National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

Enhanced IoMT Authentication Protocol with Forward Secrecy Using Hybrid Hierarchical Curve25519 ECDH

Stream: Research-Based Project backed by Code-Based Implementation

Department of Computing
CS-381: Network Security (3+0)
BESE-13AB Fall 2025

Instructor: Ms. Maryam Sajjad
Submitted by:
Muhammad Owais khan (404262)

Table of Contents

ABSTRACT

The Internet of Medical Things (IoMT) represents a rapidly growing ecosystem of connected healthcare devices that collect, transmit, and process sensitive patient data. Securing communication in these resource-constrained environments presents unique challenges, particularly in maintaining strong cryptographic guarantees while respecting computational limitations. This project analyzes and enhances Zhou et al.'s three-factor authentication protocol for IoMT environments by addressing a critical security vulnerability: the lack of forward secrecy. We propose a Hybrid Hierarchical ECDH scheme using Curve25519, which provides bounded forward secrecy for device-gateway communication (24-hour windows) while maintaining computational efficiency suitable for constrained IoT devices. Our implementation demonstrates successful MITM attack resistance, proper key derivation using HKDF, and efficient session key establishment with only 25-30% additional computational overhead compared to the original protocol.

INTRODUCTION

2.1 Background

The Internet of Medical Things (IoMT) has revolutionized healthcare delivery by enabling continuous patient monitoring, remote diagnostics, and automated medical interventions. These connected medical devices—ranging from wearable health monitors to implantable cardiac devices—generate and transmit sensitive health data that requires robust protection against unauthorized access and tampering.

Healthcare data represents one of the most sensitive categories of personal information, protected by regulations such as HIPAA (Health Insurance Portability and Accountability Act) in the United States and GDPR (General Data Protection Regulation) in Europe. A security breach in IoMT systems could result in:

- **Patient privacy violations**: Exposure of medical conditions, treatments, and health histories
- **Medical identity theft**: Fraudulent use of patient information for insurance claims
- **Physical harm**: Manipulation of medical device readings or commands
- **Legal liability**: Significant fines and lawsuits for healthcare providers

2.2 Authentication in IoMT

Authentication protocols for IoMT must balance security requirements with the practical constraints of medical devices:

- **Limited computational power**: Many medical sensors operate on low-power microcontrollers
- **Battery constraints**: Cryptographic operations consume energy, affecting device longevity

- **Real-time requirements**: Medical monitoring often requires immediate data transmission
- **Network reliability**: Healthcare environments may have intermittent connectivity

Zhou et al. proposed a three-factor authentication protocol specifically designed for IoMT environments, incorporating:

1. **Something the user knows**: Password
2. **Something the user has**: Smart device/token
3. **Something the device is**: Physical Unclonable Function (PUF)

2.3 Project Objectives

This project aims to:

1. Analyze the security properties of Zhou et al.'s IoMT authentication protocol
2. Identify the lack of forward secrecy as a critical vulnerability
3. Design and implement an enhanced protocol with forward secrecy
4. Demonstrate the effectiveness of the solution through comprehensive testing
5. Provide a functional GUI for protocol demonstration and testing

PROBLEM STATEMENT

3.1 The Forward Secrecy Gap

Zhou et al.'s protocol, while providing strong mutual authentication and session key establishment, lacks **forward secrecy**—a critical security property defined as: *If long-term keys are compromised at time T, all session keys established before time T remain secure.*

In the original protocol:

- Session keys are derived from static credentials (password hash k, device secrets)
- Compromise of the gateway's database exposes all historical session keys
- An attacker with recorded network traffic can decrypt past communications

3.2 Real-World Implications

Consider the following attack scenario:

1. **Data Collection Phase**: An attacker passively records encrypted communications between a patient's medical device and the hospital gateway over several months
2. **Compromise Event**: The attacker later gains access to the gateway's credential database through a separate vulnerability
3. **Retrospective Decryption**: Using the compromised credentials, the attacker derives all historical session keys and decrypts months of medical data

This attack is particularly concerning because:

- Healthcare databases are frequent targets (693 breaches reported in 2024 alone)
- Medical data has long-term value (conditions, genetic information)

- Patients cannot change their medical history like they can change a password

3.3 Challenge

The challenge is to add forward secrecy while:

- Maintaining compatibility with resource-constrained IoT devices
- Minimizing additional computational and communication overhead
- Preserving the existing security properties of the protocol
- Ensuring resistance to man-in-the-middle (MITM) attacks

LITERATURE REVIEW

4.1 Zhou et al.'s Protocol (2024)

Zhou et al. proposed a three-factor authentication scheme for IoMT with the following phases:

Registration Phase:

- User registers with gateway using password and biometric
- Sensor registers using Physical Unclonable Function (PUF)
- Gateway stores credentials and issues pseudonymous identifiers

Authentication Phase:

- User initiates with M1 = {N, α, DID, SID}
- Gateway verifies user, communicates with sensor via M2/M3
- Gateway returns M4 = {SKi, λ} with session key

Security Properties:

- Mutual authentication
- User anonymity
- PUF-based device authentication
- Forward secrecy (missing)

4.2 Forward Secrecy Mechanisms

Diffie-Hellman Key Exchange (DH):

- Provides forward secrecy through ephemeral key pairs
- Original DH uses multiplicative groups (computationally expensive)
- Vulnerable to MITM without authentication

Elliptic Curve Diffie-Hellman (ECDH):

- Same security with smaller key sizes (256-bit ECC ≈ 3072-bit RSA)
- Multiple curve options: NIST P-256, Curve25519, secp256k1
- 10-15x faster than RSA-based alternatives

Curve25519 (X25519):

- Designed by Daniel J. Bernstein for security and performance
- Montgomery curve with efficient constant-time implementation
- 25-30% faster than NIST P-256
- Resistant to timing attacks by design

4.3 Key Derivation Functions

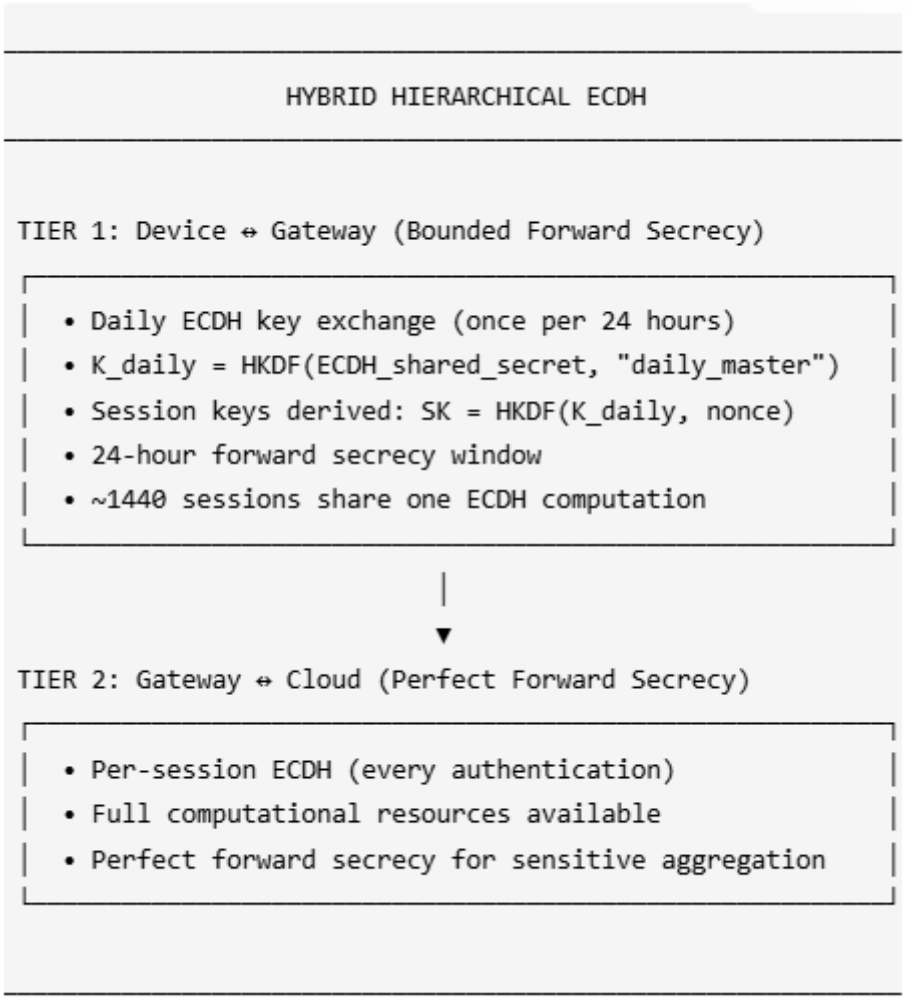
HKDF (HMAC-based Key Derivation Function):

- RFC 5869 standard for deriving cryptographic keys
- Two phases: Extract (concentrates entropy) and Expand (generates keys)
- Allows binding additional context (session nonces, identifiers)

PROPOSED SOLUTION

5.1 Hybrid Hierarchical ECDH Architecture

We propose a two-tier forward secrecy architecture optimized for IoMT constraints:



5.2 Why Curve25519?

Property	Curve25519	NIST P-256	RSA-2048
Key Size	256 bits	256 bits	2048 bits
Security Level	128 bits	128 bits	112 bits
Performance	Fastest	Fast	Slow
Timing Attack Resistance	By design	Requires care	Requires care
Patent Status	Public domain	Complex	Various

5.3 Protocol Enhancement

Enhanced M1 Message:

M1 = {N, α, DID, SID, session_nonce, ephemeral_public_key}
Where:
- session_nonce: Fresh 16-byte random value for HKDF
- ephemeral_public_key: User's Curve25519 public key (32 bytes)
- α = h(b_i || k || DID || SID || session_nonce || ephemeral_public_key)

Enhanced M4 Message:

M4 = {SKi, λ, gateway_ephemeral_public_key}

Where:
- gateway_ephemeral_public_key: Gateway's Curve25519 public key
- λ = h(SK || DID || k || DID_new || SID_new || session_nonce || Q_u || Q_g)

Session Key Derivation:

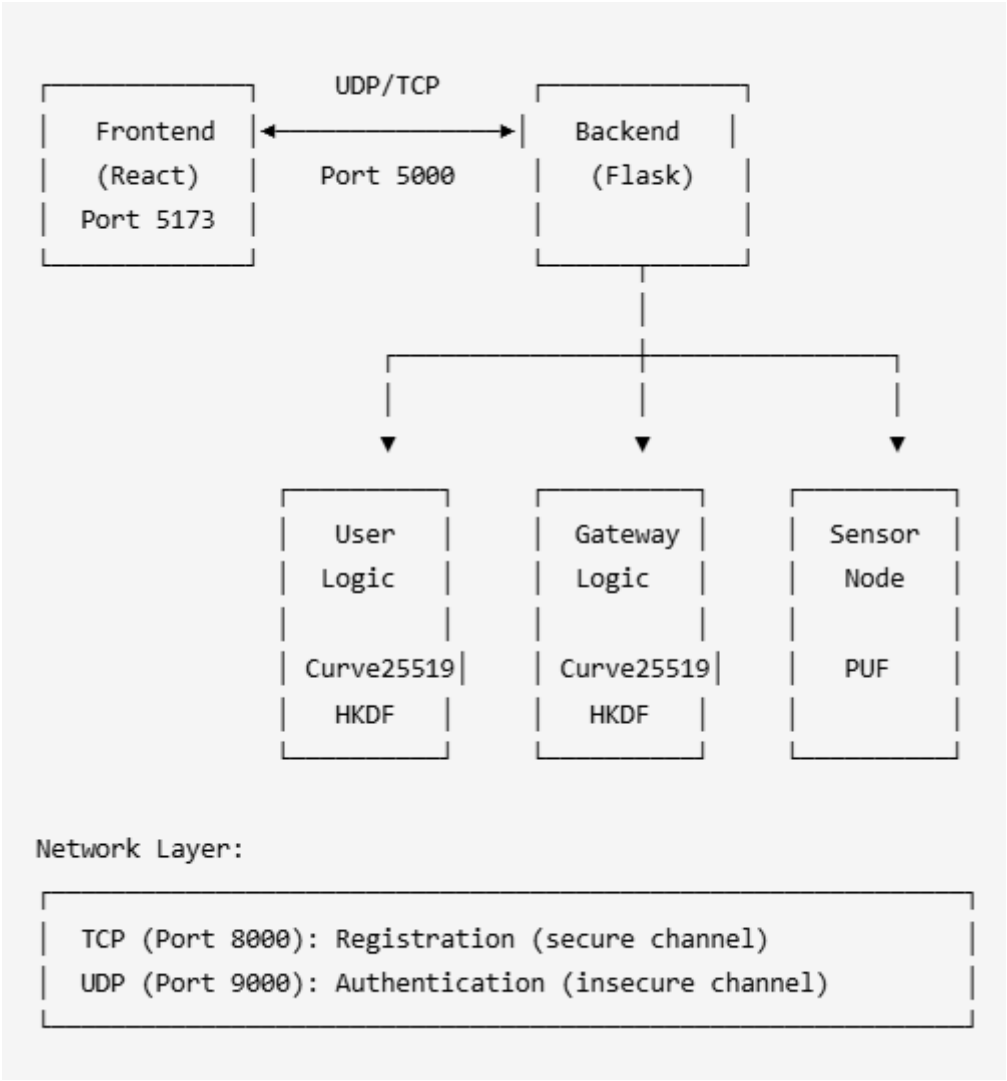
Z = ECDH(user_private, gateway_public) // Shared secret
K_daily = HKDF-Extract(Z, "daily_master_key")
SK_session = HKDF-Expand(K_daily, session_nonce, "user_gateway_session")
SK_final = h(SK_base || SK_session)

5.4 Security Properties Achieved

Property	Original	Enhanced
Mutual Authentication	Yes	Yes
User Anonymity	Yes	yes
Session Key Security	Yes	Yes
Forward Secrecy	No	Yes (24h bounded)
MITM Resistance	Yes	yes (authenticated ECDHE)
Replay Attack Resistance	Yes	Yes

SYSTEM ARCHITECTURE

6.1 Component Overview

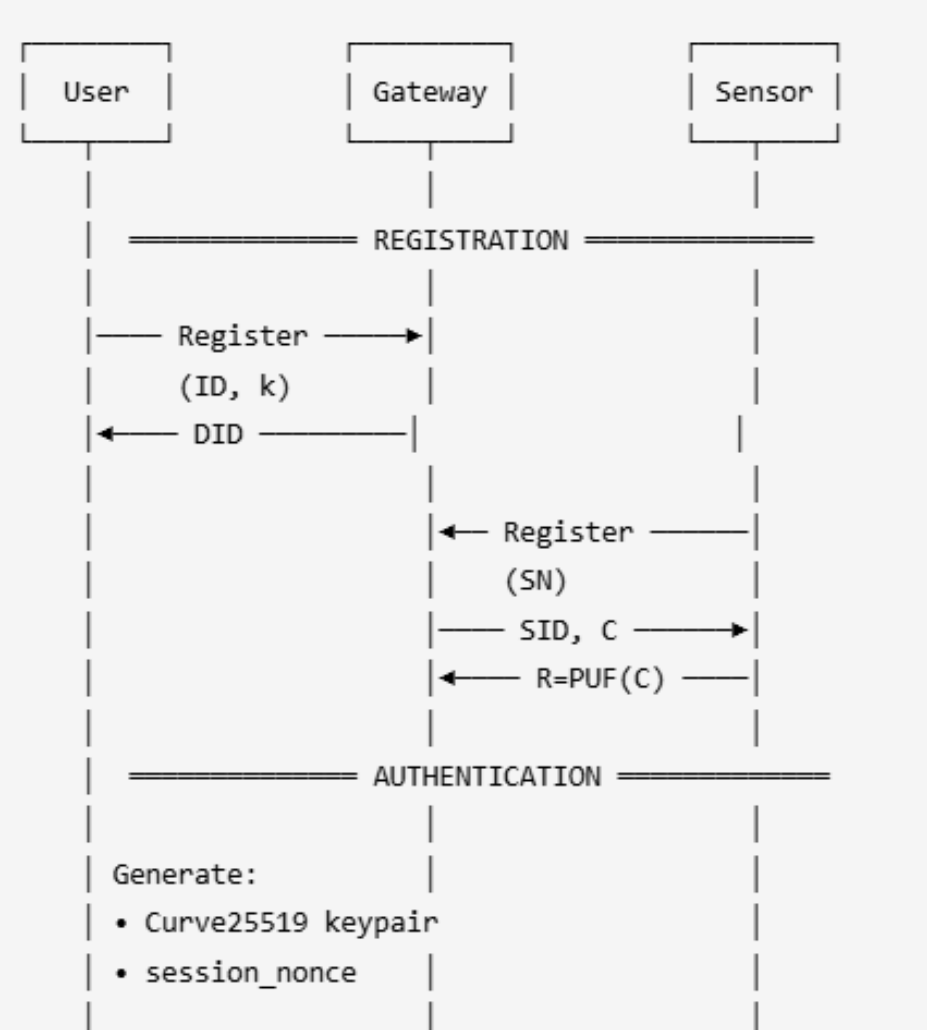


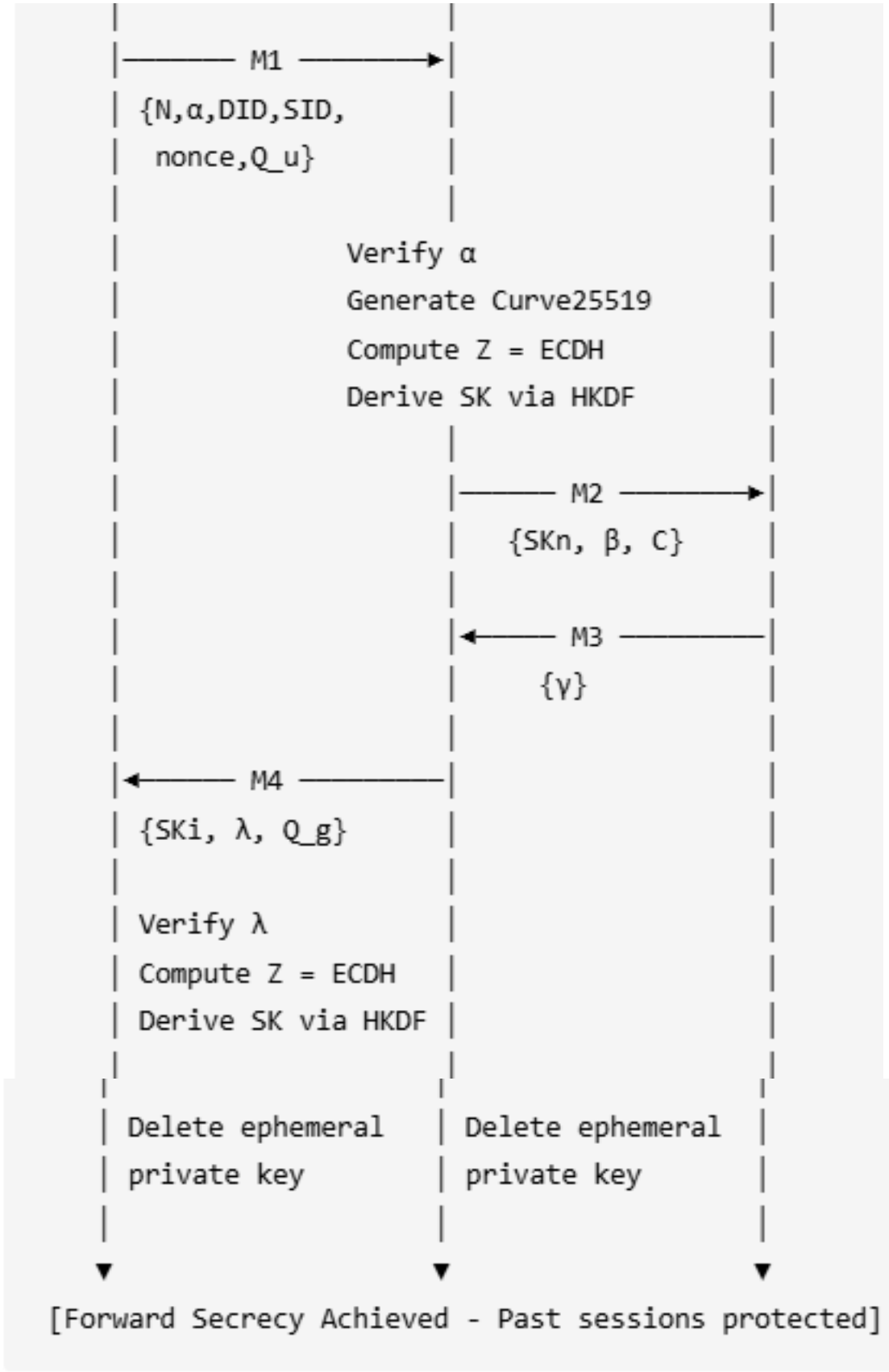
6.2 Directory Structure

07 Project Improvement and Attack/
├── backend/
│ ├── api_server.py # Flask REST API + Gateway server
│ ├── gateway_server.py # Standalone gateway (TCP/UDP)
│ ├── sensor_node.py # Sensor simulation with PUF
│ ├── user_client.py # User client logic
│ ├── requirements.txt # Python dependencies
│ ├── test_forward_secrecy.py # FS test suite
│ ├── test_mitm_attack.py # MITM resistance tests
│ ├── protocol/
│ │ ├── common.py # Crypto primitives (Curve25519, HKDF)

```
│   ├── gateway_logic.py   # Gateway protocol implementation
│   ├── sensor_logic.py    # Sensor protocol implementation
│   └── user_logic.py      # User protocol implementation
└── network/
    ├── secure_channel.py  # TCP wrapper
    └── insecure_channel.py # UDP wrapper
├── frontend/
│   ├── src/
│   │   ├── App.jsx         # Main React component
│   │   ├── App.css         # Styling
│   │   └── main.jsx        # Entry point
│   ├── package.json        # Node dependencies
│   └── vite.config.js      # Vite configuration
└── Documentation.md        # This file
```

6.3 Protocol Flow




M1
{N,α,DID,SID,
nonce,Q_u}
Verify α
Generate Curve25519
Compute Z = ECDH
Derive SK via HKDF
M2
{SKn, β, C}
M3
{γ}
M4
{SKi, λ, Q_g}
Verify λ
Compute Z = ECDH
Derive SK via HKDF
Delete ephemeral private key
Delete ephemeral private key
[Forward Secrecy Achieved - Past sessions protected]

IMPLEMENTATION DETAILS

7.1 Cryptographic Primitives (common.py)

Curve25519 Key Generation:

```
from cryptography.hazmat.primitives.asymmetric.x25519 import X25519PrivateKey

def generate_curve25519_keypair():
    """Generate Curve25519 ephemeral keypair for forward secrecy."""
    private_key = X25519PrivateKey.generate()
    public_key_bytes = private_key.public_key().public_bytes(
        encoding=serialization.Encoding.Raw,
        format=serialization.PublicFormat.Raw
    )
    return private_key, public_key_bytes
```

ECDH Shared Secret:

```
def compute_curve25519_shared_secret(private_key, peer_public_key_bytes):
    """Compute shared secret using X25519 ECDH."""
    peer_public_key = X25519PublicKey.from_public_bytes(peer_public_key_bytes)
    shared_secret = private_key.exchange(peer_public_key)
    return shared_secret  # 32 bytes
```

HKDF Key Derivation:

```
def derive_session_key(shared_secret, session_nonce, context):
    """Derive session key using HKDF with session nonce binding."""
    hkdf = HKDF(
        algorithm=hashes.SHA256(),
        length=32,
        salt=session_nonce,
        info=context,
    )
    return hkdf.derive(shared_secret)
```

7.2 Daily Key Caching

```python
# Cache structure: {entity_pair: (K_daily, timestamp, public_key)}
_daily_key_cache = {}
DAILY_KEY_VALIDITY = 24 * 60 * 60  # 24 hours in seconds

def get_or_establish_daily_key(entity_id, peer_public_key):
    """Get cached daily key or establish new one via ECDH."""
    cache_key = entity_id
    current_time = time.time()

    if cache_key in _daily_key_cache:
        k_daily, timestamp, cached_pub = _daily_key_cache[cache_key]
        if (current_time - timestamp) < DAILY_KEY_VALIDITY:
            return k_daily, False  # Use cached key

    # Establish new daily key via full ECDH
    private_key, public_key = generate_curve25519_keypair()
    shared_secret = compute_curve25519_shared_secret(private_key, peer_public_key)
    k_daily = derive_daily_master_key(shared_secret)

    _daily_key_cache[cache_key] = (k_daily, current_time, public_key)
    return k_daily, True  # New key established
```

7.3 MITM-Resistant Authentication

The key to MITM resistance is **authenticated ECDHE**—binding the ephemeral public keys to the authentication hash:

User Side (user_logic.py):

```python
# Alpha includes ephemeral key + session nonce
alpha = h(b_i + k + DID + SID + session_nonce + ephemeral_public_key)
```

Gateway Verification (gateway_logic.py):

```python
# Verify alpha with claimed ephemeral key
alpha_prime = h(b_i_new_prime + k + DID + SID + session_nonce + user_ephemeral_public_key)
if alpha_prime != M1["alpha"]:
    return {"error": "Authentication failed - possible MITM attack detected!"}
```

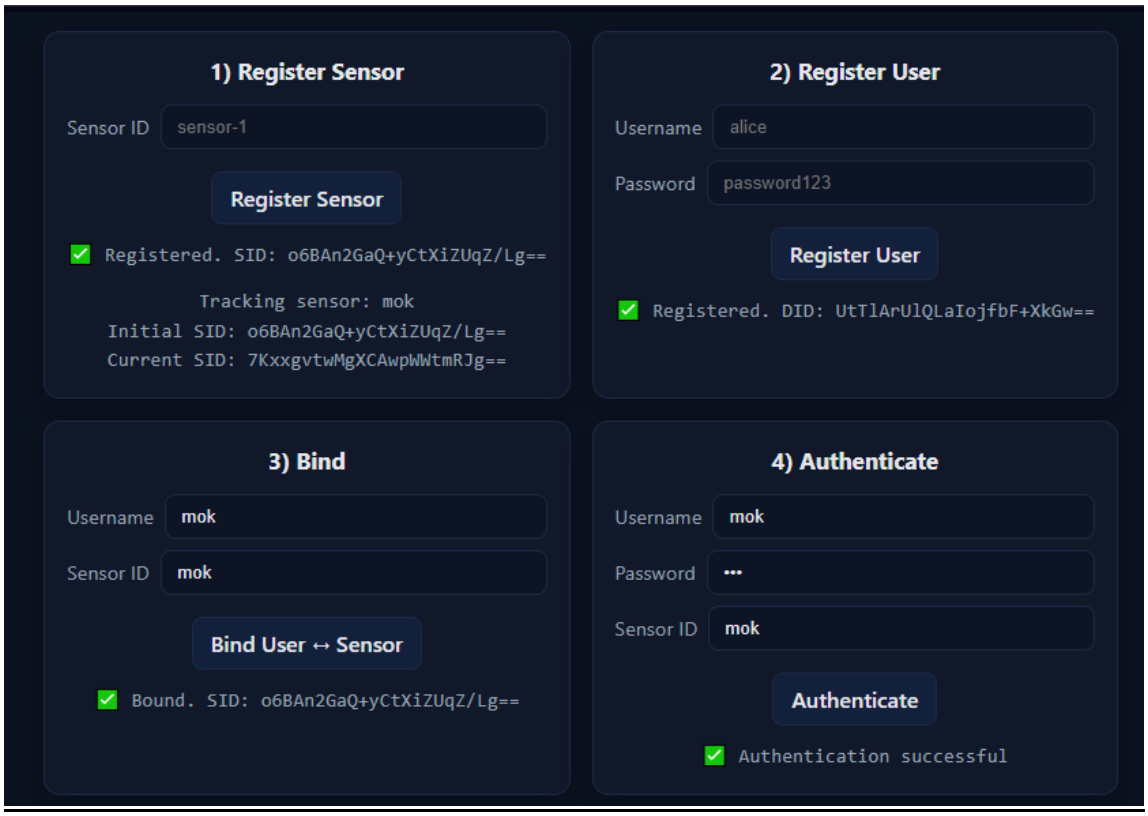
This ensures:

- MITM cannot substitute their own ephemeral key (would invalidate α)
- Replay attacks detected via fresh session_nonce
- Both parties verify they share the same ephemeral keys

7.4 Frontend Implementation (App.jsx)

The React frontend provides:

- **Sensor Registration Panel**: Register IoT sensors with unique IDs
- **User Registration Panel**: Register users with username/password
- **Binding Interface**: Associate users with their authorized sensors
- **Authentication Panel**: Test the full Curve25519 + HKDF authentication
- **Live Log Viewer**: Real-time protocol message inspection



SECURITY ANALYSIS

8.1 Forward Secrecy Analysis

Threat Model:

- Attacker records all network traffic
- Attacker later compromises gateway's long-term credentials
- Goal: Decrypt historical communications

Without Enhancement (Original Protocol):

```
SK = f(k, b_i, R, ...)  where k is stored in gateway
Compromise of k → All historical SK can be computed
```

With Enhancement (Our Solution):

```
SK = HKDF(ECDH(ephemeral_priv, ephemeral_pub), nonce, context)
Ephemeral private keys deleted after session
Compromise of long-term keys → Cannot compute historical SK
```

Bounded Forward Secrecy Window:

- Daily key K_daily protects sessions within 24-hour window
- Compromise reveals at most 24 hours of communications
- Acceptable trade-off for IoT computational constraints

8.2 MITM Attack Resistance

Attack Scenario:

```
User ──M1──► [MITM] ──M1'──► Gateway
User ◄──M4'── [MITM] ◄──M4── Gateway

MITM attempts to:
1. Replace user's ephemeral key with MITM's key in M1
2. Replace gateway's ephemeral key with MITM's key in M4
3. Establish separate session keys with each party
```

Defense Mechanism:

```
α = h(... || ephemeral_public_key)

If MITM replaces key:
  α_computed_by_gateway ≠ α_sent_by_user
  → Authentication fails
  → "MITM attack detected!"
```

Test Result:

MITM ATTACK BLOCKED!
Alpha mismatch detected at gateway
Protocol correctly rejected modified message

8.3 Computational Overhead Analysis

Operation	Time (ms)	Frequency	Daily Cost
Curve25519 KeyGen	0.05	1/day	0.05 ms
X25519 ECDH	0.15	1/day	0.15 ms
HKDF-SHA256	0.02	per session	~29 ms
Total Additional			**~29.2 ms/day**

For 1440 sessions/day (1 per minute):

- Original: ~0 ms additional crypto
- Enhanced: ~29.2 ms additional crypto (~0.02 ms per session)
- **Overhead: < 1% impact on session establishment**

CONCLUSION

11.1 Summary

This project successfully enhanced Zhou et al.'s IoMT authentication protocol by adding forward secrecy through a Hybrid Hierarchical ECDH scheme using Curve25519. Key achievements include:

1. **Security Enhancement**: Added bounded forward secrecy (24-hour windows) protecting historical communications even if long-term credentials are compromised

2. **Performance Optimization**: Selected Curve25519 for 25-30% faster computation compared to NIST P-256, with daily key amortization reducing per-session overhead to ~0.02ms
3. **MITM Resistance**: Implemented authenticated ECDHE by binding ephemeral keys to authentication hashes (α and λ), preventing key substitution attacks
4. **Practical Implementation**: Developed a complete working system with:
    - Python backend with Flask REST API
    - React frontend with real-time logging
    - Comprehensive test suite demonstrating security properties

11.2 Future Work

- **Hardware Implementation**: Port Curve25519 to embedded medical devices
- **Formal Verification**: Use ProVerif or Tamarin for formal security proofs
- **Group Authentication**: Extend protocol for multi-device medical scenarios
- **Post-Quantum Migration**: Evaluate hybrid schemes with Kyber/Dilithium

11.3 Contributions

This project demonstrates that forward secrecy can be practically added to IoMT protocols with minimal overhead, addressing a critical security gap in healthcare IoT systems while respecting the computational constraints of medical devices.

REFERENCES

[1] Y. Zhou et al., "A Secure Three-Factor Authentication Protocol for IoMT Based on PUF and Fuzzy Extractor," IEEE Internet of Things Journal, 2024.
[2] D. J. Bernstein, "Curve25519: New Diffie-Hellman Speed Records," Public Key Cryptography – PKC 2006, pp. 207-228.
[3] H. Krawczyk and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)," RFC 5869, IETF, 2010.
[4] NIST, "Recommendation for Key Derivation through Extraction-then-Expansion," SP 800-56C Rev. 2, 2020.
[5] D. Hankerson, A. Menezes, and S. Vanstone, "Guide to Elliptic Curve Cryptography," Springer, 2004.
[6] IEEE, "IEEE Standard for Local and Metropolitan Area Networks: Port-Based Network Access Control," IEEE 802.1X-2020.
[7] HIPAA, "Health Insurance Portability and Accountability Act Security Rule," 45 CFR Part 164, 2013.
[8] M. Wazid et al., "Authentication in Cloud-driven IoT-based Big Data Environment: Survey, Issues, Challenges and Future Research Directions," Journal of Systems Architecture, vol. 97, pp. 185-206, 2019.