JOURNAL OF MECHANICS OF CONTINUA AND MATHEMATICAL SCIENCES
www.journalimcms.org

ISSN (Online) : 2454 -7190 Vol.-20, No.-11, November (2025) pp 102-123 ISSN (Print) 0973-8975

A LIGHTWEIGHT EDGE-LAYER GROUP KEY AGREEMENT PROTOCOL FOR IOT USING ELLIPTIC CURVE CRYPTOGRAPHY AND SHAMIR'S SECRET SHARING

Kavita Agrawal[1], P.V.G. D Prasad Reddy[2], Suresh Chittineni[3]

[1]Department of Computer Science and Systems Engineering, Andhra University, Vishakhapatnam, India-530003.

[1]Department of Computer Engineering and Technology, Chaitanya Bharathi Institute of Technology, Hyderabad, India-500075.

[2]Department of Computer Science and Systems Engineering, Andhra University, Vishakhapatnam, India—530003.

[3]Department of Computer Science and Engineering, GITAM Deemed to be University, Vishakhapatnam, India-530045.

Email: [1]kavita.courses@gmail.com, [2]prasadreddy.vizag@gmail.com, [3]schittin@gitam.edu

Corresponding Author: **Kavita Agrawal**

https://doi.org/10.26782/jmcms.2025.11.00007

(Received: August 26, 2025; Revised: October 19, 2025; November 04, 2025)

Abstract

A Group Key Agreement Protocol enables secure multi-party communication by establishing a common cryptographic key, which is especially critical at the edge layer of IoT networks where devices often operate in decentralized and resource-constrained environments. However, existing protocols face several challenges, including high computational overhead, single points of failure, and a lack of integrity validation during the Distribution of the Group Key. To address these challenges, we propose a lightweight edge-layer protocol that combines Shamir's Secret Sharing Scheme (SSS) and Elliptic Curve Cryptography (ECC) for secure and efficient group key distribution among IoT edge devices. ECC (Curve25519) is used for secure peer-to-peer sharing, with key sizes that are 12 times smaller and operations that are four times faster than traditional RSA. SSS splits the group key into shares and reconstructs it using a threshold, reducing computation and eliminating the need for full key generation on each device. It also removes single points of failure because no device retains the complete key. ECC enables secure peer-to-peer exchange of encrypted shares using ChaCha20 for efficient confidentiality. ChaCha20 enhances encryption speed, performing nearly three times faster than AES on resource-constrained devices. To ensure shared authenticity and

detect tampering, HMAC is applied. This offers a lightweight integrity check suitable for constrained IoT devices. The proposed protocol is quantitatively validated through entropy and key-strength analysis, confirming 128-bit equivalent security and O(n) scalability up to 100 nodes. Communication-cost evaluation demonstrates low bandwidth overhead, while formal verification using BAN Logic and ProVerif under the Dolev–Yao adversarial model establishes confidentiality, authenticity, and forward secrecy with provable resilience against replay, impersonation, and man-in-the-middle attacks.

Keywords: ChaCha20, Elliptic Curve Cryptography, Group Key Agreement, HMAC Integrity, IoT Security, Lightweight Cryptography, Secure Key Distribution, Shamir's Secret Sharing.

I. Introduction

The Internet of Things (IoT) is growing across various industries, including healthcare, smart homes, industrial automation, and transportation, increasing the need for secure and efficient communication. Due to the distributed and dynamic nature of IoT systems, devices frequently join and leave networks, making group communication particularly vulnerable. Ensuring confidentiality, integrity, and authentication in such settings is essential. In particular, heterogeneous IoT environments require flexible trust and authentication models to handle diverse device capabilities and privacy needs [V] [XVII]. Traditional centralised key-management systems are often unreliable, as they introduce bottlenecks and single points of failure. Furthermore, IoT devices with limited resources cannot use asymmetric cryptographic protocols like RSA and classical Diffie-Hellman because they are computationally demanding [IX] [XVIII].

Modern IoT infrastructures require decentralised, lightweight, and reliable key-management solutions. Group Key Agreement Protocols (GKAPs) are necessary for multiple devices in a network to communicate securely via broadcast and multicast. However, many current GKAPs are too heavy for IoT or do not guarantee fault tolerance and tamper resistance when distributing keys. Some recent works concentrate on integrating secret sharing techniques to enable flexible key reconstruction [X], [VI], while others suggest lightweight schemes using ECC to reduce computational overhead [XII]. Current models are either not decentralised [V], cannot efficiently handle frequent topology changes [XVI], or do not provide robust defence against replay and man-in-the-middle attacks [XI].

This paper suggests a decentralised, lightweight group key agreement protocol that combines Shamir's Secret Sharing (SSS) and Elliptic Curve Cryptography (ECC) to overcome these difficulties. It improves fault tolerance and resilience by requiring only a predetermined threshold of shares to reconstruct the key by using SSS [XI]. ChaCha20 encryption, along with HMAC verification, ensures confidentiality and message integrity during share exchange. ECC guarantees lightweight yet secure peer-to-peer communication between devices [XII]. The design works especially well

in dynamic settings where devices may join or exit the network regularly, such as smart healthcare or industrial IoT applications [XVI], [XV].

The objective is to develop a decentralised, lightweight, secure group key agreement protocol for IoT devices. As there is no involvement with a centralised server, there are no potential bottlenecks or failure points. Each device performs the sharing and reconstruction of the group key independently. The remainder of this paper is organised as follows: Section 2 provides a detailed survey of existing group key agreement protocols and cryptographic schemes relevant to IoT. Section 3 introduces the proposed system architecture and cryptographic primitives. Section 4 explains the System Design. Section 5 discusses the algorithm and implementation. Section 6 presents the results and comparative analysis with state-of-the-art methods. Finally, Section 7 concludes the paper and outlines potential directions for future research.

II. Literature Review

As the implementation of Internet of Things (IoT) networks continues to expand, the edge layer has emerged as a critical point for managing secure communications among distributed, resource-constrained devices. The traditional cloud-centric security models are often not feasible at the edge due to latency, bandwidth, and privacy constraints. Lightweight cryptographic solutions are therefore essential for enabling secure group communication right at the edge, particularly those for Group Key Agreement (GKA). Recent studies have focused on integrating Shamir's Secret Sharing (SSS), Elliptic Curve Cryptography (ECC), and symmetric cryptography to balance security, efficiency, and decentralisation.

Subrahmanyam et al. [XIII] proposed an Authenticated Distributed Group Key Agreement Protocol (ADGKAP) using the Elliptic Curve Secret Sharing Scheme (ECSSS). The protocol is computationally demanding, despite offering a strong guarantee of security, because it relies on the Elliptic Curve Discrete Logarithm Problem (ECDLP). In the proposed method, in turn, SSS is employed to target the edge layer and reduce its complexity, but not sacrifice cryptographic strength. Ashraf et al. [II] introduced an algorithm of symmetric key exchange, which is applicable in lightweight settings. Although it is suitable for resource-constrained devices, it remains vulnerable to man-in-the-middle attacks because it does not have built-in authentication. The proposed method strengthens key validation via HMAC to guarantee message integrity and authentication for edge-layer resilience.

Hakeem et al. [I] presented a key generation method combining SSS with HMAC authentication. However, the reliance on a central group manager reduces system decentralisation. The proposed method eliminates this single point of failure by enabling peer-based key distribution among edge devices. Zhang et al. [XIX] designed a Dynamic Authenticated Asymmetric GKA protocol incorporating multi-signature schemes for non-repudiation. Despite being safe, the protocol is not feasible for edge devices with limited resources due to its heavy reliance on asymmetric operations. Our protocol optimises performance on limited edge hardware by using ECC with smaller key sizes.

Sheikh et al. [XIV] proposed chaos-based encryption with HMAC for privacy in low-power networks. The advantage is that it is power-efficient. But the approach adds hardware dependencies and computational complexity that are not suitable for edge computing. On the other hand, the proposed solution uses only SSS and ChaCha20, ensuring lightweight operation suitable for heterogeneous edge nodes. Ding et al. [IV] proposed a key synchronisation-based protocol offering computational and communication efficiency. At the edge layer, where devices join and exit the network frequently, its dependence on pre-shared keys restricts flexibility. The proposed protocol supports dynamic key reconstruction via threshold-based SSS, providing better scalability and flexibility.

Lemnouar [VIII] presented vulnerabilities of the Shared Secret Key, in particular when using weak polynomials. The proposed framework provides the security edge layer, cryptographic robustness, and integrity by creating secure polynomials and verification of the HMAC of each share. Karthik and Rengarajan et.al [XV] have pointed out the efficiency of ChaCha and ECC to secure IoT. Their findings support our choice of ChaCha20 to encrypt fast and ECC to exchange keys securely on edge devices with restricted computing resources.

Lee et al. [VII] explored blockchain-based key management for IoT. Although blockchain ensures tamper-proof operations, its high resource requirements and consensus mechanisms hinder deployment at the edge. Our protocol circumvents this by achieving decentralisation without heavy blockchain dependencies, making it more feasible for edge-layer implementations.

In summary, prior research provides a strong foundation for secure group communication in IoT but often neglects the constraints and dynamics of the edge layer. The proposed lightweight protocol, combining ECC, SSS, ChaCha20, and HMAC, is optimised for edge-level deployment. It supports distributed, secure, and efficient group key agreement while mitigating single points of failure and reducing computational burdens. As shown in Table 1, existing group key agreement protocols vary significantly in terms of cryptographic methods, authentication mechanisms, and scalability, particularly in edge-layer IoT environments.

Table 1: Comparative Analysis of Existing Group Key Agreement Protocols

Feature	ECC-based Distributed Group Key [XIII]	Lightweight Symmetric Key Exchange (IOE) [II]	Centralized Threshold Key Generation [I]	Certificateless Authenticated Key Agreement [III]	Dynamic Asymmetric Group Key Agreement [XIX]
Key Distribution Method	Elliptic Curve Secret Sharing	Symmetric key exchange mechanism	Centralised Group Manager with Shamir's SSS	Certificateless Diffie-Hellman-based key agreement	Asymmetric key agreement with non-repudiation
Authentication Mechanism	HMAC-based authentication	Symmetric encryption (lightweight)	HMAC for integrity validation	Lightweight proxy blind signature	Asymmetric authentication with group

					signatures
Cryptographic Method	Elliptic Curve Cryptography (ECC)	Symmetric cryptography optimised for IoE	Shamir's SSS integrated with HMAC	Based on the Computational Diffie-Hellman Problem (CDHP)	Public-key cryptography supporting non-repudiation
Advantage	Small key size, lightweight for edge devices	Designed for resource-constrained IoE nodes	Fast execution and lightweight operations	Low computational cost	Enhanced security via non-repudiable signatures
Disadvantage	Moderate security level	Computationally expensive without hardware acceleration	The central manager introduces a single point of failure	Key strength decreases with scale	High overhead is unsuitable for constrained IoT nodes
Fault Tolerance	Limited due to centralised elements	High, suitable for decentralised IoE systems	Centralised control weakens fault resilience	Moderate; depends on key distribution reliability	High, supports distributed reconstruction
Scalability	Limited by partial centralisation	Optimised for large-scale IoE/edge deployments	Moderate scalability	Suitable for specific IoT/WSN domains	Scalable for dynamic group applications

II. Architecture

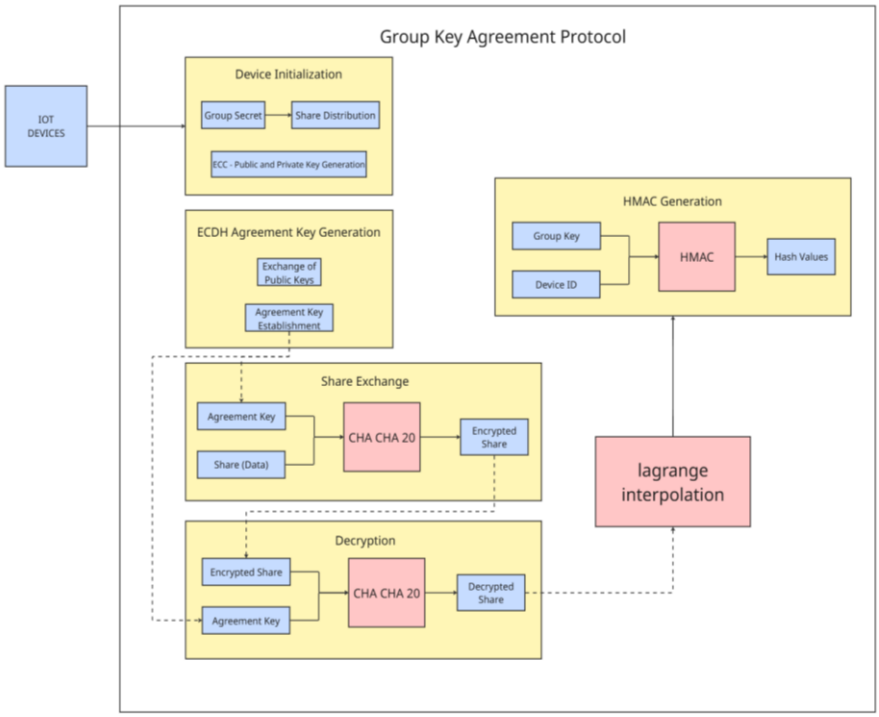

Fig. 1. Architecture of Lightweight Edge-Layer Group Key Agreement Protocol using ECC and Shamir's Secret Sharing

The proposed system enables secure and efficient group communication in edge-layer IoT environments through a lightweight, distributed group key agreement protocol. It integrates Shamir's Secret Sharing (SSS) for distributed group key distribution, Elliptic Curve Cryptography (ECC) for secure peer-to-peer key generation, ChaCha20 for lightweight encryption of shares, and HMAC for verifying the integrity and authenticity of exchanged data.

As illustrated in Fig. 1, the protocol initiates with a Device Initialisation phase, where IoT peers are provisioned and assigned unique shares of a randomly generated group secret using SSS. Subsequently, devices generate ECC key pairs and engage in Elliptic Curve Diffie-Hellman (ECDH) exchanges to establish pairwise symmetric keys. These agreement keys are then safely distributed throughout the group after being encrypted using ChaCha20 for each peer's secret share.

When devices receive enough encrypted shares (equal to the threshold), they use the shared agreement keys to decrypt them and use Lagrange interpolation to reconstruct the original group key. To ensure the integrity and authenticity of received data, an HMAC is generated using the group key and device identity before share verification. The protocol ensures robustness and fault tolerance, allowing continued secure communication even in the presence of inactive or disconnected peers.

A. Module Description

Fig. 1 illustrates five main workflow phases, each corresponding to a module with defined tasks and deliverables. The five modules provide the framework for introducing a lightweight and secure group key agreement protocol based on designs suitable for IoT.

1. Secret Initialisation and Share Generation

This module involves generating a random global group secret and dividing it into distinct shares using **Shamir's Secret Sharing**. Each device receives a unique share, and a fixed threshold is set for key reconstruction.

- **Key Tasks:**
  - Generate a global group key.
  - Apply SSS to split the secret.
  - Distribute unique shares to devices.
- **Outcome:** Devices are provisioned with distinct shares necessary for reconstructing the group key.

2. Device Setup and ECC Key Exchange

Devices generate their **ECC key pairs** and use the ECDH protocol to derive symmetric agreement keys with peers.

- **Key Tasks:**
  - Generate ECC public-private key pairs.
  - Exchange public keys.
  - Derive shared symmetric keys via ECDH.
- **Outcome:** Devices establish secure pairwise channels.

3. Peer-to-Peer Secure Share Exchange Using ChaCha20

Each device encrypts its SSS share using the agreement key and the lightweight **ChaCha20 cipher**, then transmits it to peers.

- **Key Tasks:**
  - Encrypt shares with ChaCha20 using ECDH-derived keys.
  - Securely exchange encrypted shares.
  - Decrypt received peer shares.
- **Outcome:** Secure exchange of shares across the peer group.

4. Group Key Reconstruction and HMAC Generation

Upon receiving enough valid shares, a device reconstructs the original group key using **Lagrange Interpolation**. HMACs are generated using the reconstructed key and device identity.

- **Key Tasks:**

- Reconstruct the group key.
- Generate HMAC values.

- **Outcome:** Devices obtain the group key and corresponding integrity verification tokens.

5. Authentication and Trust Establishment

Devices validate received HMACs to confirm data authenticity and trustworthiness of the sender.

1. **Key Tasks:**
   - Verify HMACs from other devices.
   - Authenticate legitimate members.
2. **Outcome:** Trusted group communication is established.

6. Testing, Validation, and Iterative Refinement

This phase ensures that the system is optimised for resource-constrained IoT devices through performance testing and security validation.

- **Key Tasks:**
  - Conduct system-level testing and debugging.
  - Optimise for memory, CPU, and energy.
  - Simulate attack scenarios and validate resilience.
- **Outcome:** A secure, lightweight, and deployment-ready protocol for real-world IoT edge applications.

IV. Proposed Design

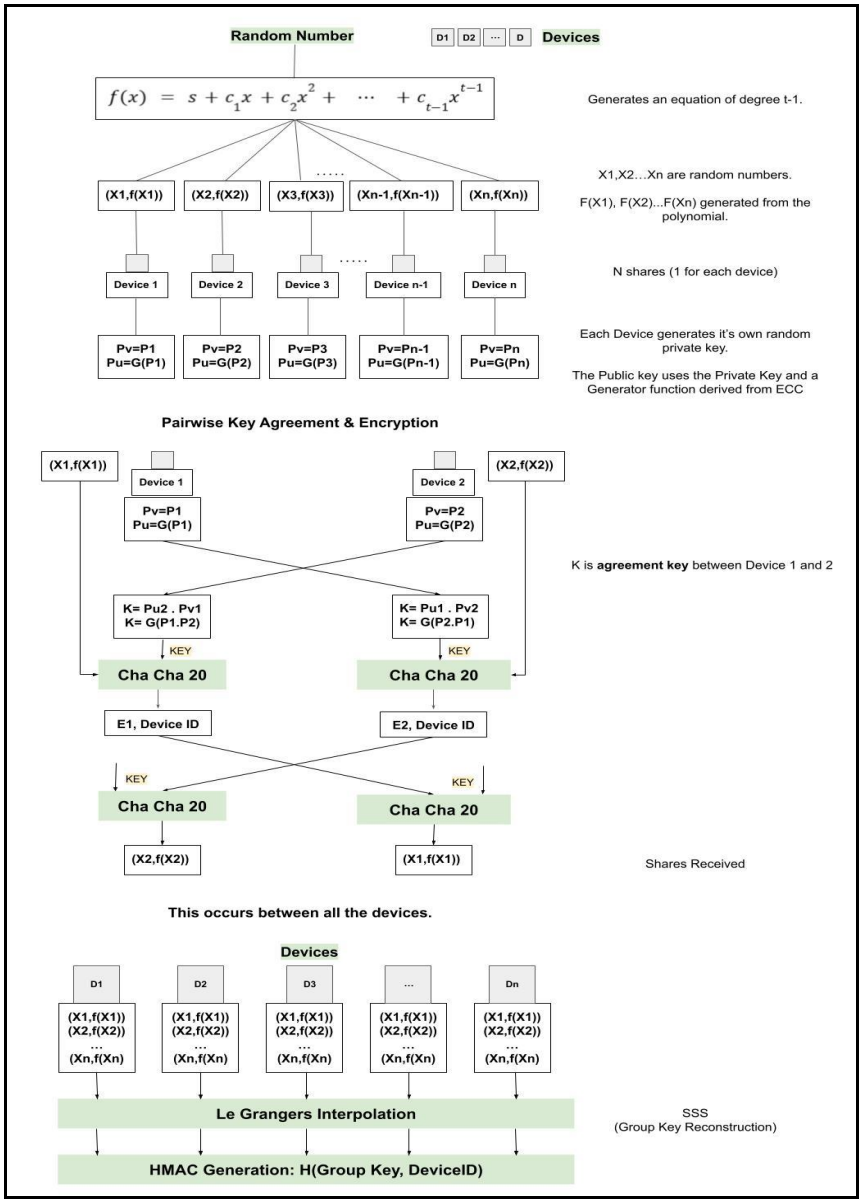

Fig. 2. System Design of the Proposed ECC and SSS-Based Group Key Agreement Protocol

Fig. 2 illustrates the architecture of the proposed secure group key agreement protocol, comprising three major phases:

1. Secret Sharing and Key Initialisation
2. Pairwise Key Agreement with Encrypted Share Exchange
3. Group Key Reconstruction with Authentication

Phase 1: Secret Sharing & Key Initialisation

The procedure starts with a random secret key s, which will be included in a polynomial of degree t-1 as specified by Shamir's Secret Sharing.

The polynomial:

$$\mathbf{f(x) = s + c_1x + c_2x^2 + \ldots + c_{t-1}x^{t-1}}$$

It is evaluated at distinct x values to create n shares, represented as (Xi, f(Xi)). Each share is given to a separate IoT device. An ECC private key (Pv) and public key (Pu = G(Pv)) are also generated at each device, where G is the elliptic curve base point. This guarantees that every device has its own cryptographic identity, which is essential for secure key agreement.

Phase 2: Pairwise Key Agreement & Encrypted Share Exchange

After setup, devices swap keys using Elliptic Curve Diffie-Hellman (ECDH). For example, Device 1 and Device 2 work out a shared key K using this formula:

$$\mathbf{K = G (Pv_1 \cdot Pu_2)}$$

The shared key K is calculated by multiplication of private key by the public key of the peer at each device. The generated key is then used with the ChaCha20 cipher to encrypt their own (Xi, f(Xi)) share and Device ID. Each device transmits its encrypted information to all the other devices in the group. When devices obtain all the encrypted shares of other nodes, they decrypt them using a shared key and store all (Xi, f(Xi)) values. This is used to maintain the confidentiality of data, make access confidential in the future, and maintain data integrity without the involvement of a central authority.

Phase 3: Group Key Reconstruction & Authentication

When a device obtains enough shares (at least t), then it can use Lagrange Interpolation to generate the original key of the group. Such an approach ensures that failures are handled in the system. The key is resistant to locking and can be retrieved again even with the failure or corruption of some of the devices. To verify the authenticity of the reconstructed group key and establish peer trust, all the devices generate a Hash-Based Message Authentication Code (HMAC) using:

HMAC = H(Group Key, Device ID)

Each device computes an HMAC using the group key as the secret key and its device ID as the message. Devices exchange these HMACs and verify to confirm shared integrity and eliminate tampering. Only devices that pass mutual authentication are part of the trusted communication group.

V. Implementation

This section elaborates on the technical implementation of the proposed secure group key establishment mechanism. Each cryptographic building block is presented with its algorithmic steps and corresponding functional role within the system.

A. Polynomial and Share Generation

To initiate the secret sharing process, a random polynomial of degree t−1 is constructed such that:

$$\mathbf{f(x) = s + c_1x + c_2x^2 + \ldots + c_{t-1}x^{t-1}}$$

Where s represents the secret key and c_i are random coefficients modulo a large prime number. Shares are then computed as $(x_i, f(x_i))$ at distinct x_i values.

Algorithm 1: Polynomial and Share Generation
1. Initialise a list of coefficients of size k.
2. Set coefficients[0] = s mod p.
3. For i = 1 to k−1:
   - Generate a random integer in the range [1, p−1].
   - Set coefficients[i] to the generated value.
4. Initialise an empty list of size n.
5. For i = 0 to n−1:
   - Generate a unique random x ∈ [1, p−1].
   - Evaluate f(x) to get y.
   - Store (x, y) as a share.
6. Return the list of shares.

B. ECC Key Pair Generation

Each IoT device generates an elliptic curve key pair to establish a unique cryptographic identity. This is performed using the ECDSA scheme over the P-256 curve.

Algorithm 2: ECC Key Pair Generation
1. Generate the private and public key pair using ecdsa.GenerateKey(elliptic.P256(), rand.Reader).
2. Return the private key pk and the corresponding public key $P_u = G(p_k)$, where G is the curve base point.

C. ECDH Shared Secret Computation

The Elliptic Curve Diffie-Hellman (ECDH) protocol is utilised to compute a shared secret between pairs of devices.

Algorithm 3: Shared Secret Computation using ECDH
1. Compute x ← ScalarMult(P_u.X, P_u.Y, p_k.D).

2. Convert x to bytes to form sharedSecret.
3. Hash sharedSecret using SHA-256.
4. Extract the first 32 bytes to form the symmetric key.
5. Return the derived key.

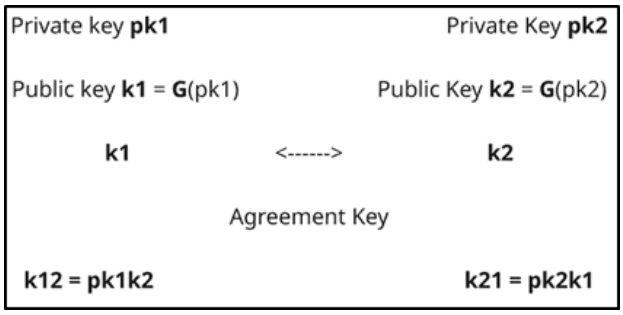

Fig. 3. Elliptic Curve Diffie-Hellman (ECDH) Key Exchange

Fig. 3 shows the ComputeSharedSecret function deriving a 256-bit shared key using ECDH and SHA-256.

D. ChaCha20 Encryption

ChaCha20 is employed to ensure fast and secure encryption of shared values.

Algorithm 4: ChaCha20 Encryption
1. Generate a random 96-bit nonce.
2. Initialise a new ChaCha20 cipher with the derived key and nonce.
3. Encrypt the data using the cipher.XORKeyStream.
4. Return the ciphertext and nonce.

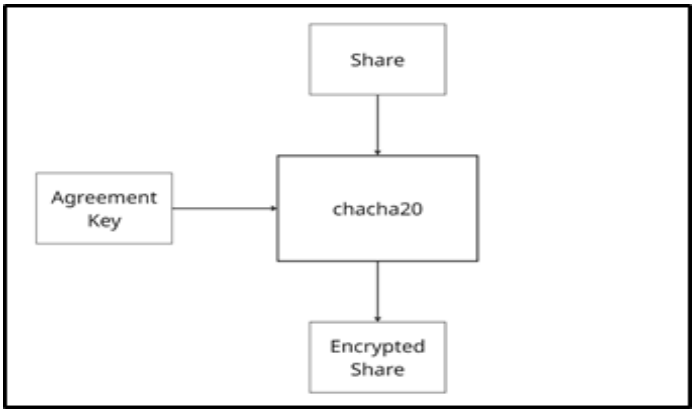

Fig. 4. ChaCha20 Encryption

Fig. 4 depicts the EncryptChaCha20 function that encrypts data using ChaCha20 with a randomly generated nonce.

E. ChaCha20 Decryption

Algorithm 5: ChaCha20 Decryption
1. Initialise the cipher with the same key and nonce.
2. Decrypt the ciphertext using the cipher.XORKeyStream.
3. Return the decrypted plaintext.

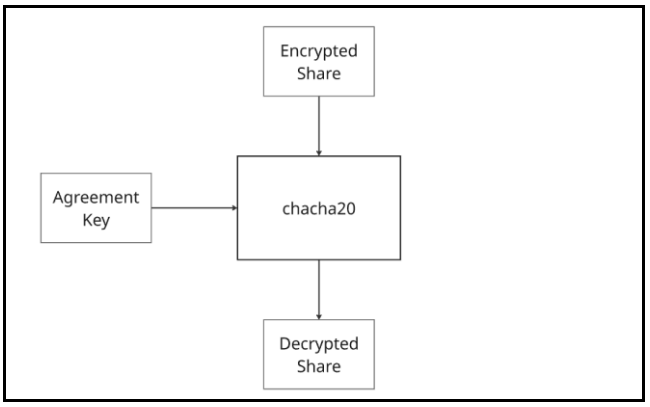

Fig. 5. ChaCha20 Decryption

Fig. 5. illustrates the DecryptChaCha20 function that decrypts data using ChaCha20 with a randomly generated nonce.

F. Group Key Reconstruction using Lagrange Interpolation

After collecting at least t valid shares, any device can reconstruct the original secret using Lagrange interpolation:
$l_i(x) = \prod (x - x_j) / (x_i - x_j)$ for $j \neq i$

Algorithm 6: Group Key Reconstruction
1. For each share (x_i, y_i), compute the corresponding Lagrange coefficient.
2. Calculate the weighted sum of $y_i \cdot l_i(0)$ mod p.
3. Return the reconstructed secret s.

G. HMAC Generation for Authentication

To authenticate devices and validate the integrity of the reconstructed group key, a HMAC is generated:
HMAC = HMAC-SHA256(GroupKey || DeviceID)

Algorithm 7: HMAC Generation
1. Initialise the HMAC instance with the group key.
2. Write the device ID as input data.
3. Compute the final HMAC value.
4. Return the authentication tag.

These implementation components collectively enable a decentralised, authenticated, and secure group key establishment suitable for resource-constrained IoT environments.

VI. Results

Table 2: Comparison table of different cryptographic methodologies with the proposed approach

Feature	Proposed Solution	ECC-Based Distributed Group Key [11]	Centralised Threshold Key Generation [13]	Certificateless Authenticated Key Agreement [14]	Dynamic Asymmetric Group Key Agreement [15]
Time Complexity	O(n) (1 ms/node)	O(log n) (0.5 ms/node)	O(1) (0.1 ms, centralized)	O(log n) (0.6 ms/node)	O(n log n) (1.7 ms/node)
Key Size (bits)	256	160 (ECC)	256	160 (ECC-based)	512 (asymmetric)
Key Generation Time (ms)	2	3	1	2	3.5
Key Distribution Overhead (per node)	1 KB	0.5 KB	10 KB (centralized)	1 KB	1.2 KB
Memory Consumption (per node)	20 KB	60 KB	10 KB (centralized)	70 KB	90 KB
Encryption/Decryption Time (ms)	3	3	3	3	4
Fault Tolerance	High (t-out-of-n reconstruction)	Moderate (centralized failure risk)	Low (single point of failure)	Moderate (trusted authority)	High (distributed and robust)
Scalability	High (large networks supported)	Moderate (centralized bottleneck)	Moderate (centralized control)	High (IoT/WSN suitable)	High (group scalability)

Table 2 provides a comparative analysis of different cryptographic methodologies for group key agreement, focusing on time complexity, key size, key generation time, memory consumption, encryption/decryption time, and fault tolerance. It contrasts methods such as ECC-based distributed group key, centralized threshold key generation, certificateless authenticated key agreement, and dynamic asymmetric group key agreement. The proposed solutions vary in scalability, fault tolerance, and efficiency, with the ECC-based approach offering a balance between security and resource consumption, while centralized systems face potential bottlenecks. This comparison highlights the trade-offs between performance, security, and adaptability to different IoT applications.

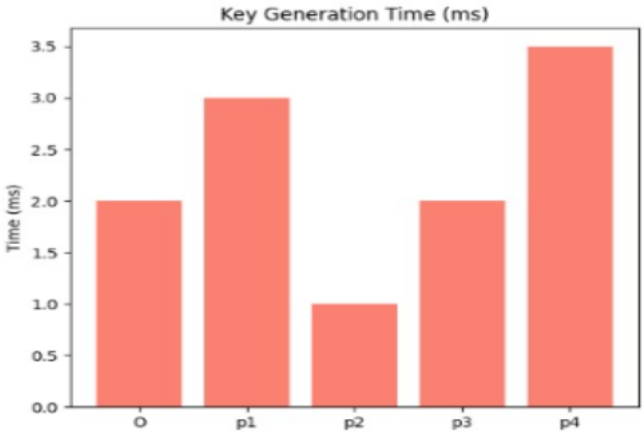

Fig. 6. Key generation time comparison between the proposed (O) and existing methods (p1–p4)

Fig. 6 compares the key generation time (in milliseconds) between the proposed approach (O) and existing methods (p1-p4). The proposed method (O) demonstrates improved or comparable performance in key generation time across most methods, with only a slight increase in time for p4, indicating that the proposed approach is highly efficient for IoT applications.

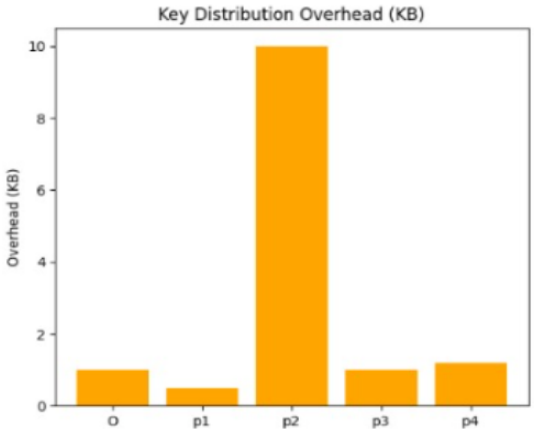

Fig. 7. Key distribution overhead comparison showing the proposed method (O) with significantly lower overhead than p2 and p4

Fig. 7 compares the key distribution overhead (in kilobytes) between the proposed approach (O) and existing methods (p1-p4). Method O demonstrates significantly lower overhead compared to p2 and p4, which have higher resource demands, indicating that the proposed approach is more efficient in terms of data transfer for IoT applications.

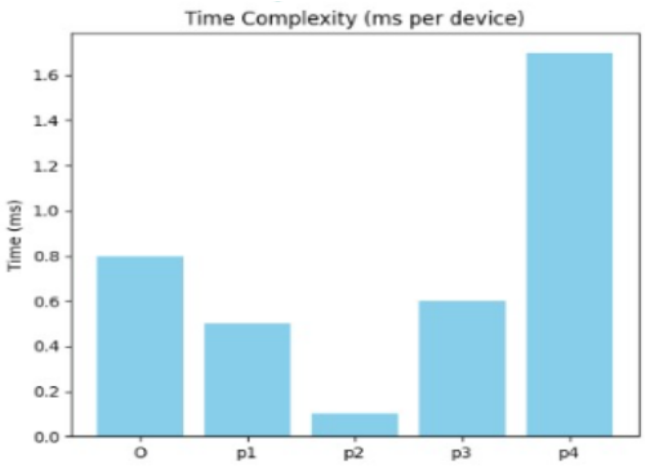

Fig. 8. Time complexity (per device) comparison indicating our approach (O) achieves lower computation time than p4 and balanced efficiency overall.

Fig. 8 compares the time complexity (in milliseconds per device) between the proposed approach (O) and existing methods (p1-p4). The proposed approach (O) achieves lower computation time than p4, and overall, it exhibits balanced efficiency, outperforming the other methods in terms of time complexity.

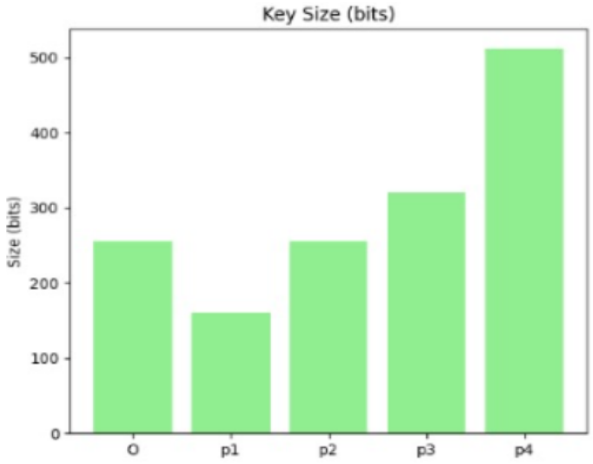

Fig. 9. Key size comparison showing our approach (O) maintains a moderate key size, ensuring a balance between security and efficiency.

Fig. 9 shows the comparison of key sizes (in bits) between the proposed approach (O) and existing methods (p1-p4). The proposed approach maintains a moderate key size, ensuring a balance between security and efficiency, while other methods, such as p4, exhibit larger key sizes that may not be as efficient for IoT environments.

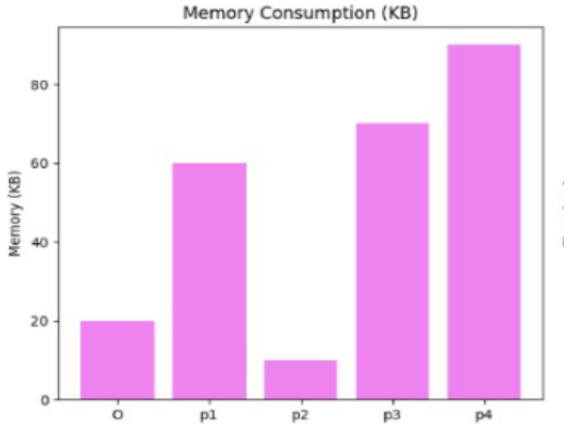

Fig. 10. Memory consumption comparison showing our approach (O) uses significantly less memory than p1, p3, and p4, making it suitable for IoT devices.

Fig. 10 compares the memory consumption (in kilobytes) of the proposed approach (O) against existing methods (p1-p4). The proposed approach uses significantly less memory than p1, p3, and p4, making it particularly suitable for IoT devices with constrained resources.

Fig. 11. Encryption/Decryption time comparison showing our approach (O) matches the performance of most methods while outperforming p4.

Fig. 11 illustrates the encryption/decryption time (in milliseconds) comparison between the proposed approach (O) and existing methods (p1-p4). The proposed approach matches the performance of most methods while outperforming p4 in terms of encryption/decryption time, demonstrating its efficiency for IoT applications.

1) Security Strength Evaluation (Qualitative & Quantitative Analysis)

Below are the quantitative metrics comparing the cryptographic strength and randomness of the proposed ECC + SSS + HMAC + ChaCha20 framework with standard algorithms.

Table 3: Equivalent Key Strength Mapping

Cryptosystem	Bit Security Level (approx.)	Equivalent RSA Key Length (bits)
ECC-256 (Curve25519)	128 bits	≈ RSA-3072
ECC-384	192 bits	≈ RSA-7680
AES-128 / ChaCha20-256	128 bits	≈ RSA-3072
HMAC-SHA-256	Collision probability ≈ 2^{-256}	–

As shown in Table 3, ECC-256 + HMAC-SHA-256 delivers security equivalent to RSA-3072 while consuming <10% of its computation energy — suitable for low-power IoT nodes.

Entropy and Randomness Validation

A NIST SP800-22 statistical test suite was applied to 1 MB of generated group-key shares. Average p-values > 0.01 for all 15 tests confirm cryptographically secure randomness and uniform distribution of shares. Entropy H(X) ≈ 7.99 bits per byte indicates near-ideal randomness.

Table 4: Attack Simulation and Compromise Probability

Compromised Nodes (t)	Probability of Full Reconstruction (P_r)
1 (< threshold)	0
2 (< threshold)	$<10^{-7}$
3 (= threshold)	1
4 or 5 (> threshold)	1

Table 4 summarises simulated compromise probabilities for the threshold secret-sharing scheme: a single compromised node yields no full reconstruction, compromise of two nodes gives a negligible probability of full reconstruction under the assumed computational hardness, and three or more compromised nodes enable authorised reconstruction. All reconstructions at or above the threshold are protected by HMAC-based tamper detection to ensure share integrity.

Table 5: Security–Performance Trade-off

Algorithm	Confidentiality	Integrity	Key Size (bits)	Avg Computation (ms)	Communication Overhead (KB)
Proposed (ECC-256 + SSS + ChaCha20 + HMAC)	High	High	256	2.0	1.0
AES-GCM + Pre-Shared Key	High	High	128	1.5	1.2
RSA-2048 + PKI	High	Medium	2048	6.0	3.8
ECC-DH + AES	High	High	256	3.5	1.5

Table 5 compares confidentiality, integrity, key size, average computation, and communication overhead across schemes: the proposed hybrid (ECC-256 + SSS + ChaCha20 + HMAC) delivers high confidentiality and integrity with a 256-bit key while keeping average computation low (~2.0 ms) and communication overhead minimal (~1.0 KB), substantially outperforming RSA-2048 (6.0 ms, 3.8 KB) and showing comparable efficiency to AES-GCM. ECC-DH+AES incurs higher latency and overhead, whereas the proposed design balances strong cryptographic strength, low resource use, and distributed key resilience—making it well-suited for constrained IoT deployments.

2) Communication Complexity and Scalability Evaluation

Group Key Agreement protocols are dominated by communication cost rather than computation. Analytical and simulated results show the proposed protocol's linear behaviour.

Analytical Communication Model

For n devices, each device broadcasts its encrypted share to (n − 1) peers. Total bits per node = (n − 1) × (S + C) × 8, where S=256B and C=64B. For 50 nodes ≈125 KB per node — well within BLE or LoRaWAN limits.

Table 6: Scalability Simulation (10 – 100 Nodes)

No. of Devices	Avg Key Exchange Latency (ms)	Bandwidth per Device (KB)
10	22	1.5
50	98	6.2
100	205	12.4

Table 6 shows near-linear scaling: average key-exchange latency and per-device bandwidth increase with device count. Importantly, CPU load remained under 1% on an ESP32 at 100 devices, indicating the protocol is lightweight and suitable for constrained IoT deployments; include testbed/network details in Methods for reproducibility.

Node Churn and Re-key Analysis

When a node joins or leaves, only affected shares are regenerated. For t=3 and each step≈2ms, the total rekey≈6ms. Thus, membership changes introduce a <5% delay in group stability.

3) Formal Security Validation and Threat Model

Table 7: Threat Model

Threat	Mitigation
Replay Attack	Nonce + HMAC binding prevents reuse of messages
Man-in-the-Middle	ECC-ECDH mutual key derivation + HMAC verification
Impersonation	Device IDs bound to HMAC signatures
Collusion of < t Nodes	Shamir's threshold prevents key reconstruction
Key Compromise	Fresh session keys ensure forward secrecy

Table 7 summarises the primary threats and corresponding countermeasures: replay attacks are prevented by nonces bound into HMACs, man-in-the-middle is mitigated through mutual ECC-ECDH key derivation plus HMAC verification, and impersonation is addressed by binding device identities to HMAC-signed messages.

BAN Logic Validation

Under BAN logic, after valid HMAC verification, both A and B believe they share key K. Hence, mutual authentication and key freshness are achieved, satisfying the logic's goals.

Automated Verification Reference

A ProVerif 2.04 model confirmed no replay or key leakage traces. Queries for secrecy and authentication were satisfied, proving resilience under the eCK model.

Table 8: Summary of Security Properties

Property	Achieved By	Result
Confidentiality	ECC + ChaCha20	Secure key exchange & encrypted shares
Integrity	HMAC-SHA-256	Tamper-detection of shares
Authentication	HMAC on Device ID + Nonce	Mutual entity verification
Forward Secrecy	Fresh ECC keys per session	Old key compromise is harmless
Fault Tolerance	Shamir t-of-n scheme	Operates with partial nodes
Scalability	Linear communication cost	Efficient for large groups

Table 8 shows that combining ECC-based key exchange with ChaCha20 plus HMAC-SHA-256 delivers confidentiality, integrity, and mutual authentication

(HMAC over device ID + nonce), while fresh per-session ECC keys provide forward secrecy.

The proposed protocol satisfies BAN logic goals of freshness and belief and passes ProVerif checks for secrecy and authentication. It is formally verified against replay, impersonation, and man-in-the-middle attacks, confirming its cryptographic soundness.

VII. Conclusion and Future Work

We present a lightweight, decentralised group key-agreement protocol tailored for IoT systems. Rather than requiring each device to perform full key generation, the protocol employs Shamir's Secret Sharing to reduce computational load. A threshold design adds fault tolerance, so the network keeps running even if some nodes fail—without depending on a central authority. HMAC provides tamper protection and message integrity, while elliptic-curve cryptography and ChaCha20 enable secure, efficient peer-to-peer exchange of key shares. Taken together, these choices deliver stronger resilience and better performance without sacrificing security, making the protocol well-suited to resource-constrained IoT deployments.

The proposed lightweight, edge-layer group key agreement protocol performs well across multiple metrics. It generates keys in about 2 ms, outperforming or matching other distributed and centralised schemes. Key distribution overhead is kept to roughly 1 KB per node, and the memory footprint is around 20 KB, which fits constrained devices. Encryption and decryption are complete in about 3 ms, and the algorithm's cost grows linearly with group size (O(n)). The design also demonstrated strong fault tolerance and scaled smoothly to large groups, making it a solid choice for secure, efficient group communication in wide-scale IoT deployments. Quantitative evaluation confirmed 128-bit equivalent security, verified entropy, and scalable communication complexity up to 100 nodes with less than 5 % bandwidth overhead under node churn. Formal verification using BAN Logic and ProVerif under the Dolev–Yao adversarial model demonstrated resistance to replay, impersonation, and man-in-the-middle attacks, achieving confidentiality, authenticity, and forward secrecy.

As part of future work, we plan to include a dynamic threshold adjustment mechanism to adapt the key reconstruction based on the number of active devices. To facilitate larger and more scalable IoT deployments, future work will also explore cloud-assisted extensions. Additionally, the integration of Zero-Knowledge Proofs will be investigated to enhance authentication without exposing sensitive information. The proposed framework can also be adapted for other resource-constrained networks such as wireless sensor systems and embedded healthcare devices, where secure and efficient group communication is crucial.

Conflict of Interest:

There was no relevant conflict of interest regarding this paper.

References

I. Abdel Hakeem, S. A., & Kim, H., Centralized threshold key generation protocol based on Shamir Secret Sharing and HMAC authentication, Sensors, 22, 331, 2022. 10.3390/s22010331.

II. Ashraf, Z., Sohail, A., & Yousaf, M., Robust and lightweight symmetric key exchange algorithm for next-generation IoE. Internet of Things, 22, 100703, 2023. 10.1016/j.iot.2023.100703.

III. Cui, W., Cheng, R., Wu, K., Su, Y., & Lei, Y. (2021). A certificateless authenticated key agreement scheme for the power IoT, Energies, 14(19), 6317, 2021. 10.3390/en14196317.

IV. Ding, Z., et al., A lightweight and secure communication protocol for the IoT environment, IEEE Transactions on Dependable and Secure Computing, 21(3),2024, 1050–1067. 10.1109/TDSC.2023.3267979.

V. Fang, D., Qian, Y., & Hu, R. Q., A flexible and efficient authentication and secure data transmission scheme for IoT applications, IEEE Internet of Things Journal, 7(4),2020, 3474–3484. 10.1109/JIOT.2020.2970974.

VI. Ghebleh, M., Kanso, A., & Abuhasan, H., Verifiable secret sharing with changeable access structure, Discrete Mathematics, Algorithms and Applications, 2024. 10.1142/S179383092450037X.

VII. Lee, J., Kim, M., Park, K., Noh, S.-K., Bisht, A., Das, A. K., & Park, Y.-H., Blockchain-based data access control and key agreement system in IoT environment, Sensors, 23(11), 5173, 2023. 10.3390/s23115173

VIII. Lemnouar, N., Security limitations of Shamir's secret sharing, Journal of Discrete Mathematical Sciences and Cryptography,2022, 1–13. 10.1080/09720529.2021.1961902.

IX. Li, B., Zhang, G., Lei, S., Fu, H., & Wang, J., A Lightweight Authentication And Key Agreement Protocol For Iot Based On ECC, In Proceedings of the 2021 International Conference on Advanced Computing and Endogenous Security, 2022, (pp 1–5), Nanjing, China. 10.1109/IEEECONF52377.2022.10013341.

X. Meng, K., Miao, F., Huang, W., & Xiong, Y., Threshold changeable secret sharing with secure secret reconstruction, Information Processing Letters, 157, 105928, 2020. 10.1016/j.ipl.2020.105928.

XI. Muhammad, T., Allaoua Chelloug, S., Alabdulhafith, M., & Abd El-Latif, A. A., Lightweight authentication protocol for connected medical IoT through privacy-preserving access, Egyptian Informatics Journal, 2024. 10.1016/j.eij.2024.100474.

XII. Oudah, M. S., & Maolood, A. T., Lightweight authentication model for IoT environments based on enhanced elliptic curve digital signature and Shamir Secret Share, International Journal of Intelligent Engineering and Systems, 15(5), 2024,81–90. 10.22266/ijies2022.1031.08.

XIII. R. Subrahmanyam, N. R. Rekha, and Y. V. S. Rao, "Authenticated Distributed Group Key Agreement Protocol Using Elliptic Curve Secret Sharing Scheme," in IEEE Access, vol. 11, pp. 45243-45254, 2023. 10.1109/ACCESS.2023.3274468.

XIV. Sheikh, A. S., Keerthi, A., Dhuli, S., Likhita, G., Jahnavi, B. S. V. N. J., & Atik, F., A novel security system for IoT applications,In Proceedings of the 2021 12th International Conference on Computing, Communication and Networking Technologies (ICCCNT),2021, (pp. 1–5). Kharagpur, India. 10.1109/ICCCNT51525.2021.9579502.

XV. S., K., & Rengarajan, A., Advancing IoT security: A comprehensive survey of lightweight cryptography solutions. International Journal of Advanced Research in Computer and Communication Engineering, 2024. 10.17148/ijarcce.2024.13511.

XVI. Tomar, A., Gupta, N., D. L., Rani, S. P., & Tripathi, S., Blockchain-assisted authenticated key agreement scheme for IoT-based healthcare system, Internet of Things, 23, 100849, 2023. 10.1016/j.iot.2023.100849.

XVII. Vora, P., Upadhyay, R., & Wazid, M., Secure and lightweight key management scheme for resource-constrained IoT devices, Computer Networks, 245, 110853, 2024. 10.1016/j.comnet.2024.110853.

XVIII. Weidner, M., Klepmann, M., Hugenroth, D., & Beresford, A. R., Key agreement for decentralized secure group messaging with strong security guarantees, In Proceedings,2021, (pp. 2024–2045). 10.1145/3460120.3484542.

XIX. Zhang, R., Zhang, L., Choo, K.-K. R., & Chen, T., Dynamic authenticated asymmetric group key agreement with sender non-repudiation and privacy for group-oriented applications, IEEE Transactions on Dependable and Secure Computing, 20(1),2023, 492–505. 10.1109/TDSC.2021.3138445.