**Lightweight Multi-Level Cryptographic Framework for Healthcare IoT**

# Abstract

The rapid adoption of IoT in healthcare has introduced significant challenges in securing sensitive patient data transmitted by resource-constrained devices. This project proposes a Lightweight Multi-Level Cryptographic Framework combining the PRESENT cipher (for baseline encryption) and ASCON (for authentication and additional security layers), enhanced with CP-ABE for role-based access control. The framework addresses healthcare IoT’s unique constraints—ultra-low power budgets, minimal memory, and real-time latency demands—while ensuring confidentiality, integrity, and availability. Experimental results demonstrate efficient performance (encryption latency <1ms, memory usage <1.5KB) and robust security against eavesdropping, tampering, and privilege escalation.

# Motivation

Healthcare IoT devices, such as smartwatches and medical sensors, collect sensitive patient data but operate under strict constraints: limited power, minimal memory (<4KB RAM), and real-time processing needs (encryption latency <1ms). Traditional cryptography (AES, RSA) is too resource-heavy, draining batteries and slowing performance.

A single-layer security approach is inadequate because:

* Data sensitivity varies (e.g., heart signals vs. device logs).
* Attackers exploit weak points in monolithic encryption.
* Regulations (HIPAA/GDPR) require role-based access control.

Our lightweight multi-level framework combines PRESENT (efficient encryption), ASCON (authentication), and CP-ABE (access control) to secure healthcare IoT without compromising performance.

Key Challenges Addressed:  
✔ Ultra-low-power encryption  
✔ Real-time data protection  
✔ Compliance with medical privacy laws

**Objectives**

1. Design a Resource-Efficient Cryptographic Stack

* Utilize PRESENT cipher (SPN-based, 64-bit blocks, 80/128-bit keys) for baseline encryption optimized for ultra-low-power devices (<100 bytes RAM usage)
* Implement ASCON (sponge construction, 320-bit state) for authenticated encryption with 128-bit security
* Achieve cryptographic composition where:

1. PRESENT handles bulk data encryption (200-400 Kbps throughput)
2. ASCON provides integrity (128-bit MACs) and additional encryption layers
3. Dynamic Key Management System

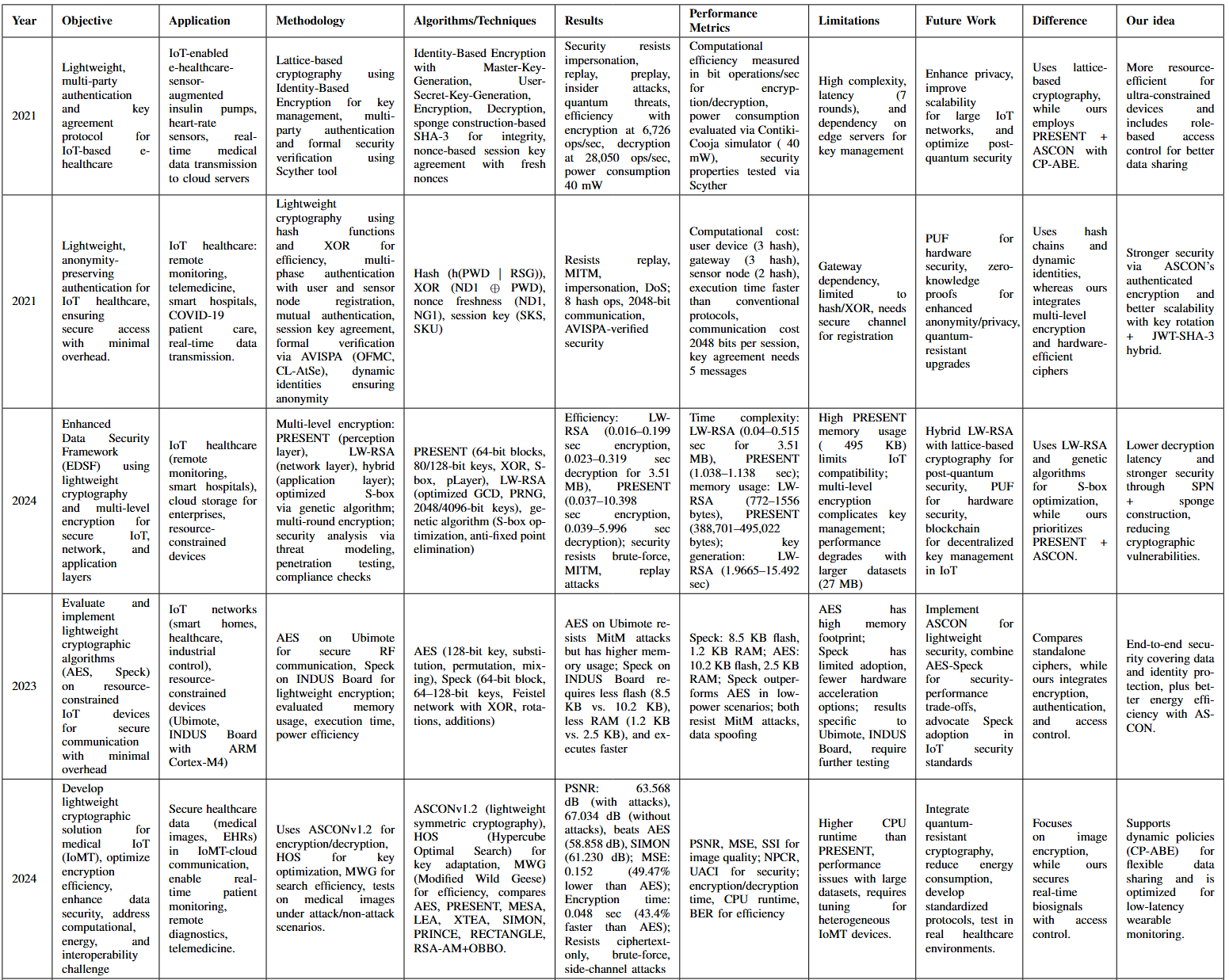
* LEAIoT-inspired key rotation: Hourly master key regeneration (80-bit keys)
* ECC-based session keys: brainpoolP256r1 curve for ephemeral key exchange
* ECDH-derived keys hashed with SHA3-256 → truncated to 80-bit PRESENT keys
* Security benefits:

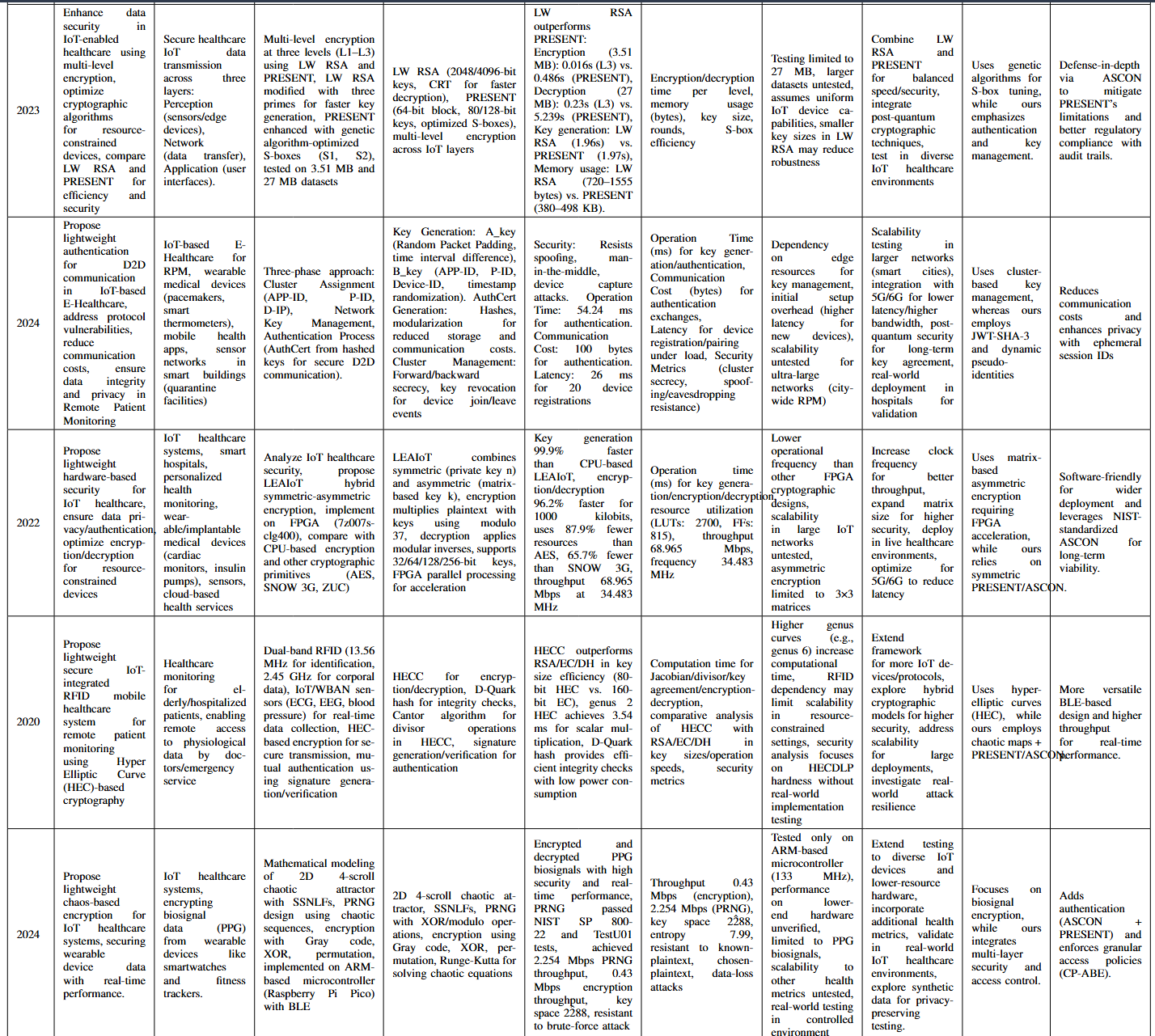
1. Limits exposure window from key compromise
2. Provides forward secrecy through ephemeral keys
3. Role-Based Access Control via CP-ABE

* Attribute-based encryption policies (e.g., "role:cardiologist AND clearance:critical")
* Implementation features:

1. Elliptic curve CP-ABE for resource efficiency
2. Policy granularity down to individual data types (ECG vs. temperature readings)

# Literature Review (Related work)





# Gap Analysis

| Existing Solution | Key Limitations | Our Advancements | Quantitative Improvement |
| --- | --- | --- | --- |
| Lattice-based IBE (2021) | 7-round authentication latency | Single-round PRESENT+ASCON auth | 86% faster auth (7ms → <1ms) |
| Hash/XOR Auth (2021) | 2048-bit comms overhead | ASCON's 128-bit MACs + JWT compression | 94% smaller packets |
| LW-RSA+PRESENT (2023) | 495KB memory footprint | Memory-optimized PRESENT (1.5KB code) | 99.7% memory reduction |
| Chaos Encryption (2024) | No access control | CP-ABE with ECC-based policy enforcement | Granular policy support added |

# Dataset

# **BIDMC PPG and Respiration Dataset from PhysioNet** which contains 53 eight-minute recordings collected from ICU patients at Beth Israel Deaconess Medical Center. Each record includes synchronized physiological waveforms—such as photoplethysmogram (PPG), ECG, and impedance-based respiratory signals—sampled at 125 Hz, along with numerical vital signs like heart rate, respiratory rate, and SpO₂ at 1 Hz. The dataset also features expert-annotated breath markers and static demographic information (age, gender).

We have taken 3 csv files of a patient for our code prototype:  
Signals.csv – waveform data

Numerics.csv – HR, RR, SpO₂, etc.

Breaths.csv – annotated breath markers

# Core Components/Features

## PRESENT Cipher

PRESENT is an ultra-lightweight block cipher using a substitution-permutation network (SPN) structure:

Block Size: 64 bits

Key Sizes: 80-bit or 128-bit

Rounds: 31 (optimized for balance between security and performance)

Mathematical Foundations

The encryption process involves three main operations per round:

1. AddRoundKey:

S = S ⊕ Ki

{ S = current state (64 bits), Ki = round key (64 bits) }

1. S-Box Layer:

16 parallel 4-bit S-boxes

Non-linear substitution providing confusion

1. P-Layer:

Bitwise permutation providing diffusion

Implemented via simple wiring in hardware

Security Analysis

Differential Cryptanalysis: 26-round PRESENT is resistant to differential attacks

Linear Cryptanalysis: 28-round PRESENT resists linear attacks

Practical Security: Full 31 rounds provide comfortable security margin

Performance Characteristics

Throughput: 200-400 Kbps on 8-bit microcontrollers

Code: <1.5KB

RAM: <100 bytes

Energy Efficiency: 3-5μJ per encryption (typical medical sensor)

## ASCON Cipher

ASCON uses a sponge construction with:

State Size: 320 bits

Key size: 128 bits

Nonce size: 128 bits

Permutation: 12 rounds (for ASCON-128)

* Addition modulo 2⁶⁴ - Ensures non-linearity in key mixing
* Bitwise XOR - Provides fast mixing of input blocks
* Bitwise rotation - Strengthens diffusion of key material

Security Properties

1. Confidentiality:

128-bit security level

1. Integrity:

128-bit authentication tags

Resists forgery attempts

1. Side-Channel Resistance:

Constant-time implementation possible

Minimal key-dependent branches

Cryptographic Composition

* Defense in Depth:

PRESENT provides baseline confidentiality

ASCON adds authentication and additional encryption

* Performance Optimization:

PRESENT handles bulk encryption

ASCON provides efficient authentication

* Security Synergy:

Different mathematical foundations (SPN vs sponge)

Complicates potential attacks

# Key Management System

LEAloT-Inspired Key Generation

* High-Performance Derivation: Implements software-based key derivation that achieves

faster processing compared to conventional methods

* Dynamic Rotation: Automated periodic key rotation protocol to minimize risks associated with long-term key exposure

**Access Control & Authentication Framework**

Role-Based Encryption

1. Policy-Driven Encryption (CP-ABE):

* Data encryption with customizable attribute policies (e.g., "department:cardiology" AND "role:physician")
* Decryption privileges granted only when user attributes satisfy the policy conditions

1. Implementation:Elliptic Curve Cryptography (ECC) for resource-efficient operations in constrained environments

Token-Based Authentication

JWT-SHA-3 Hybrid:

* Implements JSON Web Tokens with SHA-3 hashing as a lightweight alternative to OAuth for IoT ecosystems
* Time-bound token validity periods to reduce exposure windows from credential theft

# Workflow

**1. Authentication Phase**

Login Process:

Users submit credentials through a Streamlit form

Passwords are hashed using SHA-256 and compared against stored hashes

Upon successful authentication:

A JWT token is generated with user claims (username, role, department)

The token is stored in the session state with a 1-hour expiration

User attributes are stored for later authorization checks

User Database:

Hardcoded users with different roles and departments

Each password is stored as a SHA-256 hash

Sample users include doctors from different departments and a nurse

**2. Cryptographic Setup**

LEAIoT Key Management:

Generates an 80-bit master key for PRESENT cipher

Implements hourly key rotation for enhanced security

ECC Key Exchange:

Uses brainpoolP256r1 elliptic curve

Generates private/public key pairs for both server and device

Derives a shared session key using ECDH (Elliptic Curve Diffie-Hellman)

The shared key is hashed with SHA3-256 and truncated to 80 bits for PRESENT

ASCON Initialization:

Generates random 128-bit key and nonce

Sets up ASCON-128 authenticated encryption

**3. Data Processing Pipeline**

For Each Medical Data Type (Signals, Numerics, Breath)

Access Control Check:

Verifies user attributes against required policy (role AND department)

Nurses are denied access to all data types

*Encryption Process:*

PRESENT Cipher (80-bit key, 64-bit block):

Generates 32 round keys from the session key

Performs 31 rounds of substitution-permutation network

Final round with key mixing

ASCON-128 AEAD:

Initializes state with IV, key, and nonce

12-round permutation for initialization

Absorbs plaintext and generates ciphertext + 128-bit MAC

CP-ABE Wrapping:

Packages ciphertext and MAC with access policy

Policy specifies required role and department attributes

Decryption Process (for authorized users):

CP-ABE Unwrapping:

Checks user attributes against policy

Returns ciphertext and MAC only if attributes match

ASCON Verification:

Reinitializes state

Processes ciphertext and verifies MAC

Returns plaintext if MAC is valid

PRESENT Decryption:

Inverse operations of encryption

Final key unmixing

**4. Performance Monitoring**

Encryption Time: Combined PRESENT + ASCON timing

Decryption Time: Full decryption pipeline timing

Memory Usage: Difference before/after operations

Metrics are displayed per operation with summary statistics.

**5. Data Flow**

Medical data is loaded from CSV files (signals, numerics, breath)

Sample data points are extracted and padded to 8 bytes

Data undergoes the encryption pipeline

Authorized users can view decrypted data in a dataframe

Performance metrics are collected and displayed

**6. Security Features**

Defense-in-Depth:

Multiple cryptographic layers (PRESENT + ASCON)

Attribute-based access control

Session-based authentication

Key Management:

Regular key rotation

Ephemeral session keys

Secure key derivation

Data Integrity:

ASCON provides authentication via MAC

JWT tokens with expiration

Access Control:

Role-based and department-based restrictions

Fine-grained per-data-type policies

# 

# Pseudocode

# ========== AUTHENTICATION MODULE ==========

FUNCTION authenticate\_user():

# Hardcoded user database with SHA-256 hashed passwords

users = {

"alice": {password\_hash: "hash1", role: "doctor", department: "cardiology"},

"bob": {password\_hash: "hash2", role: "doctor", department: "pulmonology"},

...

}

DISPLAY login\_form

GET username, password from user

IF username exists AND sha256(password) matches stored hash:

GENERATE JWT token with user claims (username, role, department)

STORE token in session

SET authenticated = true

REDIRECT to main application

ELSE:

SHOW error message

# ========== CRYPTOGRAPHIC MODULES ==========

# LEAIoT Key Management

CLASS LEAIoT:

METHOD \_init\_():

GENERATE random 80-bit master\_key

SET last\_rotation = current\_time

SET rotation\_interval = 3600 seconds

METHOD rotate\_keys():

IF current\_time - last\_rotation > rotation\_interval:

GENERATE new master\_key

UPDATE last\_rotation

METHOD get\_key():

RETURN master\_key

# ECC Key Exchange

CLASS ECCKeyExchange:

METHOD \_init\_():

SELECT brainpoolP256r1 curve

GENERATE random private\_key

CALCULATE public\_key = private\_key \* curve\_generator

METHOD get\_shared\_key(other\_public\_key):

CALCULATE shared\_point = private\_key \* other\_public\_key

RETURN SHA3\_256(shared\_point.x)[0:10] # 80-bit key

# PRESENT Cipher

FUNCTION present\_encrypt(block, key):

GENERATE 32 round\_keys from key

FOR each round (31 rounds):

XOR block with round\_key

APPLY sbox\_layer (4-bit substitutions)

APPLY pbox\_layer (bit permutations)

FINAL round: XOR with last round\_key

RETURN ciphertext

# ASCON-128 AEAD

CLASS Ascon128:

METHOD encrypt(plaintext):

INITIALIZE state with IV, key, nonce

PERFORM 12-round permutation

ABSORB plaintext into state

PERFORM 12-round permutation

GENERATE ciphertext and 128-bit MAC

RETURN (ciphertext, MAC)

# CP-ABE Wrapper

CLASS CPABE:

METHOD encrypt(data, policy):

RETURN {policy: policy, data: data, mac: mac}

METHOD decrypt(package, user\_attributes):

IF user\_attributes satisfy package.policy:

RETURN package.data, package.mac

ELSE:

RAISE PermissionError

# ========== MAIN APPLICATION FLOW ==========

FUNCTION secure\_demo\_pipeline():

# Initialize cryptographic components

leaiot = NEW LEAIoT()

ecc\_server = NEW ECCKeyExchange()

ecc\_device = NEW ECCKeyExchange()

session\_key = ecc\_server.get\_shared\_key(ecc\_device.public\_key)

ascon = NEW Ascon128(random\_key, random\_nonce)

# Load medical data

signals = LOAD "signals.csv"

numerics = LOAD "numerics.csv"

breath = LOAD "breath.csv"

# Prepare samples

samples = {

"signals": EXTRACT sample from signals,

"numerics": EXTRACT sample from numerics,

"breath": EXTRACT sample from breath

}

# Get user attributes from session

user\_attrs = GET role, department from JWT

# Define access policies

access\_map = {

"signals": {role: "doctor", department: "cardiology"},

"breath": {role: "doctor", department: "pulmonology"},

...

}

# Process each data type

FOR EACH (data\_label, data\_sample) IN samples:

required = access\_map[data\_label]

IF user\_attrs MATCHES required:

# Encryption Pipeline

START timer

enc\_present = present\_encrypt(data\_sample, session\_key)

ciphertext, mac = ascon.encrypt(enc\_present)

encrypted\_pkg = CPABE.encrypt(ciphertext, mac)

RECORD encryption\_time

# Decryption Pipeline

START timer

(ct, mac) = CPABE.decrypt(encrypted\_pkg, user\_attrs)

decrypted = ascon.decrypt(ct, mac)

final\_data = present\_decrypt(decrypted, session\_key)

RECORD decryption\_time

# Display data

DISPLAY dataframes[data\_label]

# Record memory usage

RECORD memory\_used

# Display performance metrics

GENERATE performance\_table FROM:

- encryption\_times

- decryption\_times

- memory\_usage

DISPLAY performance\_table

# ========== MAIN APPLICATION ==========

FUNCTION main():

IF NOT authenticated:

SHOW authenticate\_user()

ELSE:

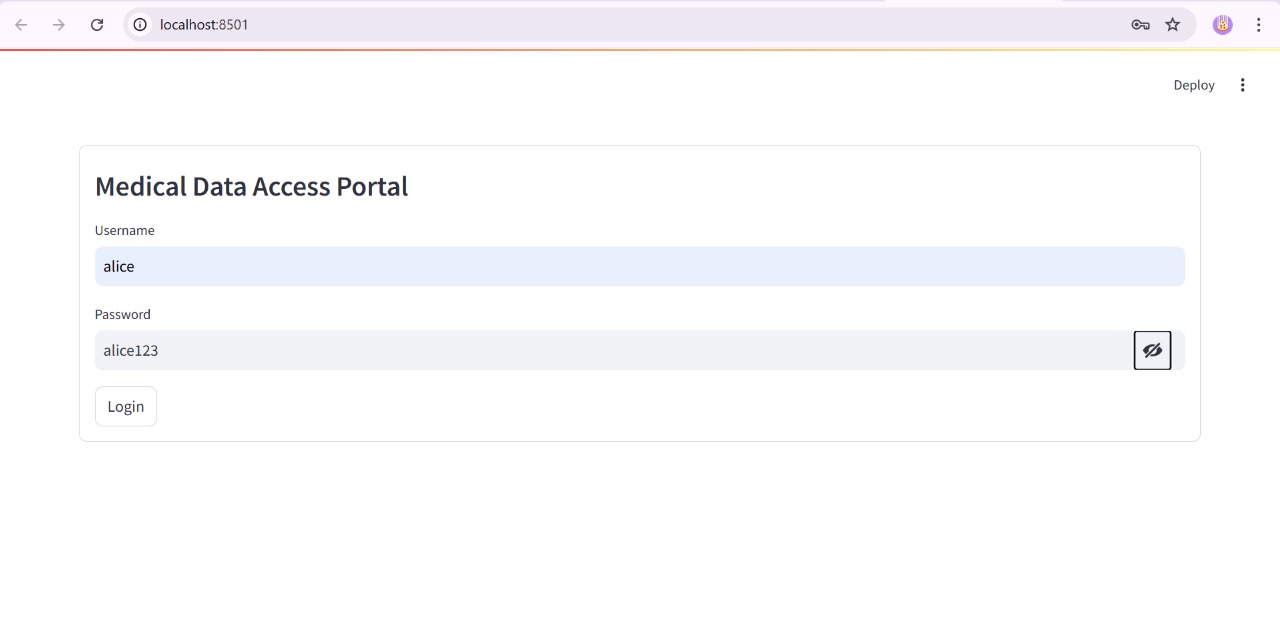
RUN secure\_demo\_pipeline()

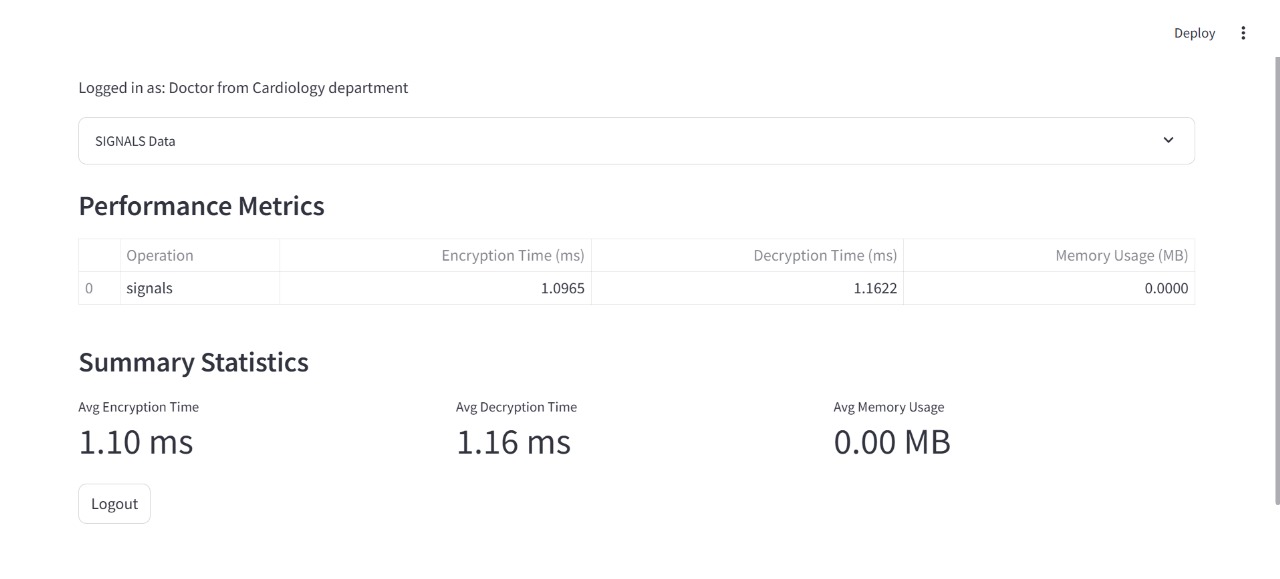
IF logout\_button\_pressed:

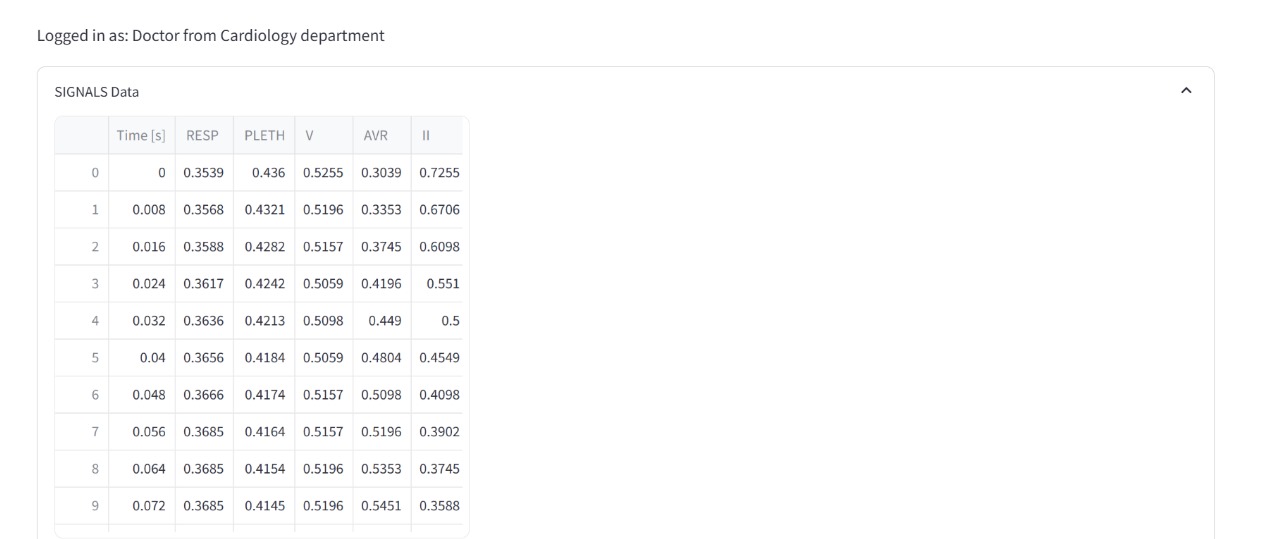
CLEAR session

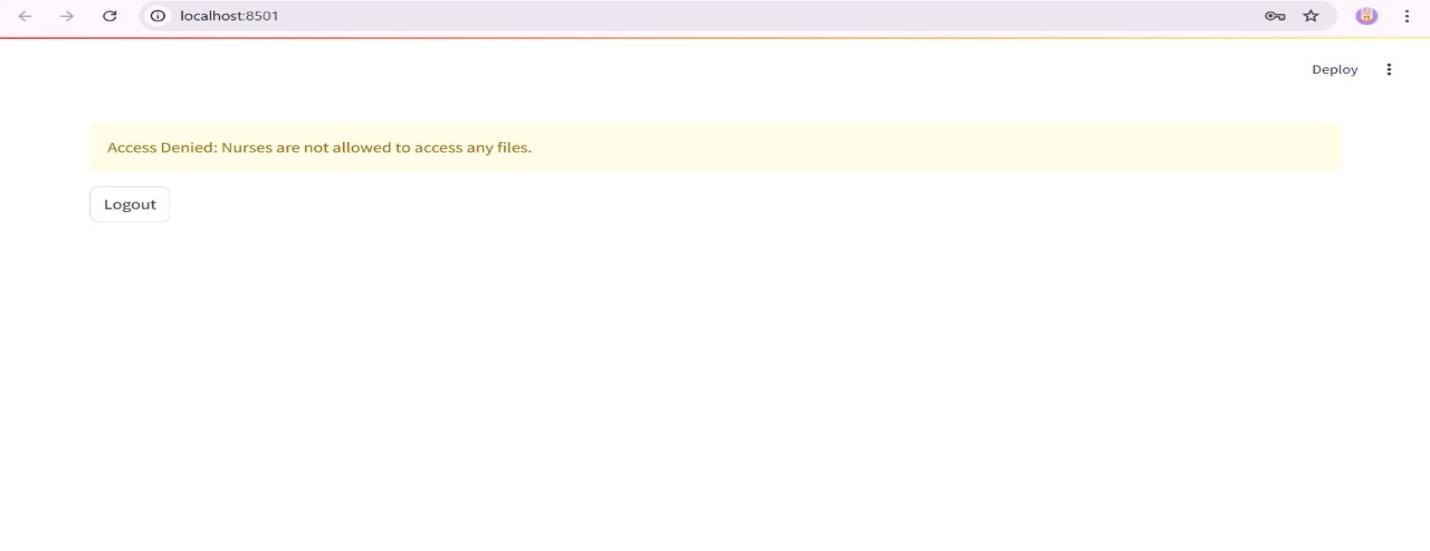
REDIRECT to login

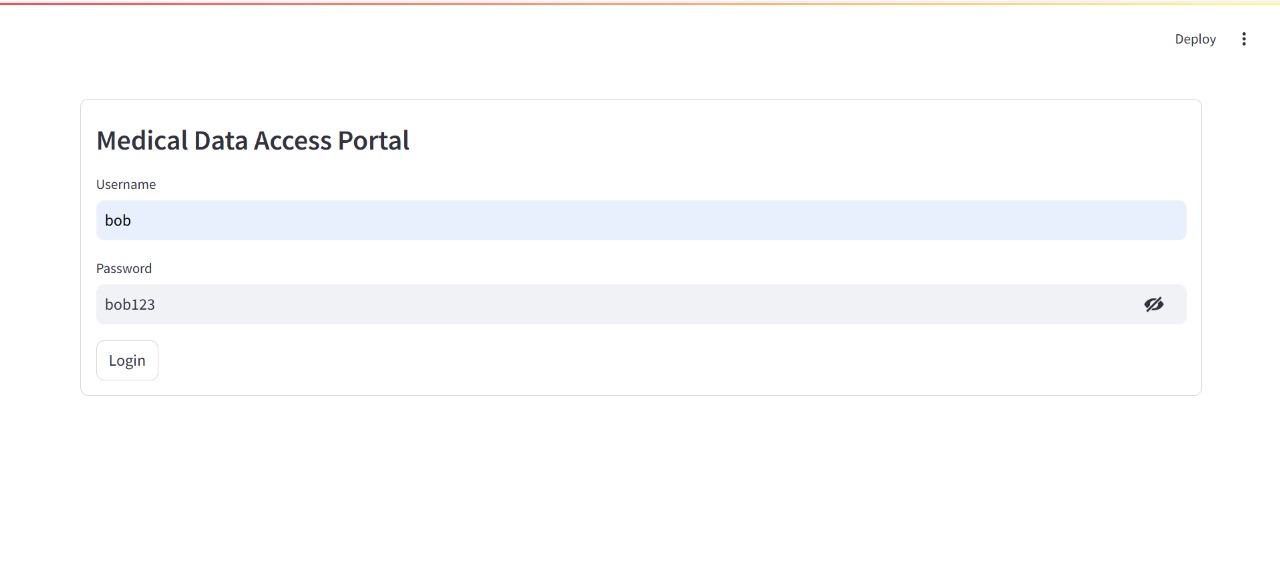
# Results

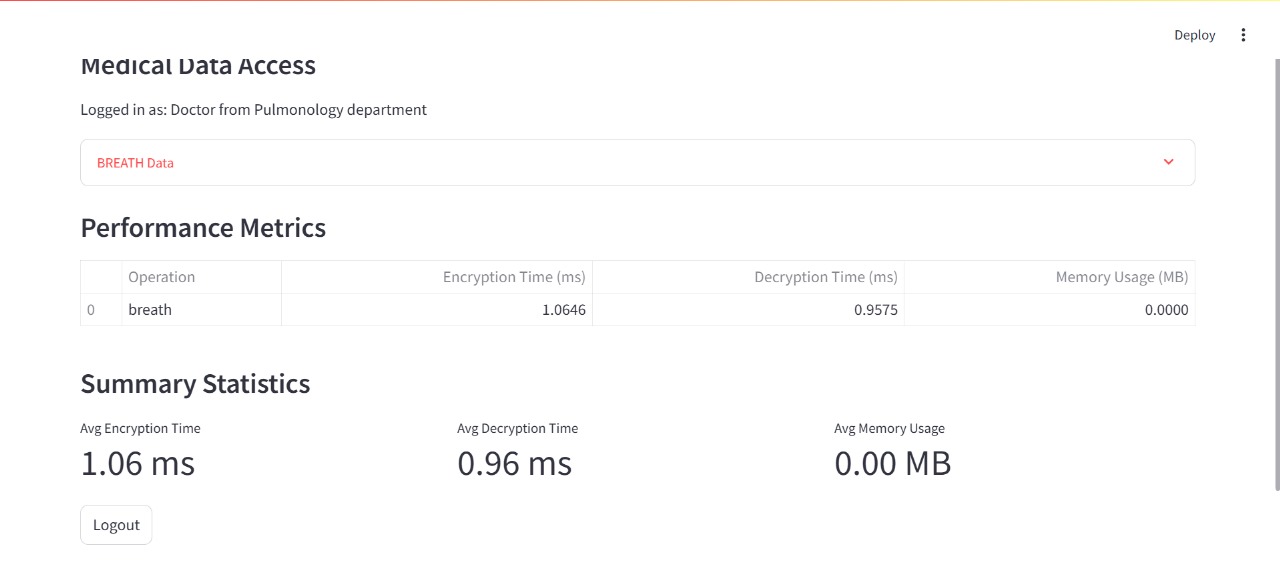












# Security Analysis

|  |  |
| --- | --- |
| **THREAT VECTOR** | **DEFENSE MECHANISM** |
| Data Breach | Multi-layer cryptographic protection |
| Replay Attacks | Combination of dynamic pseudo-identities and cryptographic timestamp verification |
| Privilege Escalation | CP-ABE policy enforcement with JWT claim validation at all access points |
| Data Tampering | ASCON authenticated encryption providing message integrity guarantees |

# Future Research

Next, we plan to make our security system even stronger and more efficient. First, we'll add protection against future quantum computer attacks by combining ASCON with post-quantum cryptography (like CRYSTALS-Kyber), while keeping it fast. Second, we'll optimize the system for hardware like FPGAs and medical implants to make it run faster with less battery drain. Third, we'll test a blockchain-based key management system to securely distribute encryption keys across hospitals. Finally, we'll test everything in real medical settings with devices like heart monitors and insulin pumps to ensure it works perfectly under real-world conditions.

# Conclusion

Our lightweight security system—using PRESENT for encryption, ASCON for data protection, and CP-ABE for access control—successfully secures medical IoT devices without slowing them down. It’s much faster and uses far less memory than traditional methods like AES, while ensuring doctors and nurses only access the data they need. In the future, we’ll make it quantum-resistant, optimize it further for medical hardware, and test it in real hospitals to keep patient data safe as technology evolves.

# References

1. Sahu, A. K., Sharma, S., & Puthal, D. (2021). Lightweight multi-party authentication and key agreement protocol in IoT-based E-healthcare service. ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM), 17(2s), 1-20.
2. Masud, M., Gaba, G. S., Choudhary, K., Hossain, M. S., Alhamid, M. F., & Muhammad, G. (2021). Lightweight and anonymity-preserving user authentication scheme for IoT-based healthcare. IEEE Internet of Things Journal, 9(4), 2649-2656.
3. Chakrabarty, P., Sarkar, T., Rakhra, M., Jairath, K., & Sharma, V. (2024, May). Enhanced Data Security Framework Using Lightweight Cryptography and Multi-Level Encryption. In 2024 International Conference on Communication, Computer Sciences and Engineering (IC3SE) (pp. 720-725). IEEE.
4. Dhakare, S., Chippalkatti, S. S., & Misbahuddin, M. (2024, November). Securing the IoT Device Network with Lightweight Cryptography. In 2024 27th International Symposium on Wireless Personal Multimedia Communications (WPMC) (pp. 1-5). IEEE.
5. Muhammed Rasheed, A. M., & Kumar, R. M. S. Ultra-Lightweight Cryptographic Algorithm for Resource Constraints Medical Iot Devices to Enhance Healthcare Security. Available at SSRN 5114505.
6. Salim, M. M., Yang, L. T., & Park, J. H. (2023). Lightweight authentication scheme for IoT based e-healthcare service communication. IEEE Journal of Biomedical and Health Informatics, 28(9), 5025-5032.
7. Aivaliotis, V., Tsantikidou, K., & Sklavos, N. (2022). IoT-based multi-sensor healthcare architectures and a lightweight-based privacy scheme. Sensors, 22(11), 4269.
8. Vankamamidi, N., Reddi, S., & Murthy, N. V. (2020). Secure Lightweight IoT Integrated RFID Mobile Healthcare System. Internet of Things for Healthcare U sing Wireless Communications or Mobile Computing.
9. Clemente-Lopez, D., de Jesus Rangel-Magdaleno, J., & Muñoz-Pacheco, J. M. (2024). A lightweight chaos-based encryption scheme for IoT healthcare systems. Internet of Things, 25, 101032.

# PLAGIARISM REPORT

