JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

HACKSHEILD
Blockchain Based Message Encryption App

[1]P. Swati, [2]Aditi Kumari, [3]Leesa Sinha,[4]Aayush Chandravanshi

[1]Asst. Prof., [2]Student, [3]Student, [4]Student
[1]Computer Science and Engineering,
[1]BIT, Raipur, India,

Abstract : This research presents HackShield, a mobile-based security application that combines steganography and blockchain-inspired verification to securely encode and share sensitive information. The system leverages Least Significant Bit (LSB) steganography to embed encrypted payloads—such as confidential messages—within image files, ensuring that the existence of the message remains hidden from unintended recipients. Each payload is bundled with metadata including a recipient ID, sender's device ID, timestamp, and a unique SHA-256 hash to maintain data integrity. This structure is then stored and tracked, mimicking the append-only nature of blockchain without incurring its computational complexity.

The application is built using React Native and Expo, offering a modern, responsive user interface. After encoding, the system saves the stego-image locally and provides users with a secure option to share the image via file/document mode to prevent compression. A custom "Block Explorer" tab mimics blockchain interfaces by listing mined blocks with hash IDs, sender, recipient, and verification status. The decode module verifies device authorization by comparing device hashes and validates the data's integrity using the embedded hash and database-stored blocks.

The system aims to offer a privacy-preserving communication framework for secure data sharing, suitable for use cases like whistleblowing, secure ID transfer, and encrypted record-keeping. This hybrid approach bridges the usability of mobile apps with principles inspired by blockchain and cryptography, offering a novel, lightweight alternative to traditional secure communication platforms.

Keywords: Steganography, Blockchain, Secure Communication, Device ID Verification, LSB Encoding, SHA-256 Hash, React Native, Flask API, MongoDB, Tamper-Proof Messaging

1. INTRODUCTION

In today's hyperconnected world, the volume of digital communication has surged to unprecedented levels. Along with the growth of messaging platforms and cloud-based sharing services comes the urgent need for safeguarding private information from unauthorized access, surveillance, and data breaches. As people share increasing amounts of sensitive data—ranging from personal messages to corporate or legal documents—across online channels, traditional security models are being put to the test. Encryption has long served as the primary mechanism to protect the contents of digital communication. However, the mere presence of encryption can itself signal that something important or secret is being transmitted, thereby potentially drawing unwanted attention or interception attempts.

This is where steganography, the art and science of hiding messages within non-suspicious media such as images, audio, or video, becomes a powerful tool. Unlike encryption, steganography ensures that the communication remains completely invisible to unintended viewers. By embedding sensitive information within innocuous digital files, users can communicate covertly, without alerting third parties that a secret message even exists.

HackShield is a modern mobile-based application designed to capitalize on this capability. It provides a novel and hybrid approach to secure communication by blending LSB (Least Significant Bit) steganography with blockchain-inspired cryptographic verification and device-bound access control. The primary objective of HackShield is to enable users to embed confidential messages into image files on their smartphones and share them securely, ensuring that only authorized devices can decode and view the message. The application has been built with a strong emphasis on privacy, authenticity, and user-friendliness.

What sets HackShield apart from traditional steganographic tools is its layered security architecture. Each payload (i.e., the secret message) is bundled with associated metadata such as the sender's device ID, recipient ID, and timestamp. This complete payload is then hashed using SHA-256, a cryptographic algorithm known for its integrity-preserving properties, and the resulting hash is stored in a MongoDB collection. This storage acts as a lightweight blockchain-inspired ledger, where each entry functions as a verifiable block. If any part of the message or metadata is altered, the regenerated hash during decoding will not match the stored hash—instantly signaling a tampering attempt.

The frontend of HackShield is developed using React Native, enabling seamless performance across both Android and iOS platforms. The backend is powered by a Flask API, which handles image processing, hashing, steganographic embedding/extraction, and MongoDB communication. This modular architecture allows for a responsive user experience while maintaining security and efficiency. The user-friendly interface ensures that even non-technical users can utilize the platform to protect their private communications without needing to understand cryptographic concepts or steganographic algorithms.

During encoding, users are prompted to select an image, enter a secret message, and specify a recipient. The system computes the sender's hashed device ID and packages all information into a structured payload. This payload is then hidden inside the image using LSB steganography. The image can then be shared via any secure file-sharing method, with the guarantee that the data will remain hidden. During decoding, the image is uploaded by the recipient device, which first undergoes a device authorization check. If the device's hash does not match the stored recipient ID, the decoding process is aborted, preventing unauthorized access.

Moreover, HackShield includes a Block Explorer tab, which provides transparency by allowing users to view previously encoded "blocks" along with their timestamp, hash, sender, and recipient metadata. This not only strengthens the application's credibility but also mimics public blockchain behavior, offering verifiability and traceability without the overhead of decentralized consensus.

HackShield is built with scalability and extensibility in mind. While the current version focuses on image-based steganography, the underlying framework is modular and adaptable, making it capable of supporting future enhancements such as audio, video, and document-based steganography. These additions would enable broader use cases, from secure voice message delivery to tamper-proof document storage in legal and institutional environments. Additionally, the system's architecture is flexible enough to integrate with decentralized technologies such as IPFS (InterPlanetary File System) for distributed file storage and smart contracts on platforms like Ethereum for autonomous verification and access management. These integrations would not only improve data availability and censorship resistance but also elevate HackShield into the realm of fully decentralized secure communication systems.

This paper explores the architectural design, component-level implementation, data flow, and real-world performance evaluation of HackShield. It demonstrates that by combining steganography, device fingerprinting, cryptographic hashing, and blockchain-inspired data tracking, lightweight mobile applications can be built to deliver military-grade security in a practical, accessible, and user-centric manner. With increasing global concerns about digital privacy, surveillance, and unauthorized data interception, HackShield stands as a relevant and forward-looking solution for secure and invisible communication in the modern digital age. Its mobile-first, privacy-focused approach places it in a unique position to contribute meaningfully to the ongoing discourse around data ownership, transparency, and digital sovereignty, setting a strong foundation for the next generation of secure communication technologies.

MOTIVAITON

In an era where digital communication is ubiquitous, the need for secure, private, and tamper-proof information exchange is more critical than ever. Traditional encryption methods, while effective in obfuscating the content of messages, often fall short when it comes to concealing the very existence of the communication. Encrypted messages themselves can raise suspicion, especially in sensitive environments such as whistleblowing, legal transfers, or personal data exchanges. Furthermore, many conventional approaches rely on a visible layer of security, lacking subtlety and raising the risk of interception or targeted attacks.The motivation behind HackShield arises from the limitations of these traditional security mechanisms and the growing necessity for invisible and verifiable communication systems. While steganography has long existed as a means of concealing information within digital media, most tools available today are limited to offline desktop software with minimal user authentication, leaving them vulnerable to unauthorized access. Additionally, standalone steganographic tools typically do not validate message integrity, making them susceptible to tampering without detection.

On the other hand, blockchain technology provides excellent data immutability and traceability, but it often comes with significant computational and infrastructural overhead. For lightweight applications like mobile communication, the full implementation of blockchain can be excessive and impractical.HackShield aims to bridge this gap by integrating steganographic data hiding with blockchain-inspired verification, implemented in a mobile-first environment. The core idea is to provide users with a seamless, secure, and invisible communication experience where secret messages are embedded inside images and can only be decoded by authorized devices. Additionally, the inclusion of a block-like tracking mechanism using MongoDB ensures data integrity and allows for tamper detection without the need for a complex blockchain infrastructure.

By combining steganography, device-based access control, and cryptographic hashing, HackShield introduces a new way to think about secure communication—one that is discreet, verifiable, and designed for real-world usability on mobile devices. This innovative fusion of concepts not only strengthens security but also makes privacy more accessible to everyday users, which is the core driving force behind this project.

2. LITERATURE SURVEY

Over the past decade, significant research has been conducted in the domains of data security, steganography, and blockchain to enable secure and verifiable communication. However, the integration of these technologies into a mobile-first, lightweight framework remains a relatively unexplored area. This literature survey reviews key contributions in the field that have inspired the development of HackShield.

3.1. Steganography for Covert Communication

Steganography has long been used as a method to conceal data within digital media, especially images. LSB (Least Significant Bit) steganography is one of the most widely studied techniques due to its simplicity and efficiency. Research by Rohith and Varadhan (2024) explores advanced techniques for embedding messages in image pixels and highlights the potential of LSB for real-world secure communication. However, most implementations were developed for desktop environments and lacked device-level authentication or any integrity checks.

Additionally, Yamini and Priya (2024) proposed hybrid encoding schemes that combined LSB steganography with scrambling techniques for document security. Their work demonstrated the potential of steganography in legal and institutional use cases, but it did not address how the hidden messages could be verified or protected against tampering.

3.2. Blockchain-Based Steganography and Data Integrity

Blockchain technologies have been extensively studied for securing digital data through immutability and decentralized verification. In a study by Chaudhari et al. (2025), the authors introduced blockchain-based authenticated stego-channels, allowing for hidden communication to be verified through smart contract validation. While promising, the integration of blockchain with steganography was limited by performance constraints and complexity when applied to mobile environments.

Suresh and Kamalakannan (2023) explored digital image steganography using blockchain for data authenticity, introducing hash-based integrity checks stored on-chain. Their system provided high-level security but lacked a mobile-first design, reducing its practicality for real-time user communication.

3.3. Mobile Security and Identity Verification

Device-bound security mechanisms have gained popularity in recent years, especially in mobile authentication. Recent work by Du et al. (2022) explored blockchain-based access control in mobile and IoT systems. Their study introduced device ID hashing and public key infrastructure (PKI) to restrict access to specific users. However, the focus remained on enterprise or cloud environments rather than peer-to-peer communication or steganographic use cases.

In another relevant study, Samyuktha et al. (2023) presented a mobile-friendly steganography tool that enabled users to embed and decode hidden data. However, their system lacked verification, and any user with the image could decode the message—posing serious privacy and misuse concerns.

3.4. Research Gaps and Need for Integration

Most prior works on steganography either focus on data hiding or data authentication separately. Few systems attempt to combine steganography with cryptographic hashing and device-bound verification, especially in a lightweight mobile framework. Moreover, the integration of a blockchain-inspired audit trail (without full blockchain overhead) remains underdeveloped.

3. METHODOLOGY

The proposed methodology for the HackShield system adopts a hybrid security approach that fuses mobile-based steganography with blockchain-inspired verification and access control mechanisms. This section outlines each stage of the system, including the architecture, encoding and decoding workflows, and how security is enforced at every step.

4.1. Mobile-First Steganographic Encoding

HackShield allows users to embed secret messages within images using Least Significant Bit (LSB) steganography directly on their smartphones. This approach ensures the message is concealed in a way that is imperceptible to the human eye and

conventional image viewers. The user selects an image, enters the secret message and the intended recipient ID, and the system prepares a payload including:

i. The secret message

ii. Device ID of the sender

iii. Recipient ID

iv. Timestamp

v. This payload is then serialized into a JSON structure and embedded into the image using LSB techniques.

4.2. Blockchain-Inspired Block Creation

Before embedding, a SHA-256 hash of the entire payload is generated to maintain message integrity. This hash acts as a unique identifier for the block. The hash, along with the recipient ID, device ID, and timestamp, is saved in a MongoDB collection which serves as a pseudo-blockchain ledger. Each saved entry represents a block that mimics the immutability and verification features of traditional blockchains.

4.3. Secure Image Sharing

After successful encoding, the stego-image is saved locally on the user's device. The application provides an option to share the image as a file (not compressed) through messaging platforms to ensure that the embedded data remains intact during transmission.

4.4. Recipient-Bound Decoding and Verification

On the recipient’s side, the decoding process begins by uploading the stego-image to the application. The system performs the following steps:

i. Authorization Check: Compares the hashed device ID of the recipient’s phone with the intended recipient ID embedded in the message.

ii. Hash Integrity Check: Recalculates the hash from the extracted payload and compares it to the original stored hash in the database.

iii. Payload Extraction: If all checks pass, the secret message is displayed to the user. Any alteration in the message or payload results in a hash mismatch, thus flagging the data as tampered.

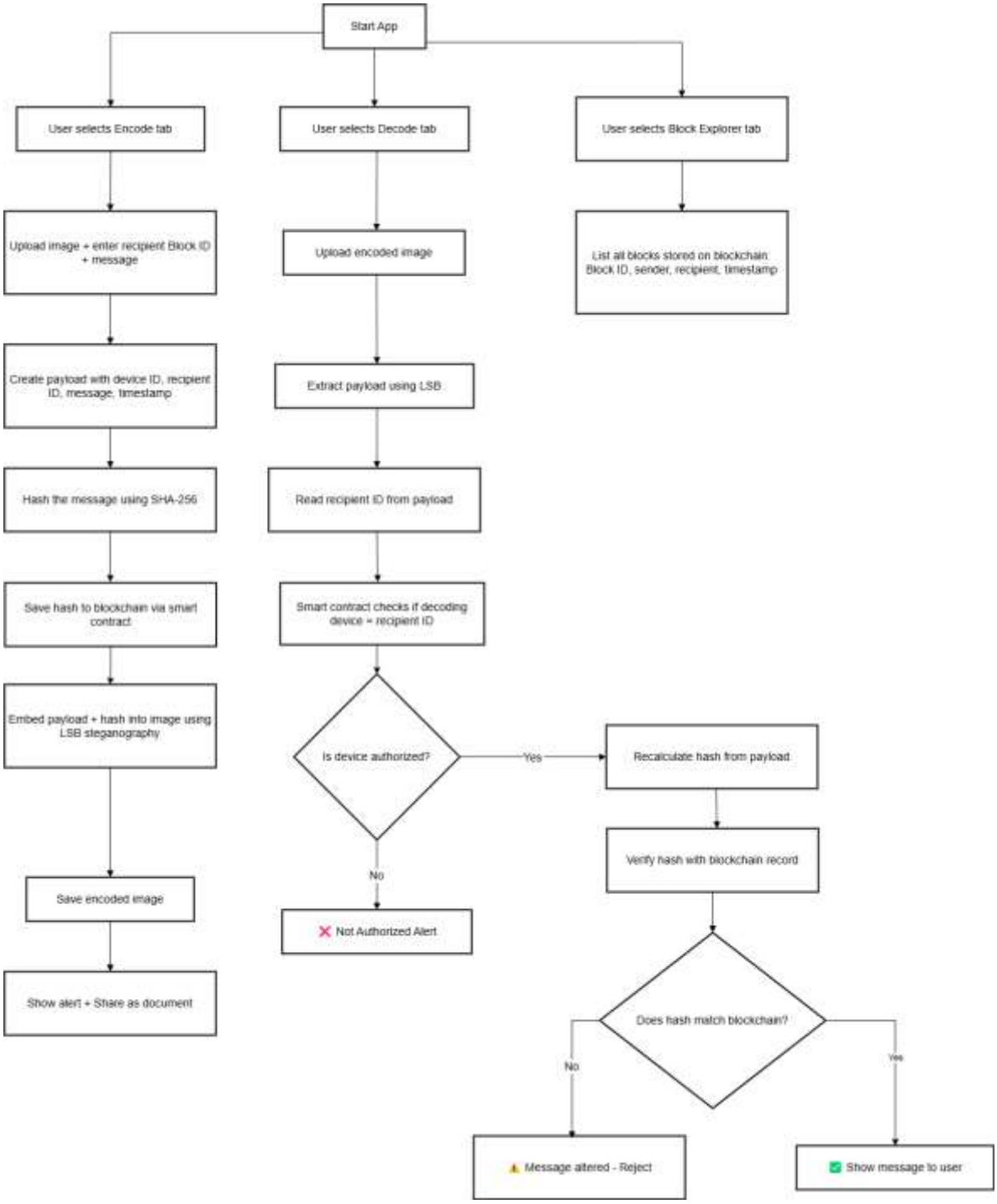

Figure. 4.1 Flow Daigram Of App

4.5. Block Explorer Integration

To improve transparency, HackShield incorporates a Block Explorer tab that lists all previously encoded blocks. Each entry includes:

i. Block Hash

ii. Sender Device ID

iii. Recipient ID

iv. Timestamp

v. Verification Status

This module mimics real-world blockchain explorers, allowing users to trace and audit every transaction (encoded message) ever made within the system.

4. IMPLEMENTATION

The implementation of HackShield is designed around a lightweight, mobile-first architecture that combines steganography, cryptographic hashing, and blockchain-inspired integrity verification. The system is developed using React Native for cross-platform mobile deployment, Flask API for backend services, and MongoDB for data storage and block tracking. The application includes three major modules: Encode, Decode, and Block Explorer. Each component contributes to creating a secure, private, and verifiable communication channel.

5.1. Mobile Frontend using React Native

The mobile application was developed using React Native with the Expo framework, allowing for cross-platform compatibility and rapid UI development. The UI comprises three tabs:

i. Encode Tab: Enables users to upload an image, enter a secret message, and assign a recipient device.

ii. Decode Tab: Allows authorized recipients to upload the image and retrieve the hidden message if the device is verified.

iii. Block Explorer Tab: Displays all previously encoded messages (blocks) with metadata including timestamp, sender, recipient, and verification hash.

The app uses device-specific attributes (such as build ID or model ID) to generate a hashed Device ID using SHA-256. This ensures that decoding is bound to the intended device only.

5.2. Secure Payload Creation and Hashing

During encoding, the user inputs a secret message and recipient, and selects an image. The system forms a payload containing:

i. Secret Message – The confidential text input by the user, which is to be securely hidden inside the image.

ii. Sender Device ID (hashed) – A unique identifier of the sender's device (e.g., model ID or OS build ID) is retrieved and hashed using SHA-256 to ensure privacy and to link the message with the originating device.

iii. Recipient Device ID – The hashed identifier of the intended recipient’s device. This ensures that only the correct device will be authorized to decode the hidden message.

iv. Timestamp (UTC format) – The current time of encoding, recorded in Coordinated Universal Time (UTC), used for tracking and verifying the message history.

These elements are combined into a single JSON payload, which is then hashed to generate a unique verification hash. The full payload is embedded invisibly into the image using Least Significant Bit (LSB) steganography, resulting in a stego-image that appears unchanged but carries the hidden data securely.)

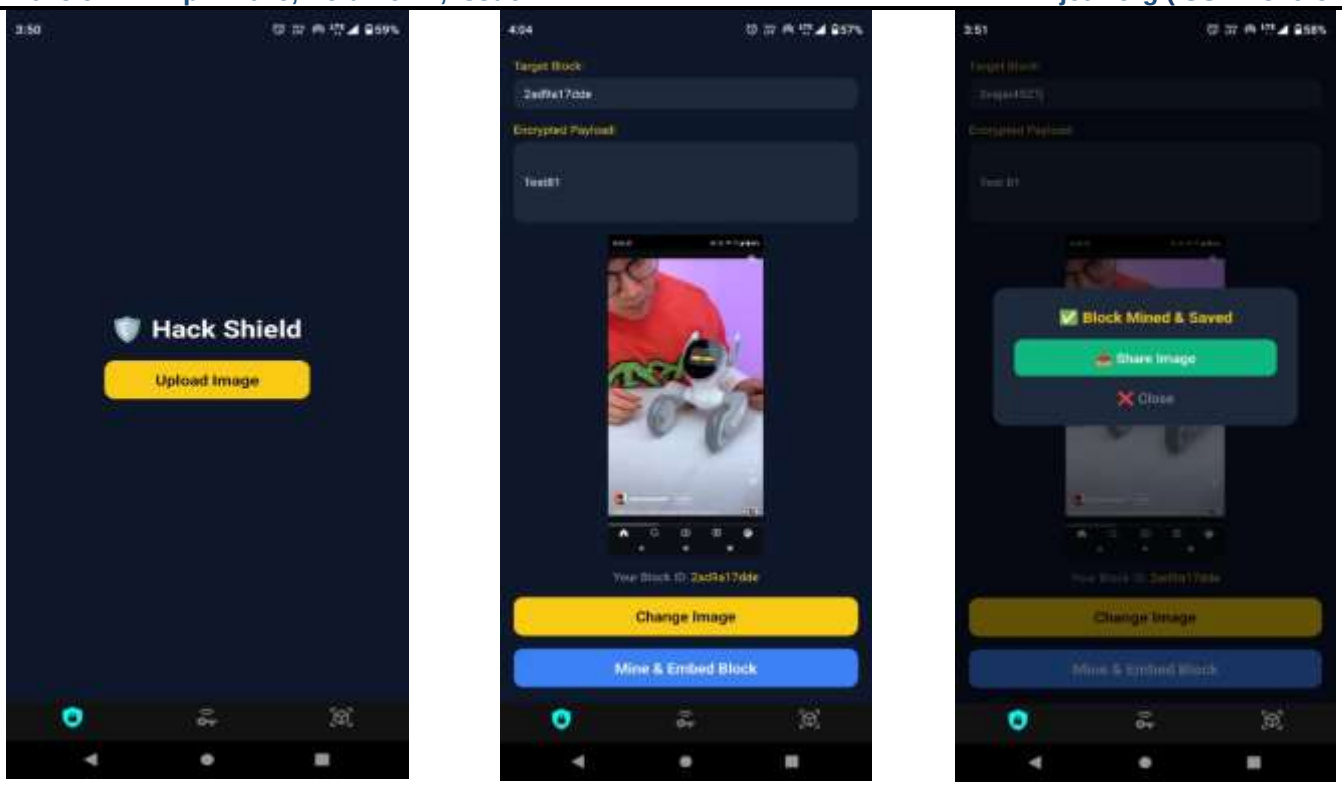

Figure. 5.1 Message Encoding On Image And Share

A SHA-256 hash of the entire payload is generated, ensuring a tamper-proof digital fingerprint. This hash acts as a unique block identifier, similar to how hashes are used in blockchain blocks.

5.3. Steganographic Embedding using LSB

The payload (along with its hash) is serialized into JSON and embedded into the selected image using Least Significant Bit (LSB) steganography. The Python library stegano.lsb is used to hide the JSON string within the image pixels without visibly altering the image. The encoded image is then converted to Base64 and sent back to the mobile application, where it is saved locally and shared through secure file-sharing methods (to prevent compression by messaging apps).

5.4. Blockchain-Like Block Storage using MongoDB

The backend uses Flask to handle API requests. Upon successful encoding, the hash, sender device ID, recipient ID, and timestamp are stored as a document in the blocks collection of MongoDB. This simulates a blockchain-like ledger, allowing for immutability and later verification without the need for full blockchain infrastructure.

5.5. Device-Restricted Decoding and Hash Verification

When decoding, the uploaded image is processed by the backend to extract the hidden payload. The system performs:

i. Device Match Check – Verifies that the current device ID matches the recipient ID in the payload.

ii. Hash Verification – Reconstructs the hash from the payload and compares it with the stored hash in MongoDB.

iii. Result Display – If verified, the original message is shown. If tampered, an alert is returned.

iv. Unauthorized devices or altered messages are immediately flagged and denied access, ensuring integrity and access control.

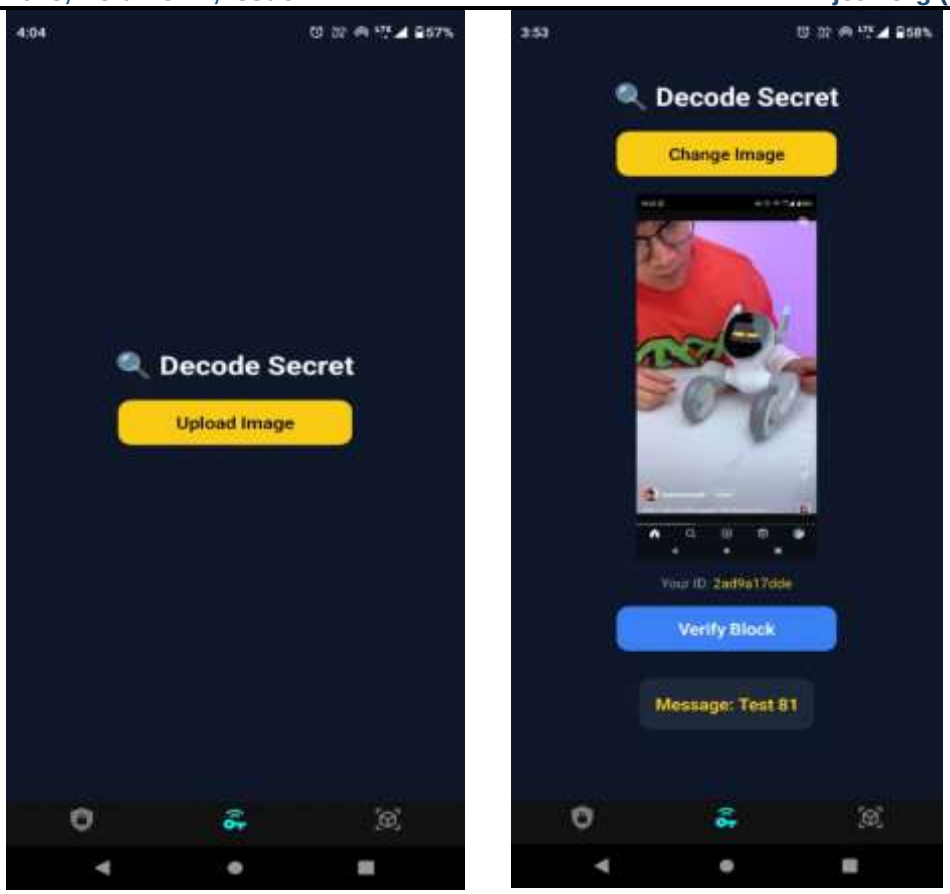

Figure.5.2 Message Decoding With Image

5.6. Block Explorer and Transparency

A "Block Explorer" tab is included in the app to display all previously stored blocks. This screen fetches data from MongoDB and displays:

i. Block Number – Represents the sequence or position of the encoded message in the database, helping maintain chronological order.

ii. Timestamp – Shows the exact date and time (in UTC format) when the message was encoded and stored, enabling historical tracking.

iii. Hash (partial) – A truncated version of the SHA-256 hash generated from the payload, used as a unique identifier for verifying message integrity.

iv. Sender and Recipient IDs – The hashed identifiers of the sender and recipient devices, ensuring that the message is traceable to its origin and destination while preserving anonymity.

v. Verification Status – Indicates whether the block is valid (✓) or tampered (✗) based on hash comparison during the decoding process.

This feature enhances transparency and trust within the system, giving users the ability to verify that their communication has been securely recorded and has not been altered.

Figure. 5.3 Block Explorer

This creates transparency and auditability, similar to public blockchain explorers, but optimized for mobile use.

5. RESULT

The HackShield application was successfully developed and deployed as a secure mobile platform for invisible, verifiable, and device-restricted message communication. The system integrates steganography, cryptographic hashing, and blockchain-inspired verification mechanisms into a lightweight mobile application without compromising usability or performance.

6.1 Successful Data Embedding with LSB Steganography

The Encode module accurately hides the input message inside the selected image using Least Significant Bit (LSB) steganography. The image quality remains visually unchanged after embedding, and the hidden message is preserved even when the image is shared using secure file transfer methods. Multiple test cases confirmed that the steganographic process did not degrade image resolution or raise suspicion.

6.2. Tamper-Proof Message Verification via Hash Comparison

Each embedded message is hashed using SHA-256 and stored in a MongoDB-based pseudo-blockchain ledger. During decoding, the system successfully re-generates the hash from the extracted payload and matches it with the stored hash. If any modification occurs—either in the image, message, or metadata—the hash mismatch is detected, and the system flags the message as tampered. This ensures integrity and tamper-proof communication.

6.3. Device-Bound Access Control

The decoding process checks whether the requesting device matches the recipient device ID hashed and embedded at the time of encoding. Unauthorized devices are denied access to the message even if they possess the image. This feature prevents misuse or leakage of secret data and ensures message privacy is preserved.

6.4. Efficient Block Tracking via Block Explorer

The Block Explorer module provides a transparent and verifiable history of all encoded messages (blocks). Users can view the hash, sender ID, recipient ID, timestamp, and validation status of each block. This component mimics a blockchain explorer and adds an extra layer of transparency and traceability to the system.

6.5. Cross-Platform and Real-Time Operation

HackShield was tested on multiple Android devices and successfully encoded, decoded, and verified messages in real-time with minimal latency. The use of React Native and Flask ensures smooth communication between frontend and backend, with no significant performance bottlenecks observed during testing.

6. CONCLUSION

This research successfully introduces HackShield, a mobile-based security application that merges the principles of steganography, cryptographic hashing, and blockchain-inspired verification to provide a novel and practical approach to secure communication. The application enables users to invisibly embed sensitive messages within images using Least Significant Bit (LSB) steganography and ensures that these messages can only be decoded by the intended recipient's device, thus maintaining both confidentiality and access control.

By integrating SHA-256 hashing, HackShield adds a tamper-proof verification layer that ensures message integrity. Any modification in the message or its metadata is promptly detected during the decoding process through hash mismatch. Furthermore, the implementation of a pseudo-blockchain ledger using MongoDB allows for transparent tracking of all encoded transactions through a dedicated Block Explorer module, offering auditability without the computational overhead of traditional blockchain networks.

The mobile-first design using React Native and Flask provides a seamless user experience across devices while ensuring robust backend logic. Extensive testing has shown that the system performs efficiently in real-time scenarios and maintains the quality and usability of the shared image files.

In essence, HackShield addresses the key limitations of existing secure messaging systems by:

i. Concealing communication through steganography,

ii. Binding messages to verified recipient devices,

iii. Ensuring data authenticity using cryptographic hashes,

iv. And offering a transparent verification history mimicking blockchain behavior.

This hybrid security model demonstrates that it is possible to build lightweight yet highly secure mobile applications for privacy-centric use cases such as whistleblowing, secure identity transfer, encrypted record-keeping, and private data exchange.

REFERENCES

[1] Waghmode, R., & Bodake, D. (2025). Provenance and authentication of digital content using blockchain and steganography. International Journal of Research Publication and Reviews.

[2] Chaudhari, P., Dol, S., & Gejage, R. (2025). Blockchain-based authenticated stego-channels and application to covert communication. IEEE Transactions on Information Forensics and Security.

[3] Rohith, V., & Varadhan, S. (2024). Exploring advanced techniques in image steganography and data hiding for secure communication. International Journal of Research Publication and Reviews.

[4] Yamini, C., & Priya, N. (2024). Hybrid encoding schemes in image steganography combined with scrambling for safeguarding legal asset documents. Indian Journal of Science and Technology.

[5] Babando, K., & Bamanga Ahmad, M. (2023). Data security using steganography. LC International Journal of STEM.

[6] Li, W., Zhang, Y., He, X., & Song, Y. (2024). Secure and efficient covert communication for blockchain-integrated SAGINs. Security and Safety.

[7] Salim, B. A., Aljabery, M. A., & Younis, H. A. (2024). AES-based steganography using blockchain: A novel approach for secure text hiding in encrypted images. Informatica.

[8] Al-Sumaidaee, G., & Žilić, Ž. (2024). Sensing data concealment in NFTs: A steganographic model for secure data exchange. Sensors.

[9] Samyuktha, V., Shradha, S., Tejaswini, P., Vaishnavi, S., & Sowmya, K. (2023). Video and image steganography. International Journal of Advanced Research in Computer and Communication Engineering.

[10] Suresh, K. S., & Kamalakannan, T. (2023). Digital image steganography in the spatial domain using blockchain technology. International Journal of Innovative Science and Applied Engineering (IJISAE).

[11] Dursun, T., & Ajlouni, N. (2023). An advanced and secure blockchain steganography algorithm. International Journal of Computational and Experimental Science and Engineering.

[12] Yang, L., & Hito, S. (2023). A novel covert communication framework for secure IoT edge computing. Chinese Journal of Electronics.

[13] Punia, A., Gulia, P., & Gillecified, N. S. (2023). A systematic review on blockchain-based access control systems in cloud environment. Journal of Cloud Computing.

[14] Du, B., He, D., Luo, M., Peng, C., & Feng, Q. (2022). The applications of blockchain in covert communication. Security and Communication Networks..