Ankita Ban, 2026, 14:6
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

International Journal of Science, Engineering and Technology
An Open Access Journal

Chatap: Decentralized Offline Messaging over Bluetooth Low Energy—Prototype, Proposed Mesh Relay and Security Design, and Evaluation Plan

Ankita Ban, Akash Ghutukade, Shital Kadam, Saniya Nagbhidkar, Prof. Prachi Tamhan
Dept. of Information Technology International Institute of Information Technology (I^2IT)
Hinjawadi, Pune, India

Abstract- Most chat applications depend on the internet and a central server, so they stop working when the network fails, in remote places, or after a disaster. This paper describes Chatap, an Android application written in Kotlin that lets nearby phones exchange text messages over Bluetooth Low Energy (BLE) with no internet, no Wi-Fi access point and no server. The current prototype discovers other phones running Chatap, connects to them automatically and supports real-time one-to-one chat through a simple interface. Around this working core, the paper specifies as proposed work a four-layer architecture, a local data model for the Room database, a managed-flooding relay protocol that uses message identifiers, a hop limit (TTL) and duplicate suppression, acknowledgements with retry, and an end-to-end security design based on X25519 key agreement and AES-256-GCM. We add a packet-size analysis, an upper bound on relay traffic, and a measurement plan for discovery time, delivery ratio, latency, multi-hop delivery, duplicate rate, battery drain and encryption overhead. No quantitative results have been collected yet and none are reported. Multi-hop relay, acknowledgements and encryption are not yet implemented, and we say so throughout. The contribution of this paper is a reproducible system design and evaluation protocol, not a demonstrated performance result.

Keywords- Bluetooth Low Energy, offline messaging, mesh networking, Android, decentralized communication, managed flooding, end-to-end encryption, disaster communication.

I. INTRODUCTION

People now expect to be connected all the time, yet that expectation rests on infrastructure. Applications such as WhatsApp and Telegram need mobile data or Wi-Fi, and their messages pass through central servers. After a hurricane, an earthquake or a large outage, that infrastructure may be unavailable exactly when people most need to reach each other; the resilience literature treats such failures as having significant economic and societal impact [1]. A central server also holds users' metadata in one place and is a single point of failure and control.

Almost every smartphone sold today supports BLE, and BLE works without a mobile network or a Wi-Fi access point [2]. Its weakness is range, which in practice is a few tens of metres. If other phones pass a message along, it can travel further than a single link allows, which is the basic idea of a mesh. The open-source BitChat project has shown that Bluetooth mesh chat can run on ordinary phones [3], [4].

Chatap is our attempt to build such a system as a final-year project. We want to be clear about what it is not. It introduces no new routing algorithm and makes no claim to outperform existing mesh messengers. What it offers is a small Android application that already works for one-to-one chat over BLE, together with a carefully specified design for the parts that are not yet built: multi-hop relay, acknowledgements, local storage and end-to-end encryption. Those parts are marked "proposed" wherever they appear.

1. Research Questions

RQ1. How can an Android application find nearby phones and exchange text messages in real time over BLE with no internet and no server?
RQ2. How can messages be relayed across several phones and kept confidential, using a design simple enough to implement and test on ordinary phones?

© 2026 Ankita Ban. his is an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly credited.

RQ3. What functional evidence exists today, and which measurements are needed before any claim about reliability, latency, range or battery use can be made?

2. Contributions

The paper makes four contributions, all about system design rather than algorithms. First, it describes a working Android prototype, written in Kotlin, that discovers nearby BLE devices, connects automatically and supports real-time one-to-one offline chat. Second, it proposes a layered architecture, a local data model and a managed-flooding relay protocol with TTL and duplicate suppression, together with an analytic bound on relay traffic. Third, it proposes a security design using X25519 key agreement and AES-256-GCM, mapped against a short list of threats. Fourth, it gives a measurement plan, with analysis rules, for the evaluation that has not yet been carried out.

The rest of the paper is organised as follows. Section II reviews related work. Section III gives the design. Section IV describes the implementation and its status. Section V reports what has and has not been verified and sets out the evaluation plan. Sections VI and VII discuss the design and its limits, and Section VIII concludes.

II. RELATED WORK

Mauthe et al. review principles and best practices for disaster-resilient communication networks, and describe how disasters such as hurricanes and earthquakes can disable the networks people rely on [1]. Chatap addresses a small part of that problem: the last hop between people's own devices when the infrastructure is gone.

Phone-to-phone communication in disasters has been explored before. The Serval Mesh platform was designed for resilient communication in disaster and crisis settings, letting phones talk without towers [5]. Its early Android versions relied on Wi-Fi ad hoc operation, which has been reported to need modified devices. Chatap's use of BLE avoids that requirement, but pays for it with lower throughput and shorter range.[2].

Table I lists the Bluetooth-mesh messaging systems we examined most closely. They are recent, and several are repositories or news reports that have not been peer reviewed. We therefore treat what they say about their technology as description, not as proven results.

Table 1: Summary of Related Systems

System	Technique	Contribution	Limitation
BitChat, official repository [3]	BLE mesh, peer discovery, multi-hop forwarding	Offline messaging with no internet or server	Limited by BLE range and nearby nodes
BitChat Android [4]	BLE mesh, store-and-forward, AES-256-GCM	Offline messaging on Android with encryption	Battery drain, connectivity; security review status
BitChat blockchain-integrated platform [6]	BLE mesh, managed flooding, end-to-end encryption	Decentralised multi-hop communication	Range limits and authentication issues
The Register report [7]	Bluetooth mesh, peer-to-peer	Local messaging without the internet	Participating devices must be close
Times of India report [8]	Bluetooth mesh, end-to-end encryption, peer-to-peer	Privacy-focused offline messaging	Short local range, no worldwide reach
Serval Mesh [5]	Wi-Fi based phone mesh	Resilient phone communication in disasters	Early Android versions needed modified devices (reported)

Older networking studies also shape the design. When messages are flooded through an ad hoc network, the same message is sent again and again, nodes compete for the channel and packets collide; this is the broadcast storm problem [9]. Epidemic routing shows that messages can still reach their destination in a partly connected network if nodes store and forward them, although this creates extra copies [10]. Because of these two results we limit traffic with a TTL and by dropping duplicates. AODV [11] finds routes only when needed instead of flooding, but it has to maintain route state, which is a burden for phones whose BLE links keep changing. The Bluetooth SIG's Mesh Profile [12] is meant for provisioned embedded devices, and Android apps generally cannot use it. On smartphones the practical choice is therefore to do the relaying in the application, over GATT.[16].

1. Positioning and Research Gap

Most existing material on Bluetooth mesh chat apps describes the product and does not document a design that others can reproduce. The gap we address is one of documentation and method. Chatap offers a clear, modular design in which every feature is labelled as implemented or proposed, a packet and traffic analysis that can be checked by hand, and a measurement plan suitable for academic evaluation. The evidence we have supports the feasibility of the basic BLE chat path. It does not show that multi-hop relay or the security design works, and it does not show any gain over existing systems.

III. SYSTEM DESIGN AND METHODOLOGY

1. Requirements

The functional requirements are BLE advertising and scanning to find peers, connection management, and writing, sending, receiving and reading text messages. The proposed design adds storage on the phone, multi-hop forwarding, acknowledgements and retry. The non-functional requirements are as follows. The system must work without internet or a server and run on Android API level 26 or higher. It must stay responsive, so database work and duplicate checks must not run on the UI thread. Battery and memory use must stay low. It must cope with lost connections, denied permissions and malformed packets. Messages must be confidential and protected against tampering. At least two BLE-capable Android phones are needed for testing, and three or more to test multi-hop relay.

2. Layered Architecture

Chatap has four layers (Fig. 1). The presentation layer holds the chat screen, the list of peers and the message status display. The application layer holds the Message Manager, the Peer Discovery Manager and the Message Processing and Routing Manager. The Bluetooth communication layer holds the BLE scanner, the BLE advertiser and GATT communication. The data layer uses the Room database to keep conversations and peer records. All data stays on the phone and no central server is used.

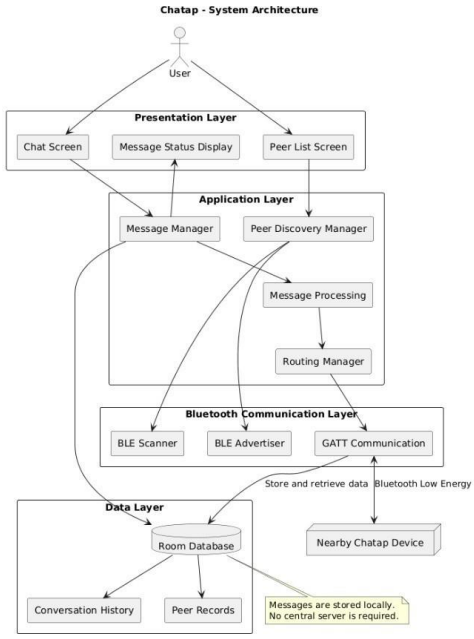


Fig. 1. Layered system architecture of Chatap.

3. Data Flow

Fig. 2 is the context-level data flow diagram. The user supplies message text and picks a recipient, and receives incoming messages and status updates. Nearby Chatap devices exchange discovery information, message packets, routing information and acknowledgements. Inside the system there are five processes: peer discovery, message processing,

transmission and routing, reception and acknowledgement, and local storage.

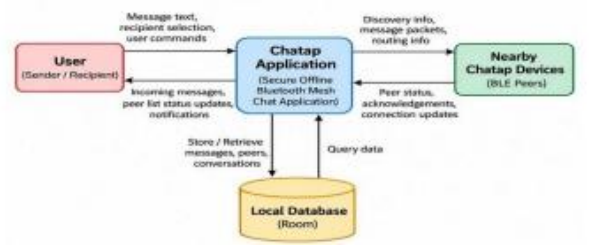


Fig. 2. Level-0 data flow diagram (context diagram).

4. Data Model

The local database has four entities (Fig. 3). A Peer has an identifier, display name, last-seen time, status and public key. A Conversation has a peer reference, title and timestamps. A Message has a conversation reference, sender, recipient, protected payload, type, timestamp, hop count, TTL and status. An Acknowledgement has a message reference, type and timestamp. One peer can be in many conversations, one conversation has many messages, and a message can have acknowledgement records. Each device uses a fixed identifier created by the app and not its hardware MAC address, because Android hides the real MAC address from apps and randomised addresses change over time.[2].

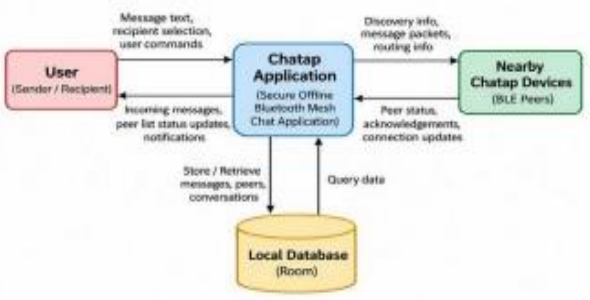


Fig. 3. Entity-relationship diagram of the local database.

Proposed Mesh Relay Protocol

Each packet carries a unique message ID, a sender ID, a destination ID, a timestamp, a message type, the payload and a TTL. When a device receives a packet it applies four steps. (1) It checks the format and rejects the packet if it is malformed. (2) It discards the packet if the message ID has been seen before. (3) If the packet is addressed to this device, it delivers and stores it.

(4) Otherwise it reduces the TTL by one and, if the TTL is still greater than zero, sends the packet to all connected peers except the one it came from. This is managed flooding. It is simple and keeps working when devices move.[9], [10].

Acknowledgements travel back as ACK packets, and the sender retries if none arrives in time. The app marks a message as delivered only after an acknowledgement is received, so it never claims a delivery that is not confirmed.

Traffic bound. Managed flooding can be bounded by hand. Let the network have N phones and E BLE links. Because every phone forwards a given message ID at most once, the total number of transmissions per message T is at most the sum of the node degrees, which equals 2E:

$$T \le \Sigma_v \deg(v) = 2E \qquad (1)$$

Without duplicate suppression there is no such bound: with an average of d neighbours, the number of copies C can grow roughly as

$$C \approx (d-1)^{\mathrm{TTL}} \qquad (2)$$

This is the reason the duplicate cache is mandatory and not an optimisation. The bound in (1) is a worst-case figure from the design; the real duplicate rate is one of the metrics in Section V. Packet size. BLE notifications and writes are limited by the ATT MTU, which defaults to 23 bytes (20 bytes of payload) and can be negotiated upwards on Android to at most 517 bytes. Table II shows a proposed packet layout. With the field sizes chosen here, the header is 61 bytes and encryption adds a 12-byte nonce and a 16-byte authentication tag, so 89 bytes of every packet are overhead. At the default MTU, a packet must therefore be fragmented across several writes. At the maximum MTU, a single write leaves about 400 bytes for the text. The layout is a proposal for discussion and should be adjusted to match the implemented format.[2], [16].

Table 2: Proposed Packet Layout (Field Sizes are Design Choices)

Field	Bytes	Purpose
Version, type, TTL	3	Format version; message type (text, ACK); remaining hops
Message ID	16	Unique identifier used for duplicate suppression and for matching ACKs
Sender ID, destination ID	16 + 16	App-generated device identifiers (not hardware addresses)
Timestamp	8	Ordering and replay limiting
Payload length	2	Length of the payload that follows
Nonce, ciphertext, tag	12 + *n* + 16	AES-256-GCM output; the nonce is unique per message

6. Proposed Security Design

Every device creates a long-term key pair and, where the phone supports it, keeps it in the Android Keystore. Two peers use elliptic-curve Diffie–Hellman, for example X25519 [13], to obtain a shared secret and then derive a session key from it. The message payload is encrypted with AES-256 in Galois/Counter Mode (GCM), which gives both confidentiality and integrity [14]. A new nonce must be used for every message under the same key. Encryption alone does not say who the other party is, so users must verify public keys, for example by comparing fingerprints in person or over another channel, to stop impersonation and man-in-the-middle attacks. Unique message IDs and timestamps limit replay. When messages are relayed, the payload stays encrypted end to end, so intermediate nodes see only routing information. Table III maps the main threats to the proposed countermeasures.[20].

Table 3: Threats and Proposed Countermeasures

Threat	Proposed countermeasure	Status
Eavesdropping on BLE traffic	AES-256-GCM on the payload, session key from X25519	Proposed
Tampering with a message	GCM authentication tag; malformed or failing packets are dropped	Proposed
Replay of old packets	Unique message IDs, duplicate cache, timestamps	Proposed
Impersonation, man-in-the-middle	Manual public-key fingerprint verification	Proposed
Relay node reading content	End-to-end encryption; relays see only routing fields	Proposed
Traffic analysis, metadata	Not addressed; routing fields remain visible to relays	Open

IV. IMPLEMENTATION

Chatap is written in Kotlin for Android using Android Studio, AndroidX, XML layouts with Material Design components, and the Room library [15]. Discovery uses the Android BLE scanner and advertiser APIs, and messages are sent using GATT services and characteristics [16]. To find and be found, a phone needs both the central role (scanning and connecting) and the peripheral role (advertising and hosting a GATT service).[2], [16].

The permissions needed for scanning, advertising and connecting differ between Android versions. From Android 12 (API 31) the Bluetooth scan, advertise and connect permissions are requested at run time, while older versions tie scanning to the location permission. The app asks for what the running version needs when it starts. If Bluetooth is off or a permission is refused, the app shows an error that explains the problem and stops.[17], [18].

Fig. 4 shows the implemented flow. The user opens the app and the app checks Bluetooth and permissions. It then scans and lists nearby Chatap devices. The user selects a peer and types a message. The message is validated, saved on the phone, packed into a packet and sent over BLE. The receiving phone validates the packet and checks whether it is a duplicate; it then either stores and

displays the message or rejects it. Table IV shows which features are implemented in the prototype and which are only proposed.

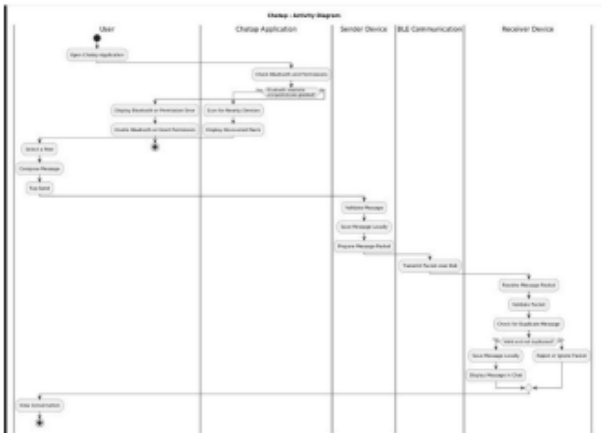


Fig. 4. Activity diagram of the messaging workflow

Table 4: Implementation Status

Feature	Status	Section
BLE discovery and automatic connection	Implemented	IV
Real-time one-to-one offline messaging and chat UI	Implemented	IV
Local storage (Room), duplicate check	[confirm status]	III-D
Multi-hop relay (TTL, flooding)	Proposed	III-E
Acknowledgements, retry, delivery status	Proposed	III-E
ECDH + AES-256-GCM encryption	Proposed	III-F
Group chat, file sharing, read status	Future work	VIII

V. VERIFICATION AND PRELIMINARY RESULTS

1. What Has Been Verified

We have not yet completed a quantitative evaluation, so we report no performance numbers. What has been shown is functional. On [number and models of phones used] running [Android versions], the prototype finds nearby phones running

the app, connects to them automatically, and delivers text messages in real time without internet. This is an observation of working behaviour in informal use. It is not a measurement, and it says nothing about delivery ratio, latency, range or battery use.

2. Planned Evaluation

Table V lists the planned metrics. The results will replace the empty column before submission. Several design choices matter for the validity of these measurements. One-way latency cannot be taken as a difference between timestamps on two phones, because their clocks are not synchronised; we will instead measure round-trip time on one device and halve it, or synchronise the clocks first and report the residual error. Delivery ratio will be measured at fixed distances in an open space and again indoors, with the environment stated. Each condition will be repeated at least 30 times and reported as a mean with standard deviation and a 95% confidence interval, and delivery ratio as a proportion with a binomial confidence interval. The phone models, Android versions, screen state and battery level will be recorded for every run.

Table 5: Planned Evaluation Metrics

Metric	Setup	Method	Result
Discovery time (s)	2 devices	Time from start of scan until the peer appears in the list	[to measure]
Delivery ratio (%)	2 devices, 100 messages	Messages received divided by sent, at 1, 5 and 10 m	[to measure]
End-to-end latency (ms)	2 devices	Round-trip time halved, or synchronised clocks with stated error	[to measure]

Multi-hop delivery	3–4 devices in a chain	Delivery ratio and latency for each hop count	[to measure]
Duplicate rate	3 or more devices	Duplicate copies received per message; compare with the $2E$ bound	[to measure]
Battery drain (%/h)	Foreground and background	Battery statistics over 1 hour, screen state fixed	[to measure]
Encryption overhead (ms)	Encrypted vs plain	Added cost per message once encryption exists	[to measure]

Further tests will cover reconnection after a dropped link, denied permissions, malformed packets, and different Android versions and phone makers. Multi-hop tests need relay and acknowledgements to be implemented first, so they depend on the work listed in Section VIII.

3. Interpretation

Taken together, the demonstration gives preliminary evidence that the basic BLE discovery, connection and one-to-one messaging path works on the tested phones. It does not show that the relay protocol delivers messages across several hops, that duplicate suppression keeps traffic within the bound in Section III-E, that the security design resists attack, or that the app is economical with battery. Those statements can be made only after the implementation and measurements in Table V are complete.

VI. DISCUSSION

Chatap treats offline messaging as a layered problem. The Bluetooth layer finds and connects to neighbours, the application layer decides what to do with each packet, and the data layer keeps records on the phone. Keeping these apart means the relay logic can be added without rewriting discovery, and the encryption can be added without changing the routing fields.

The most useful design decision is, in our view, the rule that a message is shown as delivered only after an acknowledgement arrives. In an unreliable radio network, a phone that reports success because it handed a packet to the Bluetooth stack would mislead the user. The cost is extra traffic and the need for retry timers, which must be tuned during evaluation.

Managed flooding was chosen over route-based schemes because phones move and links come and go. Its price is redundant transmission. The bound of 2E transmissions per message shows that the cost is controlled by the number of links, so it should be acceptable in small groups and will grow in dense crowds. Whether the real duplicate rate stays near that bound is an open empirical question.

Limitations and Threats to Validity

Construct validity. A working demonstration shows that messages can be sent. It does not show that the system is reliable, fast or energy efficient, and it should not be read as evidence of any of these.

Internal validity. Tests so far were run by the developers on a small number of phones in favourable conditions. The planned evaluation must control distance, obstacles, radio interference and background apps.

External validity. Nothing here generalises to crowds, large buildings, outdoor disaster conditions or phones other than those tested. BLE behaviour and the number of simultaneous connections differ between manufacturers.

Range and relaying. BLE links are short, and multi-hop delivery needs relay phones that are present, awake and within range.

Background execution. Android restrictions and battery optimisation can limit scanning and relaying, so a foreground service may be needed.[19].

Flooding overhead. Managed flooding sends redundant transmissions and does not scale to dense networks [9].

Security. The security claims hold only after the mechanisms are implemented, reviewed and tested. Peer authentication needs a key-verification step, and relays can still observe routing metadata.

Reproducibility. The related-work table draws partly on repositories and news reports that are not peer reviewed. The prototype has not yet been released publicly, so readers cannot repeat the demonstration without contacting the authors.

VII. CONCLUSION AND FUTURE WORK

This paper described Chatap, an Android application for offline messaging over BLE that needs no internet and no central server. The prototype can discover devices, connect and exchange messages in real time. The paper also specifies a layered architecture, a data model, a managed-flooding relay protocol with a traffic bound, a packet-size analysis and an X25519/AES-256-GCM security design. These are steps towards a secure mesh messenger, not a finished one.

At this stage the contribution is a system design and a prototype, not a new routing method and not a measured result. The next step is to implement multi-hop relay, acknowledgements, local storage and encryption, and to measure them as planned in Table V. After that, the most promising directions are group chat, file sharing, delivery and read status, store-and-forward for devices that are offline, a foreground service for reliable background operation, and an independent security review. Claims about effectiveness should wait until that evaluation is done.

Data and Code Availability

The prototype source code is [add repository link, or state "available from the authors on request"]. No performance data are reported in this version of the paper.

Acknowledgment

The authors thank Dr. Jyoti Surve, Head of the Department of Information Technology, and Dr. Vaishali Patil, Principal, International Institute of Information Technology, Pune, for their support.

REFERENCES

1. A. U. Mauthe et al., "Disaster-resilient communication networks: Principles and best practices," in Proc. 8th Int. Workshop Resilient Netw. Design Modeling (RNDM), Halmstad, Sweden, 2016, doi: 10.1109/RNDM.2016.7608262.
2. Bluetooth SIG, "Bluetooth core specification," Ver. 5.4, 2023. [Online]. Available: https://www.bluetooth.com/specifications/specs/
3. permissionlesstech, "BitChat (official repository)," GitHub, 2025. [Online]. Available: https://github.com/permissionlesstech/bitchat
4. permissionlesstech, "BitChat Android,"GitHub, 2025. [Online]. Available: https://github.com/permissionlesstech/bitchat-android
5. P. Gardner-Stephen, R. Challans, J. Lakeman, A. Bettison, D. Gardner-Stephen, and M. Lloyd, "The Serval Mesh: A platform for resilient communications in disaster & crisis," in Proc. IEEE Global Humanitarian Technol. Conf. (GHTC), 2013.
6. "BitChat: A decentralized blockchain-integrated messaging platform," ResearchGate, 2025. [Online]. Available: https://www.researchgate.net/publication/393777552
7. "Jack Dorsey debuts BitChat," The Register, Jul. 8, 2025. [Online]. Available: https://www.theregister.com/2025/07/08/jack_dorsey_debuts_bitchat/
8. "What is BitChat?," Times of India, 2025. [Online]. Available: [add the full article URL]

9. S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu, "The broadcast storm problem in a mobile ad hoc network," in Proc. ACM MobiCom, 1999, pp. 151–162.
10. A. Vahdat and D. Becker, "Epidemic routing for partially connected ad hoc networks," Duke Univ., Durham, NC, USA, Tech. Rep. CS-2000-06, 2000.
11. C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc on-demand distance vector (AODV) routing," IETF, RFC 3561, Jul. 2003.
12. Bluetooth SIG, "Bluetooth mesh profile specification," Ver. 1.0.1, 2019.
13. A. Langley, M. Hamburg, and S. Turner, "Elliptic curves for security," IETF, RFC 7748, Jan. 2016.
14. M. Dworkin, "Recommendation for block cipher modes of operation: Galois/Counter Mode (GCM) and GMAC," NIST, Gaithersburg, MD, USA, Spec. Publ. 800-38D, 2007.
15. Android Developers, "Room persistence library." [Online]. Available: https://developer.android.com/training/data-storage/room
16. Android Developers, "Bluetooth Low Energy overview." [Online]. Available: https://developer.android.com/develop/connectivity/bluetooth
17. Android Developers, "Bluetooth permissions." [Online]. Available: https://developer.android.com/develop/connectivity/bluetooth/ bt-permissions
18. Android Developers, "Behavior changes: Apps targeting Android 12." [Online]. Available: https://developer.android.com/about/versions/ 12/ behavior-changes-12
19. Android Developers, "Background tasks overview." [Online]. Available: https://developer.android.com/develop/background-work/ background-tasks
20. Android Developers, "Android Keystore system." [Online]. Available: https://developer.android.com/privacy-and-security/keystore
21. Sharjil Iqbal. "NETWORK ATTACK DETECTION BY GROUP SEARCH OPTIMIZATION USING CONVOLUTIONAL DEEP LEARNING MODEL