

Real Time Secure Data Encryption and Decryption Using Steganography and IoT

Meghana V¹, Charan G², Thanushree³, Manjula G Hegde⁴

¹Department of Electronics and Communication Engineering, R. R. Institute of Technology, (VTU), India

²Department of Electronics and Communication Engineering, R. R. Institute of Technology, (VTU), India

³Department of Electronics and Communication Engineering, R. R. Institute of Technology, (VTU), India

⁴Department of Electronics and Communication Engineering, R. R. Institute of Technology, (VTU), India

Corresponding author: meghanmeghana8006@gmail.com

Abstract

In the rapidly evolving digital era, the secure trans-mission of data has become a central requirement across almost every technological domain from military intelligence and medical diagnostics to industrial automation and smart urban infrastructures. This paper presents a hybrid security model integrating real-time AES-Fernet encryption with video steganography within an IoT network for secure, covert data transmission. Real-time environmental data collected via MQ135 and DHT22 sensors through an Arduino microcontroller is encrypted and embedded into live video frames using Least Significant Bit (LSB) steganography. The stegovideo is transmitted between two computing endpoints over a standard network; the receiver extracts and decrypts the hidden payload to recover original sensor values. The dual-layer mechanism encryption for confidentiality and steganography for concealment effectively thwarts unauthorized access and traffic analysis attacks. Experimental results confirm zero data loss, no visible video distortion, and real-time operational feasibility on low-cost hardware, validating the practicality of this approach for secure IoT communication in real-world environments.

Keywords - Terms Steganography, IoT, AES Encryption, LSB Em-bedding, Video Steganography, DHT22, MQ135, Arduino, Secure Data Transmission, Real-Time Communication

1. Introduction

The rapid expansion of the Internet of Things (IoT) has enabled heterogeneous devices to sense, collect, and exchange real-time data across insecure communication channels. Ensuring the confidentiality, authenticity, and integrity of transmitted data has therefore become a critical challenge in modern network security. Traditional cryptographic mechanisms transform plaintext into unreadable ciphertext, but they do not conceal the existence of the encrypted traffic itself, leaving it exposed to traffic-pattern analysis and denial-of-service attacks. Steganography complements encryption by masking the presence of hidden information within a benign-looking medium. Video steganography using digital video frames as carriers offers greater payload capacity and imperceptibility than image or audio steganography, making covert extraction nearly impossible under standard signal processing analysis or casual inspection.

IoT environments generate continuous lightweight data streams such as temperature, gas concentration, and humidity readings, which require frequent and secure transmission to remote processing platforms. Instead of transmitting raw or plaintext-encrypted values, this work embeds AES-encrypted sensor data inside live video frames using LSB substitution, creating an additional layer of cover security. The driving philosophy is “security through invisibility”: data disguised as routine multimedia traffic is far less likely to attract adversarial attention or trigger monitoring alarms. This is especially critical in sensitive domains such as military surveillance, healthcare IoT systems handling private patient data, and smart industrial systems where proprietary parameters must remain confidential.

Real-time operation introduces additional constraints. The system must be computationally lightweight enough to operate on resource-limited microcontrollers with minimal trans-mission latency.

The proposed framework addresses these challenges through frame-based payload distribution, modular encryption, and optimized LSB embedding mechanisms. The dual defense model where encryption prevents interpretation and steganography prevents detection establishes a robust security architecture for IoT data communication. Furthermore, video steganographic data blends naturally into normal internet multimedia traffic, making it indistinguishable from ordinary video streaming to any network observer.

This work contributes to interdisciplinary research bridging IoT engineering, cryptography, multimedia processing, and real-time systems. Beyond the core contribution, the system is built using inexpensive, accessible hardware Arduino UNO, MQ135, and DHT22 making it reproducible and deploy-able in both academic and industry settings. Applications include smart cities, remote industrial monitoring, environmental surveillance, healthcare data transmission, and distributed control networks. The paper is organized as follows: Section II reviews related literature; Section III presents the proposed system architecture and methodology; Section IV describes hardware and software requirements; Section V covers implementation; Section VI presents results and discussion; Section VII outlines applications and future scope, followed by the conclusion.

1.1 Objectives

The primary objectives of this work are:

- To design a system capable of securely transmitting IoT sensor data using steganographic embedding inside live video frames.
- To implement video-based covert data hiding that prevents detection by unauthorized users or monitoring systems.
- To integrate MQ135 and DHT22 sensors with Arduino UNO for real-time environmental data acquisition.
- To enable encrypted communication between two remote laptops through embedded video transmission over standard networks.

- To provide efficient extraction and decryption at the receiving end with minimal data loss and latency.
- To demonstrate a practical, low-cost method for real-time secure IoT communication in real-world environments.

1.2 Problem Statement

Secure transmission of IoT sensor data over conventional communication channels is vulnerable to interception, manipulation, and unauthorized access. While encryption protects data content, it does not conceal the existence of the transmission, making encrypted streams still detectable and susceptible to traffic analysis attacks. IoT networks using cloud or wireless links expose environmental data such as air quality and climatic readings to potential exploitation. Existing security frameworks lack mechanisms for covert real-time data transfer, particularly on resource-constrained devices such as Arduino-based systems. Combining cryptography with steganographic data hiding remains technically challenging in live applications. A solution is therefore required that embeds encrypted IoT data invisibly within live video streams to ensure secret, undetectable communication between sender and receiver systems.

2. Related Work

Several prior works have explored the combination of cryptography and steganography for secure data communication.

Khari et al. [1] concealed encrypted IoT sensor data within digital media using a combined cryptography-steganography scheme, achieving high secrecy with low visual distortion and optimized computational effort. However, restricted data capacity, dual-layer processing delay, and limited scalability for large IoT networks were identified as limitations.

Manohar N et al. [2] proposed a system where data is encrypted and embedded into digital images, achieving good imperceptibility and reliable extraction, though limited payload size and reduced real-time applicability remained concerns.

Balaji R et al. [3] distributed secret data across video frames via pixel-modification steganography with minimal perceptual impact, but faced vulnerability to

compression and degraded real-time performance. Varghese and Sasikala [4] conducted a comprehensive review of hybrid cryptography-steganography models and confirmed that combining strong encryption with steganography significantly enhances confidentiality and resilience, though system complexity and steganalysis vulnerability remained trade-offs.

Papadopoulos et al. [5] applied sequential multi-layer LSB embedding for real-time data hiding with low latency, but noted susceptibility to statistical steganalysis and compression sensitivity.

Vivek et al. [6] combined chaos-based encryption with HEVC video compression for improved secrecy and storage optimization, though chaotic key synchronization complexity limited real-time streaming.

Liu et al. [7] integrated blockchain-based privacy mechanisms with HEVC steganography for decentralized data validation, introducing blockchain overhead and latency trade-offs.

Abed et al. [8] demonstrated FPGA-accelerated LSB steganography achieving hardware-level throughput improvements with reduced CPU load, but with limited algorithmic flexibility.

Atee et al. [9] used dynamic encryption with color-image LSB steganography for enhanced secrecy through color-channel variation, though higher computation time for dynamic keying was a drawback.

3. Materials and Methods

3.1 System Overview

The proposed system consists of five functional stages: (1) Sensor/Arduino data acquisition, (2) Sender-side encryption and embedding, (3) PNG frame encoding, (4) Network transmission, and (5) Receiver-side extraction and decryption. The overall pipeline is illustrated in Fig. 1.

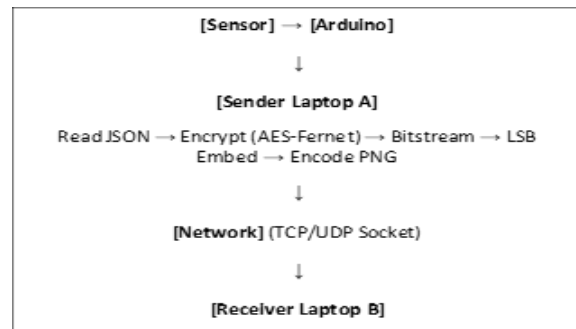


Fig. 1. Main Block Diagram of Proposed Model

The Arduino samples the DHT22 and MQ135 sensors at fixed intervals, formats readings into a timestamped JSON packet, and transmits to Laptop A via USB serial. The sender encrypts the payload using AES-Fernet, serializes it to a bitstream, and embeds the bits into the LSBs of PNG video frame pixels using a pseudo-random key-driven sequence. PNG encoding is mandatory since it is lossless JPEG compression would destroy embedded bits. The encoded stream foremost are streamed over TCP to Laptop B. At the receiver, each frame is pixel-decoded, LSBs are extracted in the correct key-driven order, the ciphertext is reconstructed, HMAC is verified, and decryption recovers the original sensor values for real-time display and logging.

3.2 Sender-Side Software Flow

The sender-side pipeline follows a sequential seven-stage process as depicted in Fig. 2.

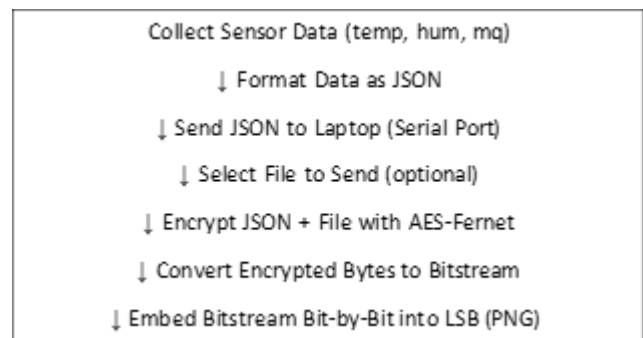


Fig. 2. Sender Part of the Proposed Model

Collect Sensor Data: DHT22 and MQ135 values are sampled at fixed intervals with range validation and smoothing applied. Each sample is tagged with a timestamp and device ID for downstream correlation and audit. Sensor errors or disconnections are logged for graceful handling at the receiver. **Format as JSON:** Raw sensor values are packaged into a structured JSON object such as handles dropped messages. A listener program on Laptop A reconstructs and

validates the JSON before proceeding. **File Selection:** An optional auxiliary file (log, image, or firmware binary) is appended to the payload. The file is chunked if large, with metadata (filename, MIME type, byte length, chunk index) embedded alongside for correct receiver side reassembly. **AES-Fernet Encryption:** The combined JSON and file payload is encrypted using AES-128 CBC mode with a secure random initialization vector (IV) and SHA-256 HMAC for integrity and authentication. Fernet ensures the receiver can both decrypt and verify the ciphertext was not tampered with or replayed. Key management (secure generation, storage, and distribution) is a critical operational dependency. **Bitstream Conversion:** Ciphertext bytes are serialized to a continuous bit sequence (MSB-first per byte) with framing headers containing payload length and CRC fields. This enables the receiver to locate the hidden message boundaries and verify completeness before decryption. **LSB Embedding:** Each payload bit replaces the LSB of a selected pixel channel in the PNG frame. The pixel selection follows a pseudo-random sequence derived from the shared secret key, ensuring the embedding pattern is unpredictable to an observer. Altering the LSB of a color channel produces imperceptible visual changes, maintaining the Steg image's natural appearance.

3.3 Receiver-Side Software Flow

The receiver pipeline follows six stages as depicted in Fig. 3.

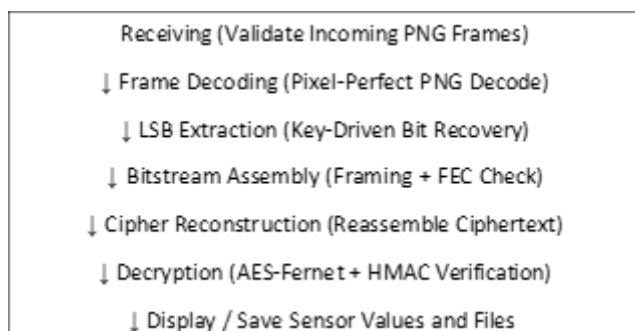


Fig. 3. Receiver Part of the Proposed Model

Model Receiving: Incoming PNG frames are validated for file integrity, expected format, and codec compliance before processing. Metadata such as source, timestamps, and file size are logged for debugging and audit purposes. **Frame Decoding:** Each PNG frame is decoded pixel perfectly using OpenCV, preserving all LSB values exactly as embedded by the sender. Dropped or corrupted

frames trigger retransmission requests or FEC-based recovery. **LSB Extraction:** The receiver reproduces the identical pseudo-random bit position sequence using the shared key and extracts LSBs from the correct pixel channels in order. Any deviation in sequence would scramble the payload.

`{"temp":27.1,"hum":45,"mq":120,"ts":"2025-1126T05:00:00Z"}`, making the payload self-describing, extensible, and easy to parse without custom parsers. **Serial Transmission:** The JSON payload is sent to Laptop A via USB serial with agreed baud rate, parity, and stop bit parameters. **Handshaking** confirms receipt; **retransmission** Bitstream Assembly: Extracted bits are organized into a coherent binary sequence. Framing markers identify message boundaries; length fields confirm completeness; FEC or CRC fields detect and correct bit errors. **Cipher Reconstruction:** Bits are reassembled into bytes to reconstruct the AES-Fernet ciphertext. HMAC and version fields are verified; chunk sequence numbers reorder any inter-leaved file segments.

Decryption: The shared symmetric key decrypts the cipher-text. On successful HMAC verification, the original JSON and file bytes are recovered. The JSON is parsed, validated, and forwarded to the real-time display dashboard. Reconstructed files are written atomically to disk to prevent corruption.

3.4 DHT22 Temperature and Humidity Sensor

The DHT22 is a high-accuracy digital sensor designed to measure both temperature and humidity in real time. It contains an NTC thermistor for temperature sensing and a capacitive humidity-sensing element, both internally calibrated to minimize measurement error. The sensor outputs digital data through a single-wire protocol requiring only one data pin on the Arduino, covering a range of -40°C to $+80^{\circ}\text{C}$ for temperature and 0–100% RH for humidity. It sends data in a structured 40-bit format, ensuring clean and stable readings. In this project, the Arduino periodically requests data from the DHT22 at fixed intervals; each reading is immediately packaged into a JSON structure and forwarded to the sender system for secure embedding.

3.5 MQ135 Air Quality Gas Sensor

The MQ135 detects air pollutants including CO_2 , NH_3 , benzene, smoke, and other harmful compounds

by measuring resistance variation in a metal oxide semiconductor (MOS) layer. This resistance change is converted into an analog output voltage read by the Arduino through the A0 ADC pin. The sensor requires a brief warm-up period to stabilize before delivering reliable readings. In this project, MQ135 readings are collected simultaneously with DHT22 data for multi-parameter environmental sensing. The pollution level values are appended into the JSON packet alongside temperature and humidity, and the combined dataset is encrypted and embedded into live video frames for secure transmission.

3.6 Arduino UNO Microcontroller

The Arduino UNO serves as the central microcontroller interfacing both sensors. It reads DHT22 via a digital I/O pin and MQ135 via the ADC on pin A0, converts analog values to meaningful digital data, and structures them into organized JSON packets containing temperature, humidity, MQ values, timestamp, and device ID. Communication with Laptop A occurs via USB serial at a fixed baud rate.

3.7 Key Algorithms

Fernet (AES-128 CBC + HMAC-SHA256): Provides authenticated symmetric encryption of the sensor and file payload with a randomly generated IV per message, ensuring confidentiality, integrity, and replay resistance.

LSB Steganography: The ciphertext bitstream is embedded into pixel LSBs of PNG frames. The pixel array is flattened, each LSB replaced with a payload bit in a key-driven pseudo-random order, and the frame is reconstructed. This achieves zero perceptible visual distortion.

PNG Lossless Encoding: PNG compression preserves embedded bits exactly. JPEG is explicitly excluded as its lossy compression destroys LSB-level data.

TCP Socket Transmission: Each frame is prefixed with a 4-byte frame-size header enabling the receiver to reconstruct full images reliably from the byte stream.

File Chunking and Reassembly: Large auxiliary files are split into indexed chunks that fit within individual frame payloads; the receiver reassembles them in sequence using chunk index metadata.

3.8 Implementation

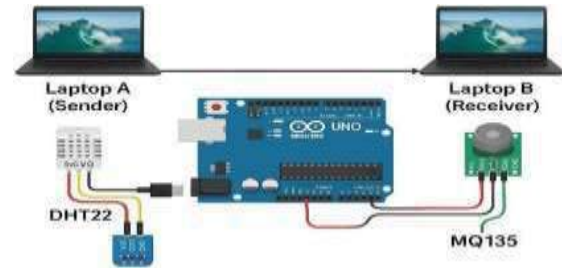


Fig.4 Circuit diagram of proposed model

The system integrates sensor-based data acquisition, micro-controller processing, video steganography, and cryptographic reconstruction in a two-laptop pipeline. Laptop A (sender) receives JSON sensor data from the Arduino via USB serial, encrypts it using AES-Fernet, embeds the ciphertext into PNG video frames via LSB substitution, and streams the stegoframes over a TCP socket to Laptop B (receiver) across a Wi-Fi or LAN connection. The resulting stegovideo is visually indistinguishable from normal video—any network observer monitoring traffic sees only routine multimedia data.

Laptop B (receiver) decodes each incoming PNG frame pixel-perfectly using OpenCV, extracts LSBs in the shared-key-derived order, reconstructs the ciphertext bitstream, verifies the HMAC, and decrypts the payload to recover original sensor values. The decoded JSON is parsed and displayed on a real-time dashboard with live sensor cards, historical graphs, and threshold alerts.

The DHT22 is interfaced with the Arduino using three connections: VCC (power), GND (ground), and DATA (signal). The MQ135 is connected via VCC, GND, and analog output (AO). Both sensors are powered directly from the Arduino board, ensuring operational consistency and stable reading acquisition throughout continuous operation. The Arduino continuously polls sensor values and streams them at 2-second intervals to Laptop A, which processes, encrypts, embeds, and transmits frames in a real-time loop.

The dual-layer security achieved in this implementation ensures that raw sensor data is protected cryptographically at the data level and simultaneously concealed steganographically at the transmission-medium level. Even if an adversary intercepts the video traffic, the presence of hidden data is undetectable without knowledge of both the embedding key and the decryption key.

4. Results and Discussion

4.1 Experimental Results

The system was tested over multiple end-to-end transmission cycles with both sensors updating at 2-second intervals. All results are summarized below.

Software Interface Outputs



Fig. 5. Software Interface at sender

Sender Interface: The sender GUI displayed a live camera preview alongside real-time sensor reading cards (28.00°C temperature, 72.90% humidity, Air Quality Index of 18). Below the sensor cards, the interface showed the full Base64 AES-Fernet ciphertext output, the binary bitstream (1120 bits), and the LSB pixel map confirming that 606 pixel positions were modified per frame. Pipeline step logs confirmed: Serial Read OK, Encrypt OK (140 bytes), Bits created (1120), Embedding OK (1152 used bits), and LSB Map showing 606 changed pixel indices.



Fig. 6. Software at receiver interface

Receiver Interface: The receiver GUI displayed the last received PNG frame, the extracted ciphertext (Base64), the reconstructed bitstream (1312 bits), and the fully decrypted JSON payload. The received log table populated in real time with timestamped rows confirming temperature, humidity, and MQ values received across successive frames. Log entries confirmed “Decryption OK and data appended” for each received frame, validating complete and accurate end-to-end pipeline operation.

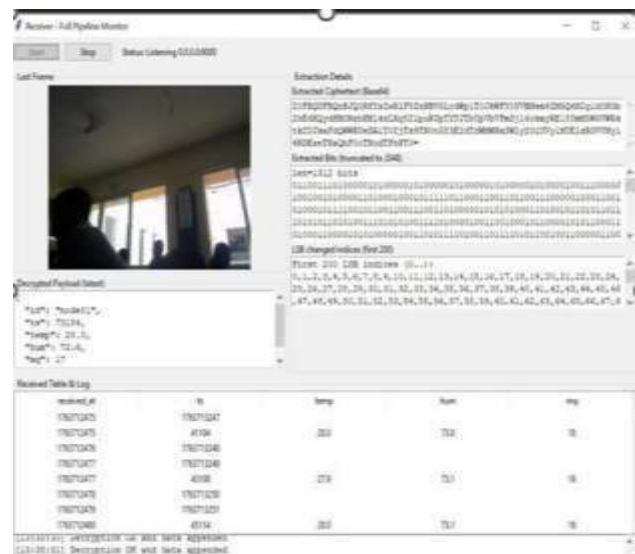


Fig.7. Receiver interface

This receiver-side output shows the successful extraction of the encrypted bitstream from the incoming video frames and its conversion back into Base64 ciphertext. The system then decrypts the payload to recover the original temperature, humidity, and MQ135 gas readings accurately. Finally, the recovered sensor data is appended to the received log table in real time, confirming correct end-to-end steganographic transmission.

4.2 Discussion

Security Analysis: The dual-layer model ensures that even if an adversary successfully performs steganographic extraction, they still face a cryptographic barrier in the form of AES-128 encryption with HMAC authentication. The stegovideo blends naturally into standard internet multimedia traffic, making it undetectable by traffic-pattern analysis tools. The HMAC prevents undetected tampering or replay attacks, adding a third layer of data integrity assurance.

LSB Embedding Efficiency: LSB substitution is computationally lightweight, enabling real-time frame processing without causing perceptible visual artifacts. The change of only 606 LSB pixels per frame—out of the thousands available—ensures minimal statistical deviation from the original frame distribution, reducing susceptibility to first-order steganalysis.

Real-Time Viability: The system operated smoothly on standard consumer-grade laptops with minimal observable latency at 2-second sensor update intervals. This confirms feasibility for deployment in resource-constrained IoT environments without requiring specialized hardware.

Limitations Observed: PNG frame sizes increase with embedded payload, which can affect available network bandwidth in high-frequency transmission scenarios. If frames are transcoded or recompressed during transmission, embedded LSB bits may be corrupted. Key management remains a critical operational dependency; any compromise of the shared secret invalidates the entire security model. Additionally, advanced steganalysis tools employing chi-square or RS analysis could potentially detect the presence of embedded data if the adversary suspects steganographic communication.

Applications

The proposed system is applicable across a broad range of sensitive IoT domains:

- **Industrial IoT:** Covert monitoring of factory, warehouse, and chemical plant sensor networks where proprietary process data must remain confidential.
- **Military Communication:** Hidden data transfer in defense environments where encrypted-only traffic may itself be a surveillance trigger.
- **Healthcare IoT:** Secure, undetectable transmission of private patient health data from wearable or ambient sensors to cloud platforms.

Advantages

The system provides dual-layer

security making data extremely difficult to intercept or detect. PNG loss-less encoding ensures hidden data remains intact throughout transfer. The solution operates over standard Wi-Fi or LAN networks without specialized communication hardware, and is suitable for low-power embedded IoT devices due to its minimal computational overhead. Real-time operation is maintained with negligible latency on consumer-grade hardware.

Disadvantages:

Embedding data inside frames increases PNG file size, affecting bandwidth consumption at high frame rates. If frames are compressed or transcoded enroute, hidden data may be corrupted. Synchronized key management between sender and receiver is an operational prerequisite. Performance may degrade with very large payload sizes.

Future Scope:

Future enhancements will focus on: (1) integration of AI-based adaptive steganography to dynamically maximize embedding capacity while minimizing statistical detectability;

1. blockchain-backed authentication for tamper-proof secure logging and key distribution.
2. GPU-accelerated real-time steganography to support higher frame rates and larger payloads.
3. mobile deployment on Android/iOS platforms for portable sensor monitoring.
4. multi-sensor fusion supporting expanded IoT sensor arrays.

5. Conclusion

This paper presented a real-time secure IoT data transmission system combining AES-Fernet encryption with LSB video steganography. Environmental sensor readings from DHT22 and MQ135 sensors, acquired via Arduino UNO, were successfully encrypted, embedded within live PNG video frames, transmitted over a standard network, and recovered at the receiver with zero data loss and no visible video distortion. The dual-layer security architecture where encryption prevents data interpretation and steganography prevents data

detection provides a robust, cost-effective, and covert communication framework for IoT applications. Experimental validation confirmed real-time operational feasibility on low-cost consumer hardware. With future enhancements including AI-driven adaptive embedding, blockchain authentication, and GPU acceleration, this framework holds strong potential for widespread deployment across sensitive real-world IoT environments including healthcare, defense, smart infrastructure, and industrial monitoring.

References (Size 10)

- [1] Khari et al., "Securing data in Internet of Things (IoT) using cryptography and steganography techniques," *IEEE Trans. Syst., Man, Cybern.: Syst.*, vol. 50, no. 1, pp. 73–80, 2019.
- [2] Manohar N et al., "Data encryption & decryption using steganography," in *Proc. 4th Int. Conf. Intell. Comput. Control Syst. (ICICCS)*, IEEE, 2020.
- [3] Balaji R et al., "Secure data transmission using video steganography," in *Proc. IEEE Int. Conf. Electro/Inf. Technol.*, IEEE, 2011.
- [4] F. Varghese and P. Sasikala, "A detailed review based on secure data transmission using cryptography and steganography," *Wireless Pers. Commun.*, vol. 129, no. 4, pp. 2291–2318, 2023.
- [5] Papadopoulos et al., "Sequential multiple LSB methods and real-time data hiding: variations for visual cryptography ciphers," *J. Real-Time Image Process.*, vol. 14, no. 1, pp. 75–86, 2018.
- [6] Vivek et al., "Video steganography using chaos encryption algorithm with HEVC for data hiding," *Int. J. Intell. Eng. Syst.*, vol. 14, no. 5, pp. 15–24, 2021.
- [7] Liu et al., "Blockchain privacy data protection method based on HEVC video steganography," in *Proc. 3rd Int. Conf. Smart BlockChain (Smart-Block)*, IEEE, 2020.
- [8] Abed et al., "An automated security approach of video steganography-based LSB using FPGA implementation," *J. Circuits, Syst. Comput.*, vol. 28, no. 05, p. 1950083, 2019.
- [9] Atee H. A. et al., "Cryptography and image steganography using dynamic encryption on LSB and color image based data hiding," *Middle-East J. Sci. Res.*, vol. 23, no. 7, pp. 1450–1460, 2015.
- [10] Magdy et al., "Advanced video steganography leveraging DNA encoding and hybrid algorithms for secure data protection," in *Proc. Int. Conf. Mach. Intell. Smart Innov. (ICMISI)*, IEEE, 2025.
- [11] Younus et al., "Video steganography using knight tour algorithm and LSB method for encrypted data," *J. Intell. Syst.*, vol. 29, no. 1, pp. 1216–1225, 2019.
- [12] Sailaja et al., "An efficient and secure data hiding technique: video steganography," *Lib. Progress-Lib. Sci., Inf. Technol. Comput.*, vol. 44, no. 3, 2024.
- [13] Dalal et al., "A secure video steganography scheme using DWT based on object tracking," *Inf. Secur. J.: Glob. Perspect.*, vol. 31, no. 2, pp. 196–213, 2022.
- [14] Patel et al., "A hybrid DST-SBPNRM approach for compressed video steganography," *Multimedia Syst.*, vol. 27, no. 3, pp. 417–428, 2021.
- [15] Thakkar et al., "Video steganography using encrypted payload for satellite communication," in *Proc. IEEE Aerosp. Conf.*, IEEE, 2017.
- [16] Kumar et al., "Development of LSB based steganography method for video and image hiding," in *Proc. 11th Int. Conf. Reliab., Infocom Technol. Optim. (ICRITO)*, IEEE, 2024.
- [17] Kale et al., "A video steganography approach with randomization algorithm using image and audio
- [18] steganography," in *Proc. Int. Conf. Emerg. Smart Comput. Inform. (ESCI)*, IEEE, 2024.
- [19] Nunna et al., "Secure data transfer through internet using cryptography and image steganography," in *Proc. SoutheastCon*, vol. 2, IEEE, 2020.