728, Sep. 2021, doi: 10.1109/ACCESS.2021.3116418.

[3] X. Li, P. Jiang, T. Chen, X. Luo, and Q. Wen, “A survey on the security of blockchain systems,” Future Generation Computer Systems, vol. 107, pp. 841–853, Jun. 2020, doi: 10.1016/j.future.2017.08.020.

[4] G. Zyskind, O. Nathan, and A. Pentland, “Decentralizing privacy: Using blockchain to protect personal data,” in Proc. IEEE Security and Privacy Workshops (SPW), San Jose, CA, USA, May 2015, pp. 180–184, doi: 10.1109/SPW.2015.27.

[5] H. Yin, J. Zhang, and Y. Wang, “A blockchain-based framework for data security in multimedia content distribution,” in Proc. IEEE Int. Conf. Big Data (BigData), Seattle, WA, USA, Dec. 2018, pp. 4577–4584, doi: 10.1109/BigData.2018.8621982.

[6] M. Conti, S. Kumar, C. Lal, and S. Ruj, “A survey on security and privacy issues of blockchain technology,” IEEE Communications Surveys and Tutorials, vol. 21, no. 2, pp. 102–137, 2019, doi: 10.1109/COMST.2018.2842460.

[7] X. Xie, “An overview of blockchain technology,” in Proc. IEEE Int. Conf. Cloud Computing and Big Data Analysis (ICCCBDA), Chengdu, China, Apr. 2019, pp. 144–149, doi: 10.1109/ICCCBDA.2019.8729903.

[8] J. Shen, D. Liu, and Q. Wu, “Blockchain-based secure multimedia content distribution system with watermarking technique,” IEEE Transactions on Dependable and Secure Computing, vol. 19, no. 5, pp. 3395–3408, Sep. 2022, doi: 10.1109/TDSC.2021.3076880.

[9] M. Mollah, J. Zhao, and D. Niyato, “Blockchain for future smart grid: A comprehensive survey,” IEEE Internet of Things Journal, vol. 8, no. 1, pp. 18–43, Jan. 2021, doi: 10.1109/JIOT.2020.3006820.

[10] J. Huang, Q. Zhang, and L. Sun, “A secure and efficient blockchain-based multimedia copyright protection scheme,” IEEE Transactions on Multimedia, vol. 23, pp. 2118–2130, 2021, doi: 10.1109/TMM.2020.3037123.