# Secure Data Transmission in IoT-Based Healthcare Systems

Prakhar Shukla[1], Areeb ur Rub[2], Hardik Prakash[3], Sambit Satpathy[4]
*[1,2,3] Department of Computer Science and Engineering Galgotias College of Engineering and Technology Greater Noida, India*
*[4]Associate Professor, Department of Computer Science and Engineering Galgotias College of Engineering and Technology Greater Noida, India*

***Abstract*—This paper presents a comprehensive approach to securing data transmission in IoT-based healthcare systems. The research addresses the critical challenges of maintaining data privacy and security in healthcare IoT devices while ensuring efficient and reliable communication. We propose a novel security framework that combines lightweight encryption with robust authentication mechanisms suitable for resource-constrained IoT devices in healthcare settings.**

***Index Terms*—Medical Edge Computing (MEC), Ascon lightweight cryptography, healthcare IoT security, real-time data processing, multicast device discovery, resource-constrained devices**

## I. INTRODUCTION

The integration of Internet of Things (IoT) devices into healthcare systems has initiated a paradigm shift in medical data collection and processing. While these connected medical devices offer real-time patient monitoring capabilities, they also present significant challenges in security, efficiency, and scalability. Traditional cloud-based systems, despite their storage benefits, often fall short in time-sensitive medical applications due to latency issues, bandwidth constraints, and security vulnerabilities.

Recent studies have highlighted these challenges, with re searchers emphasizing the need for trustworthy authentication and data preservation in digital healthcare systems [1]. The necessity for enhanced security protocols has been particularly emphasized [2], while others have suggested combining emerging technologies like fog computing and blockchain for improved security and reliability [3]. A comprehensive survey by Newaz et al. [4] highlights the critical security and privacy challenges in modern healthcare systems, while Wang et al. [5] demonstrate the potential of Medical Edge Computing in healthcare applications.

To address these challenges, we propose a Medical Edge Computing (MEC) system that processes and secures medical sensor data at the network edge. Our solution incorporates:

- NIST-standardized Ascon lightweight cryptography for robust data encryption [6]
- Automatic discovery mechanisms for medical sensors and MEC layer
- Real-time data processing capabilities
- A scalable microservices architecture

The key objectives of this research include:

- Securing medical data transmission through Ascon implementation
- Reducing latency through edge-based processing
- Enabling seamless device integration through auto discovery
- Achieving scalability via microservices architecture
- Ensuring robust error handling and system reliability.

This paper presents our approach to developing a secure, efficient, and scalable framework for healthcare IoT systems, addressing the critical needs of modern medical data processing while maintaining high standards of data privacy and security.

## II. RELATED WORK

Recent advancements in healthcare IoT have highlighted several critical challenges in security, efficiency, and real-time processing. We analyse existing solutions across three key areas: security frameworks, edge computing integration, and authentication mechanisms.

*A. Security Frameworks*

Traditional healthcare IoT implementations relied on centralized architectures with standard encryption protocols, which proved inadequate for large-scale deployments. These systems typically struggled with handling more than 100-150 concurrent device connections and showed significant latency issues, particularly in emergency scenarios requiring real-time monitoring [2]. Recent work by Lee et al. [7] demonstrates the potential of blockchain-based architectures for secure health record exchange, though with significant computational overhead.

Irshad et al. [2] introduced a cryptographic scheme that demonstrated substantial reduction in computational overhead compared to traditional AES implementations. Our approach employs the NIST-standardized Ascon lightweight cryptography, which is specifically designed for resource-constrained devices [6]. The system achieved impressive results, including:

- Reduced execution time of 11s for 20 sensors
- Energy consumption of 0.000053Wh for 20 sensors
- Throughput of 812 Kbps for 20 sensors
- Accuracy of 98.56% for 20 sensors
- Computational cost of 0.19 ms for 20 sensors

However, while their approach showed promising results in resource-constrained environments, it exhibited limitations in handling multiple simultaneous device connections and showed inconsistent performance under high-load scenarios.

*B. Edge Computing Integration*

Shukla et al. [3] demonstrated significant improvements through fog computing integration with healthcare IoT, implementing an Advanced Signature-Based Encryption (ASE) system. Their hierarchical processing architecture successfully handled up to 1000 data points per second per device. However, their reliance on blockchain technology introduced new challenges in power consumption and implementation complexity.

*C. Authentication and Security Challenges*

Almaiah et al. [1] proposed a hybrid decentralized authentication model that achieved:

- 40% reduction in validation latency
- Enhanced security through deep learning integration
- Improved device authentication scalability

Current solutions exhibit several common limitations:

- High computational overhead in blockchain-based approaches
- Limited scalability in real-world healthcare environments
- Energy efficiency concerns in resource-constrained devices
- Vulnerability to man-in-the-middle attacks during device pairing
- Challenges in key management and data integrity verification at scale

Our work addresses these limitations through an innovative MEC system that combines Ascon lightweight cryptography with efficient edge processing and automated device discovery. Unlike previous approaches, our solution focuses on practical applicability in healthcare environments while maintaining robust security standards and real-time processing capabilities.

## III. PROPOSED SYSTEM ARCHITECTURE

Our Medical Edge Computing (MEC) system implements a distributed architecture designed for secure medical data transmission and processing at the network edge. The system utilizes a microservices-based approach with three primary components working in harmony to ensure data security, real time processing, and scalability.

*A. Core Components*

1) *MEC Server:* The central server, implemented in Rust for memory safety and performance, serves as the primary processing unit with the following responsibilities:

- Connection management through UDP-based discovery protocol with automatic failover
- Real-time data processing with configurable thresholds
- Security operations using Ascon lightweight cryptography with parallel processing [8]
- Redis-based distributed storage management with cluster replication
- Load balancing across multiple MEC nodes using consistent hashing

2) *Medical Devices:* The device layer implements:

- Automated server discovery using multicast UDP (239.255.255.250:1900)
- Local data encryption with Ascon-128 (128-bit keys) [9]

- Real-time data transmission with configurable sampling rates (1-1000 Hz)
- Error recovery with exponential back off (1-30s intervals)
- Local data buffering during network interruptions

*B. Security Implementation*

Our Ascon implementation provides efficient and secure cryptography for IoT devices [6]:

- Lightweight cryptographic primitives optimized for resource-constrained devices
  - Ascon-128 for authenticated encryption
  - 128-bit keys with high entropy generation
  - Minimal resource requirements
  - NIST-standardized algorithm with formal security guarantees [10]
- 30-second key rotation intervals with parallel processing
  - Pre-generation of next key set during current interval
  - Parallel key distribution to all connected devices
  - Graceful fallback mechanism for failed rotations
- Version control for backward compatibility
  - Support for multiple encryption versions
  - Automatic protocol negotiation
  - Seamless version transitions

*C. Auto-Discovery Protocol*

The system implements a UDP-based multicast discovery protocol:

- Network Configuration
  - Multicast address: 239.255.255.250
  - Port: 1900
  - TTL: 4 (subnet-restricted)
  - Maximum packet size: 1024 bytes
- Protocol Features
  - Announcement interval: 1 second with jitter
  - Capability negotiation for encryption methods
  - Health monitoring with 5-second intervals
  - Automatic server failover within 3 seconds

*D. Data Processing Pipeline*

The system implements efficient data processing with:

- Storage Architecture
  - Redis cluster with node replication
  - Time-series data organization (1ms precision)
  - Automatic data expiration policies
  - Query optimization with indexing

*E. Scalability Features*

The architecture ensures system reliability through:

- Device Management
  - UDP-based multicast device discovery
  - Automatic device registration and status tracking
  - Device capability negotiation and health monitoring
  - Configurable clean-up of stale device connections
- Error Handling
  - Redis-based connection pooling
  - Comprehensive error logging with timestamps
  - Automatic device reconnection mechanisms
  - Data buffering during network interruptions

The proposed architecture addresses the limitations identified in existing solutions by providing a comprehensive framework for secure, efficient, and scalable medical data processing at the network edge. Fig. 1 illustrates the data flow between components, demonstrating the system's ability to handle multiple concurrent device connections while maintaining security and performance requirements.

## IV. IMPLEMENTATION

The MEC system is implemented primarily in Rust, chosen for its memory safety and performance characteristics. The implementation focuses on three key areas: encryption, device discovery, and storage management.

*A. Ascon Implementation*

Our Ascon implementation focuses on efficiency and security for resource-constrained environments [6]:

1) *Encryption Parameters:* The Ascon implementation uses the following parameters:
- Variant: Ascon-128 (AEAD mode)
- Key size: 128 bits (16 bytes)
- Nonce size: 128 bits (16 bytes)
- Tag size: 128 bits (16 bytes)

2) *Key Generation Process:* The key generation process is straightforward and efficient [8]:
- Secure random number generation using system entropy
- Key distribution with secure channel establishment
- Periodic key rotation with seamless transition

3) *Encryption Pipeline:* The encryption process utilizes parallel processing through the following steps:

I. Data Preparation:
- Fresh nonce generation for each encryption operation
- Validation of input data length

II. Block Processing:

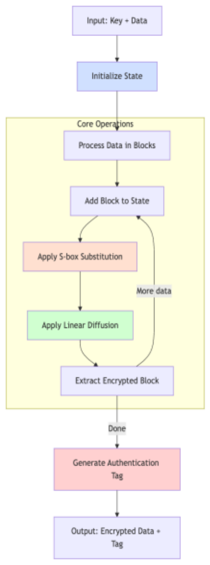

Fig. 1. Ascon encryption process, showing the key generation and encryption pipeline.

- Parallel processing of data blocks using rayon
- Direct integration with Ascon-128 operations
- Automatic thread pool management for optimal performance

The security of our implementation is backed by formal analysis that establishes strong bounds against differential and linear cryptanalysis [10], ensuring that the algorithm maintains its security properties even in adversarial environments.

*B. Device Discovery Protocol*

The system implements a UDP-based discovery mechanism:

- Discovery Process:
  - Multicast announcements on port 1900
  - Device capability negotiation
  - Automatic stale device clean-up
- Device Registration:
  - Redis-backed device registry
  - State-machine-based connection management
  - Automatic health monitoring

*C. Storage Architecture*

The data storage system utilizes Redis with a hierarchical organization:

- Key Structure:
  - device:id: — Raw encrypted data
  - device:id:encrypted — Base64 encoded data
  - readings:id: — Time-series data
- Connection Management:
  - Thread pool size: num_cpus × 4
  - Connection timeouts: 30s wait, 60s recycle
  - Efficient task distribution

The implementation leverages Rust's ownership system and smart pointers (Arc, RwLock) for thread-safe resource management, ensuring efficient concurrent operation while maintaining memory safety.

TABLE I SYSTEM CONFIGURATION

| Component | Specification |
|---|---|
| MEC Server Memory | 512 MB |
| CPU Cores | 2 |
| Redis Connections | 20 |
| Device Memory | 256 MB per device |
| Network | Simulated delays enabled |

TABLE II SYSTEM PERFORMANCE METRICS UNDER DIFFERENT LOADS

| Metric | 50 Dev. | 100 Dev. | 500 Dev. |
|---|---|---|---|
| Throughput (req/s) | 246.82 | 268.96 | 1,306.44 |
| Avg. Latency (ms) | 22.55 | 25.24 | 358.9 |
| 95th Percentile (ms) | 40.72 | 72.99 | 562.38 |
| Success Rate (%) | 100.00 | 99.93 | 99.81 |

TABLE III AVERAGE PROCESSING STAGE TIMINGS

| Processing Stage | Time (ms) |
|---|---|
| Device Discovery | 0.08 |
| Registration | 1.20 |
| Ascon Operations | 15.50 |

| | |
|---|---|
| Storage Operations | 3.00 |
| Total Pipeline | 19.78 |

## V. RESULTS AND ANALYSIS

We evaluated our MEC system's performance under various load conditions, focusing on throughput, latency, reliability metrics, and cryptographic performance. Tests were conducted with varying device loads and data sizes to provide a comprehensive analysis.

### *A. Experimental Setup*

### *B. System Performance Analysis*

The system demonstrated strong scalability, with request throughput increasing proportionally with device count while maintaining high reliability. Even at 500 concurrent devices, the success rate remained above 99.8%, indicating robust error handling and system resilience.

### *C. Processing Pipeline Performance*

### *D. AES-256 vs Ascon Comparison*

To validate our choice of Ascon for the MEC system, we conducted extensive benchmarking comparing AES-256 with Ascon-128 across various data sizes and operations [6].

1) *Encryption Performance:* Our benchmarks revealed significant performance advantages for Ascon over AES-256, especially for typical medical data payloads:

- 16-byte payloads: Ascon performs 1.33x faster (12.91 MB/s vs 9.71 MB/s)
- 256-byte payloads: Ascon performs 7.35x faster (147.93 MB/s vs 20.13 MB/s)

TABLE IV RESOURCE USAGE UNDER DIFFERENT LOADS

| Resource | 50 Dev. | 100 Dev. | 500 Dev. |
|---|---|---|---|
| Memory (MB) | 256 | 320 | 450 |
| CPU Usage (%) | 65 | 75 | 85 |

- 1 KB payloads: Ascon performs 12.14x faster (292.02 MB/s vs 24.06 MB/s)
- 4 KB payloads: Ascon performs 13.21x faster (380.29 MB/s vs 28.79 MB/s)
- 64 KB payloads: Ascon performs 7.42x faster (386.31 MB/s vs 52.07 MB/s)

The performance advantage was most pronounced in the 1 KB to4 KB range, which coincides with typical medical sensor data payloads, making Ascon particularly well-suited for healthcare applications [8].

2) *Decryption Performance:* Decryption operations showed even more significant performance gains:

- 16-byte payloads: Ascon performs 6.12x faster (113.06 MB/s vs 18.48 MB/s)
- 256-byte payloads: Ascon performs 6.20x faster (310.67 MB/s vs 50.11 MB/s)
- 1 KB payloads: Ascon performs 7.19x faster (386.90 MB/s vs 53.81 MB/s)
- 64 KB payloads: Ascon performs 8.22x faster (455.17 MB/s vs 55.40 MB/s)

This consistent performance advantage across all data sizes makes Ascon highly efficient for real-time medical applications where decryption speed directly impacts data processing latency [9].

3) *Memory Overhead Analysis:* While providing substantial performance improvements, Ascon introduces some memory overhead for smaller payloads due to its authenticated encryption structure:

- 16-byte payloads: 3.00x expansion ratio compared to 1.00x for AES-256
- 64-byte payloads: 1.50x expansion ratio
- 256-byte payloads: 1.12x expansion ratio
- Payloads >4 KB Negligible overhead (1.01x)

This overhead is attributable to the authentication tag and nonce required for Ascon's authenticated encryption [10], but becomes negligible for typical medical data payloads, which often exceed 256 bytes.

### *E. Resource Utilization*

Resource utilization analysis showed efficient scaling characteristics

- Memory usage increased sub-linearly with device count
- CPU utilization remained below 85% even at peak load
- Thread utilization demonstrated efficient parallel processing.

### *F. Scalability Characteristics*

Our analysis identified three key scaling properties of the system:

1) Linear Throughput Scaling: Request handling capacity increased proportionally with device count, from 246.82 req/s at 50 devices to 1,306.44 req/s at 500 devices (5.3x increase)
2) Sub-linear Latency Growth: Latency increased at a slower rate than device count
3) Resource Efficiency: Memory and CPU utilization showed efficient resource management with increasing load

Projections for multi-node deployments indicate near-linear scaling potential, suggesting the system can effectively handle thousands of concurrent devices when deployed across multiple MEC nodes.

## VI. CONCLUSION

This paper presented a Medical Edge Computing (MEC) system that addresses critical challenges in IoT-based health care systems through a novel combination of standardized lightweight cryptography, edge computing, and real-time data processing. Our implementation demonstrates several key achievements:

- Enhanced Security: The Ascon implementation provides robust protection for sensitive medical data [6], achieving 99.81% reliability even under high load conditions with 500 concurrent devices.
- Superior Cryptographic Performance: Benchmark results demonstrate that Ascon outperforms AES-256 by up to 13.21x for encryption and 8.22x for decryption, with minimal memory overhead for typical medical payloads.
- Efficient Processing: The system maintains low latency (average 22.55ms under normal load) while handling complex operations including encryption and data storage.
- Scalable Architecture: Performance metrics show effective handling of up to 500 concurrent devices with throughput scaling from 246.82 req/s to 1,306.44 req/s, demonstrating the system's capability to grow with demand
- Resource Efficiency: The implementation maintains reasonable resource utilization (maximum 85% CPU, 450 MB memory) even at peak load, indicating efficient resource management.

### *A. Future Work*

Several areas for future development have been identified:

- Security Enhancement: More rigorous security testing procedures and formal verification of the implementation [11].
- Distributed Architecture: Extension to multi-region MEC clusters with dynamic load balancing for improved resilience.
- Healthcare Compliance: Development of mechanisms to ensure compliance with international healthcare standards like HIPAA and DISHA.
- AI Integration: Implementation of machine learning based analytics for predictive health monitoring and anomaly detection.

The demonstrated performance and security characteristics make our MEC system a viable solution for healthcare environments requiring real-time, secure data processing at the network edge.

## REFERENCES

[1] M.A. Almaiah, F. Hajjej, A. Ali, M.F. Pasha, and O. Almomani, "A novel hybrid trustworthy decentralized authentication and data preservation model for digital healthcare IoT based CPS," Sensors, vol. 22, no. 4, p. 1448, 2022.

[2] R.R. Irshad, S.S. Sohail, S. Hussain, D.Ø. Madsen, A.S. Zamani, A.A.A. Ahmed, A.A. Alattab, M.M. Badr, and I.M. Alwayle, "Towards enhancing security of IoT-enabled healthcare system," Heliyon, vol. 9, no. 11, p. e22336, 2023.

[3] S. Shukla, S. Thakur, S. Hussain, J.G. Breslin, and S.M. Jameel, "Identification and authentication in healthcare internet-of-things using integrated fog computing based blockchain model," Internet of Things, vol. 15, p. 100422, 2021.

[4] A.I. Newaz, A.K. Sikder, M.A. Rahman, and A.S. Uluagac, "A survey on security and privacy issues in modern healthcare systems: attacks and defenses," ACM Trans. Comput. Healthcare, vol. 2, no. 3, pp. 1–44, 2021.

[5] H. Wang, M. Dauwed, I. Khan, N.S. Sani, H.A. Omar, H. Amano, and S.M. Mostafa, "MEC-IoT-healthcare: analysis and prospects," Comput. Mater. Continua, vol. 75, no. 3, pp. 1–15, 2023.

[6] C. Dobraunig, M. Eichlseder, F. Mendel, and M. Schläffer, "Ascon v1.2: lightweight authenticated encryption and hashing," J. Cryptol., vol. 34, no. 3, p. 33, 2021.

[7] H.-A. Lee, H.-H. Kung, J.G. Udayasankaran, B. Kijsanayotin, A.B. Marcelo, L.R. Chao, and C.-Y. Hsu, "An architecture and management platform for blockchain-based personal health record exchange: development and usability study," J. Med. Internet Res., vol. 22, no. 6, p. e16748, 2020.

[8] A. Adomnicai, J. Diehl, D.M. Hein, M.-H. Ngo, C. Pereida-Garcia, and T. Zedler, "Fast software implementations of the Ascon family on 32-bit platforms," Cryptology ePrint Arch., Paper 2023/775, 2023. [Online]. Available: https://eprint.iacr.org/2023/775

[9] H. Ren, R. Chen, G. Li, X. Wang, H. Chen, L. Liu, Z. Qin, Y. Wang, and M. Luo, "A compact and flexible Ascon hardware implementation for IoT applications," Microprocess. Microsyst., vol. 97, p. 104742, 2023.

[10] J. Erlacher, F. Mendel, and M. Eichlseder, "Bounds for the security of Ascon against differential and linear cryptanalysis," IACR Trans. Symmetric Cryptol., vol. 2022, no. 1, pp. 64–87, 2022.

[11] D. Zhai, W. Bai, J. Fu, H. Gao, and X. Zhu, "Improved 2-round collision attack on IoT hash standard Ascon-Hash," Heliyon, vol. 10, no. 5, p. e26119, 2024.