Journal of Network and Computer Applications 219 (2023) 103723

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Enabling secure lightweight mobile Narrowband Internet of Things (NB-IoT) applications using blockchain

Vamshi Sunku Mohan [a,*], Sriram Sankaran [a], Priyadarsi Nanda [b], Krishnashree Achuthan [a]

[a] Center for Cybersecurity Systems and Networks, Amrita Vishwa Vidyapeetham, Amritapuri, India
[b] University of Technology Sydney, Australia

ARTICLE INFO

Keywords:
Blockchain
InterPlanetary File System (IPFS)
Memory optimisation
NarrowBand Internet of Things (NB-ioT)
Zero-knowledge proof

ABSTRACT

Narrowband Internet of Things (NB-IoT) is a low bandwidth 3GPP communication standard transmitting small quantities of data over long distances at random intervals. However, as NB-IoT cannot support seamless handover between base stations, its applications are limited to stationary devices, which may result in the potential risk of fake base station connections in an attempt to maintain connectivity across cells. Considering characteristics such as low power consumption and high connection density, researchers envision using NB-IoT in mobile applications such as public-bike sharing, pet tracking etc. Connecting NB-IoT devices using decentralised architecture such as blockchain ensures seamless communication in mobile applications and eliminates bottlenecks due to multiple data requests observed in centralised networks. In this paper, we develop a hybrid blockchain framework facilitating mutual authentication between base stations to enhance user privacy and prevent fake base station connections and certificate transfers. Zero-knowledge proof used as the consensus algorithm enhances user privacy and message confidentiality. IoT devices are designed to store the hashes of their approved transactions as a linear hash chain instead of the complete merkle tree to minimise hash verification complexity. Additionally, base station memory is partitioned dynamically to enhance scalability and memory utilisation efficiency. We prototype our framework on Remix IDE in Ethereum and implement it on Raspberry Pi 4. The security of the proposed framework is formally verified using Scyther. Further, we show that our approach achieves 80.50% lower computational power, 74.73% lower execution time and 50% lower memory, respectively, in comparison with existing schemes making our proposed scheme lightweight.

1. Introduction

Narrowband Internet of Things (NB-IoT) (Anon, 2017b) is a Low Power Wide Area Network (LPWAN) developed by 3GPP (Anon, 2019b) providing wider coverage, high connection density, longer battery life etc. It works on a reduced bandwidth of 180 kHz per carrier using Orthogonal Frequency Division Multiplexing (OFDMA) for Downlink communication and Single-Carrier FDMA (SC-FDMA) for Uplink communication. NB-IoT neither supports low latency applications nor provides seamless handover (Praptodiyono et al., 2020) of devices travelling between base stations. Hence, it is ideally suited in low-power stationary IoTs (Anon, 2012; Samara et al., 2022) such as water meters, smoke detectors (Anon, 2023a) etc., transmitting small quantities of data at irregular intervals.

Numerous studies have been conducted to maximise the battery life of devices using NB-IoT by varying eDRX (Extended Discontinuous Reception) and PSM (Power Saving Mode) to optimise ON-OFF periods (Sultania et al., 2021). Since NB-IoT is currently implemented in stationary applications (Samara et al., 2022), researchers envision its future usage in mobile applications such as public-bike sharing, pet tracking, etc. Research in this direction (Anon, 2020) would develop energy-efficient NB-IoT mobile applications with longer battery life, wider coverage and scalability.

As NB-IoT does not include support for authentication and certificate transfer (Anon, 2019b) while moving between various cells, devices are not automatically allocated to visited base stations to ensure seamless, uninterrupted services. Hence, devices are required to choose base stations manually (Anon, 2019b). This may lead to fake base station connections and Denial of Service (DoS) attacks due to reduced bandwidth, resulting in user data and privacy loss. Research in this regard (Dorri et al., 2017) has proposed a centralised architecture to

* Corresponding author.
E-mail addresses: vamshis@am.amrita.edu (V. Sunku Mohan), srirams@am.amrita.edu (S. Sankaran), Priyadarsi.Nanda@uts.edu.au (P. Nanda), krishna@amrita.edu (K. Achuthan).

https://doi.org/10.1016/j.jnca.2023.103723
Received 15 March 2023; Received in revised form 20 June 2023; Accepted 16 August 2023
Available online 19 August 2023
1084-8045/

connect devices to cloud servers hosted by the service provider. Cloud servers authenticate devices, ensure a seamless handover and handle communication requests between devices. However, cloud servers become a bottleneck in serving the device requests during peak-traffic hours resulting in delayed responses. Additionally, a centralised architecture inhibits system scalability as the number of devices and transactions increase.

Limitations caused due to the centralised architecture can be overcome by implementing a decentralised technology such as a blockchain (Sankaran et al., 2018; Kumar et al., 2020) capable of providing peer-to-peer communication between participants. The decentralised architecture eliminates the bottleneck arising from a single point of connection to the centralised server and the requirement for certificate generation and manual transfer between base stations to authenticate the devices and visited base stations during handover. However, in public blockchain architecture (Jabbar et al., 2020), blocks generated by devices are stored in the block pool to be authenticated by the peer nodes. Hence, the device details and transaction contents are visible to all the nodes in the network, which may lead to impersonation, fake block generation and double spending attacks. Hence, device details need to be verified when devices join the blockchain network. In private blockchains (Anon, 2023b), only devices with good transaction performance are allowed access to smart contract functions to avoid attackers entering and manipulating the blockchain to cause 51% attacks and double-spending. Additionally, smart contracts' transparency must be limited to ensure device and transaction privacy.

These system requirements can be addressed using hybrid blockchains (Anon, 2023b), which combines the merits of public and private blockchains. Hybrid blockchain enables the implementation of authentication protocols between IoT devices (Ghasempour, 2019) and base stations to enhance security, reduce smart contract transparency and alleviate the limitations caused due to reduced bandwidth and centralised architecture. The hybrid architecture allows devices to choose the level of transparency for personal and block details. Reduced node transparency limits the possibility of information tracking and fake block generation. Hybrid blockchain eliminates certificate transfer and provides decentralised data storage as in conventional blockchains.

In this paper, we develop a hybrid blockchain framework to secure NB-IoT and store the hashes of their approved transactions as a linear hash chain instead of the entire merkle tree to minimise hash verification complexity, thereby reducing energy and memory consumption. We implement Non-interactive Zero-Knowledge Proof (ZKP) (Li et al., 2020) as the consensus algorithm to provide user privacy and enable rapid block verification and scalability. Base station memory is dynamically partitioned to store ID's and transaction hashes of incoming devices as linear hash chains to enhance scalability and reduce the memory required to authenticate devices. This design eliminates the requirement of updating and verifying the merkle tree each time devices initiate a transaction or enter a new cell. Further, we prototype the algorithm on Remix IDE in Ethereum (Anon, 2017d) and implement it on Raspberry Pi 4 (Anon, 2019c). The security of our approach was formally validated using Scyther (Anon, 2014). We evaluate the model's performance, compare our results with those in existing works and show that our approach consumes 80.50%, 74.73% and 50% lower computational power, execution time and memory, respectively. As per our knowledge, we are the first to propose an approach to partition base station memory and store device hashes individually as linear hash chains instead of merkle trees for efficient memory utilisation in resource-constrained mobile NB-IoT applications.

1.1. Primary contributions

Primary contributions of our paper are listed as follows.

- We propose a hybrid blockchain framework facilitating mutual authentication between base stations to implement mobile applications in NB-IoT. This framework eliminates certificate transfer and provides seamless connectivity across cells.
- We propose to dynamically partition base station memory into *'write only'* and *'read and write'* to store the merkle tree and device details to eliminate repeated merkle tree verification and increase memory usage efficiency.
- We leverage non-interactive ZKP as the consensus algorithm to enhance user privacy and minimise transaction verification delay.
- Security of the proposed framework is formally verified using Scyther to ensure protection against attacks.

The paper is organised as follows: Section 2 discusses the Related Work. Section 3 lists the salient features of our work. Section 4 explains various components of the proposed framework, such as consensus algorithm, unique ID generation, reputation updation etc., and a computational complexity study. Section 5 presents the experimental prototype, formal validation of protocols with Scyther and model performance evaluation and provides a survey on security analysis and mitigation strategies. Section 6 discusses inherent security and regulatory issues and ways to mitigate channel congestion and interference in blockchain. Section 7 concludes the paper and describes possible future work.

2. Related work

Few approaches exist in developing a blockchain framework to facilitate seamless handover in mobile NB-IoT applications. A detailed study showed that a majority of them (Popli et al., 2019; Zhang and Zhao, 2022; Cao et al., 2020; Nguyen et al., 2021; Jiang et al., 2019; Gao et al., 2018; Javaid et al., 2020; Wang et al., 2020; Qi et al., 2021) provide support for scalability. Whereas only a few of them (Wang et al., 2020; Qi et al., 2021) propose memory optimisation schemes. However, none of them provide support for mobility in NB-IoT devices. In this paper, we develop a hybrid blockchain framework providing scalability and memory optimisation architecture while ensuring mobility in NB-IoT applications. We categorise the existing works in the following manner.

NB-IoT Applications and Performance Evaluation: Nguyen et al. (2020) proposed to transmit vehicle data to decentralised servers via stationary NB-IoT edge routers. The authors show that NB-IoT routers provide good connectivity and security. However, the paper does not propose memory optimisation in routers, implement NB-IoT in mobile devices and study the security aspects of devices outside the range of the base station. Popli et al. (2019) have published an extensive survey on technical features and resource allocation in NB-IoT. The authors proposed energy-efficient techniques for health monitoring systems and zonal thermal-pattern analysis. However, the authors do not mention using these approaches specifically for mobile scenarios. They do not discuss the memory allocation and security features implemented to prevent DoS and impersonation attacks due to low bandwidth in NB-IoT. Routray et al. (2021b,a), An and Routray (2017), Routray et al. (2021c) have proposed several works describing the theoretical implementation of NB-IoT in smart grids, healthcare and smart cities. However, the performance and security of these models have not been evaluated. In contrast to these approaches, Zhang and Zhao (2022) proposed a vehicular tracking application where vehicular load and position are measured using precision inclination sensors attached to the vehicles. Values thus measured are transmitted to the cloud through static NB-IoT modules. Martín et al. (2018) proposed a signal propagation model using Random-Access Procedure (RAP). The performance of the model was evaluated for varying modulation parameters, power control mechanisms and transmission modes. Similarly, Cao et al. (2020) have also proposed a random access load control model using Reinforcement learning by utilising time slot analysis and coverage level transition mechanism to predict network congestion and adjust access-level restriction parameters to improve packet delivery rate and improve system performance.

NB-IoT with Blockchain: Hong et al. (2019) proposed a detailed mathematical model to integrate blockchain with NB-IoT in each of

Table 1
Comparison of existing works with our approach

Proposed approach	Practical implementation	Memory optimisation scheme	Lightweight architecture support	Computational complexity	Device security	Support for scalability	Support for mobility
Nguyen et al. (2020)	Yes	No	No	Medium	High	No	No
Popli et al. (2019)	No	No	Yes	Low	Low	Yes	No
Routray et al. (2021b)	No	No	Yes	Medium	–	No	No
Routray et al. (2021a)	No	No	Yes	–	–	No	No
An and Routray (2017)	No	No	Yes	High	Medium	No	No
Routray et al. (2021c)	No	No	Yes	High	–	No	No
Zhang and Zhao (2022)	Yes	No	Yes	Low	Medium	Yes	No
Martín et al. (2018)	Yes	No	No	Low	High	No	No
Cao et al. (2020)	Yes	No	Yes	Low	High	Yes	No
Hong et al. (2019)	Yes	No	No	Low	High	No	No
Ile et al. (2019)	Yes	No	No	Medium	–	No	No
Nguyen et al. (2021)	Yes	No	Yes	Low	High	Yes	No
Jiang et al. (2019)	Yes	No	Yes	Low	Low	Yes	No
Gao et al. (2018)	Yes	No	Yes	Low	High	Yes	No
Javaid et al. (2020)	Yes	No	Yes	Low	High	Yes	No
Wang et al. (2020)	Yes	Yes	No	Low	–	Yes	No
Qi et al. (2021)	Yes	Yes	No	Low	High	Yes	No
Du et al. (2023)	Yes	No	Yes	Low	Medium	Yes	No
Mohammed and Chopra (2023)	Yes	No	Yes	Low	–	No	Yes
Shahjalal et al. (2022)	Yes	Yes	Yes	Low	High	No	No
Na and Park (2022)	Yes	Yes	Yes	Medium	Medium	Yes	No
Our Work	Yes	Yes	Yes	Low	High	Yes	Yes

the Sensing, Transport and Transaction layers to overcome possible cyber attacks. However, the paper does not present a seamless handover mechanism, does not discuss memory and energy optimised protocols to communicate in low bandwidth channels and does not simulate the model's performance on hardware. The University of Zurich (Ile et al., 2019) studied the implementation of blockchain on NB-IoT and LTE-m wallet clients. Their approach was evaluated in terms of energy consumed and data overhead. Nguyen et al. (2021) designed a blockchain system to establish communication between stationary buyers and seller NB-IoT nodes through the base station. Transactions initiated were forwarded to the distributed ledger for approval. The hash of the approved blocks was added to the blockchain. However, authors do not evaluate the robustness of the proposed authentication protocols against attacks on blockchain and the work is limited to transactions between stationary devices and does not consider mobile applications.

Our Proposed Method: In contrast, we propose implementing blockchain on mobile IoT devices using NB-IoT. Devices are designed to eliminate merkle tree by storing transaction hashes as linear hash chains. This architecture reduces devices' energy and memory consumption during block verification as nodes are designed to check and update the latest block hash. Non-interactive ZKP is implemented to enable rapid block verification, scalability and user privacy by eliminating the requirement of group consensus. We analyse energy-performance-security trade-offs of the algorithm, formally evaluate using Scyther and show that our approach is energy efficient and secure.

A detailed comparison of our approach with some of the existing works with respect to primary contributions of our paper, such as memory optimisation, support for scalability, computational complexity, device security etc., are given in Table 1. Terms listed in Table 1 are explained as follows.

1. *Low, Medium and High computational complexity* - *Low*, *Medium* and *High* computational complexity represents varying level of memory storage complexity and traversal time complexity required to execute the proposed mechanisms. Algorithms considered for low, medium and high complexity are listed as follows.
    - *Low memory complexity* - Low memory required to execute. Eg. $O(1)$, $O(n)$, $O(\log(\log n))$, $O(\log n)$
    - *Medium memory complexity* - Moderate memory required to execute. Eg. $O(n + k)$
    - *High memory complexity* - High memory required to execute. Eg. $O(n^k)$, $O(nk)$, $O(n \log(n))$
    - *Low traversal time complexity* - Define the input for which algorithm takes less time or minimum time. In the best case, lower bound of an algorithm is computed. Eg. $T(1)$, $T(\log n)$, $T(n)$
    - *Medium traversal time complexity* - Takes all random inputs and calculates the computation time for all inputs. Eg. $T(n+k)$
    - *High traversal time complexity* - Define the input for which algorithm takes a maximum time. In the worst case, upper bound of an algorithm is computed. Eg. $T(n \log(n))$, $T(nk)$, $T(n^k)$

2. *Low, Medium and High device security* - *Low*, *Medium* and *High* device security represents the security of the proposed approach against lower, moderate and higher number of attacks.
3. A *hyphen* is written in place of parameters not examined by the authors.

3. Salient features

In this section, we list the salient features of our work.

1. Mobility in NB-IoT devices - We implement NB-IoT in mobile applications by connecting the devices and base stations using a blockchain framework. Since traditional consensus algorithms, such as Proof of Work (PoW), require higher computational resources, they are unsuitable for NB-IoT applications. Hence, we leverage non-interactive ZKP as the consensus algorithm. Our design ensures mobility and low battery and memory requirements in NB-IoT devices.
2. Lightweight - Since NB-IoT devices are memory-constrained and operate at short intervals, the number of handshake protocols required to generate and encapsulate the messages and verify the transaction hashes upon block approval needs to be minimised to reduce battery consumption. In this paper, we design devices to be lightweight to store hashes only related to the transactions initiated by them.
3. Memory Optimisation - Base stations store transaction hashes of all devices within their range. Therefore, the transaction verification process results in increased memory and execution time to verify the block contents along with the device's previous hashes stored in the merkle tree. Hence, we optimise performance by partitioning the memory to store the merkle tree while dynamically allocating the rest to store information and hashes related to individual devices.
4. Scalability - To ensure efficient usage of memory and reduce hash verification time during block approval, the base station memory is dynamically partitioned and allocated to store the details of devices joining and leaving the cells. Memory thus allocated stores the transaction hashes and devices' information such as unique ID, previous message hash etc. As our work is developed based on the assumption that base stations have high storage capacity, they are equipped to support the increase in transactions and incoming devices.
5. Device Security - As we connect NB-IoT devices using a hybrid blockchain framework, only the base stations are authorised to verify transactions and devices are permitted to initiate transactions. This design eliminates fake block generation and verification. As non-interactive ZKP is used as the consensus algorithm, group consensus is eliminated and message transparency is limited. When devices leave the cell, home base stations verify the legitimacy of the visited base station that the device intends to join. This design eliminates impersonation attacks caused due to fake base station connections, which might occur when NB-IoT devices manually connect to visiting base stations.
6. Computational Complexity - IoT devices are designed to check, update and store hashes of transactions generated by them as linear hash chains in contrast to the entire merkle root in conventional blockchain framework. It is assumed that the number of hashes in individual linear hash chains is much lower than the total number of hashes in the merkle tree of the same height. This design reduces the hash chain computational complexity from $O(2^n)$ to $O(n)$, where n refers to the number of hashes verified. Similarly, base station memory is dynamically allocated to store transaction hashes and unique IDs of individual devices under their influence as linear chains. This design reduces computational time for hash verification upon receiving a *'Leave Request'* and *'Device Join'* from $T(\log_2 n)$ to $T(m)$. *'m'* refers to hashes in linear hash chains stored in individual memory partitions and $m << n$. $T(\log_2 n)$ refers to the time taken by the base station to verify all the transactions of the merkle tree and $T(m)$ refers to the time taken to check transaction hashes in linear hash chains. However, during transaction verification, the base station verifies only the latest device hash as a part of zero-knowledge proof reducing time complexity to $T(1)$.

4. Proposed framework

In this section, we describe the blockchain framework securing mobile IoTs shown in Fig. 1. The framework is designed to contain the following entities.

1. *IoT Device* - As NB-IoT is resource-constrained, IoT devices are designed to store and verify the latest block hash to reduce energy and memory consumption.
2. *Base Station* - The base station verifies the device authenticity upon receiving a request from a device to join and relays device transactions to InterPlanetary File Storage (IPFS) (Anon, 2015a) to be stored. Mobile devices in a particular cell form a hybrid blockchain with a base station acting as a central node. Devices communicate with base stations using non-interactive ZKP for block approval.
3. *InterPlanetary File Storage (IPFS)* - IPFS connects base stations to provide decentralised data storage and sharing.
4. *Authentication Server* - Authentication Server verifies the device details upon receiving a request from a device to join to avoid impersonation attacks.
5. *Data Server* - Upon successful device verification, Authentication Server authorises the Data server to provide a Unique ID to the device. The device uses the unique ID as an identifier during block generation and verification to avoid fake block generation.

A detailed working of the framework is described as follows.

4.1. Hybrid blockchain

A hybrid blockchain Alkhateeb et al. (2022) is a combination of public and private blockchains where any node can join the blockchain framework but can access smart contract functions only when permitted by the administrators. Hybrid blockchain can choose the transparency of information, i.e., devices can allow block details to be public and device details to be private (Innocent and Prakash, 2019).

In this paper, we develop a hybrid blockchain framework to implement authentication protocols between devices and base stations by defining the transparency of functions in the smart contract as shown in Algorithms 1 and 2. Transparency is set by defining the functions using 'Only Owner' in Remix IDE, which means that only the component (device or base station) for which the function is defined can access it. Accordingly, NB-IoT devices are permitted to generate a block and construct a non-interactive ZKP request to send to the base station for block verification. Similarly, base stations are allowed to verify blocks, update merkle trees with verified blocks, join devices into their respective blockchains and upload and retrieve hash from IPFS. This design prevents impersonation attacks, fake block generation and approval, thus limiting 51% attacks (Aponte-Novoa et al., 2021). The function 'Only Owner' allows devices to select personal and block details to be made public, limiting the possibility of information tracking.

4.2. IoT device

IoT devices are designed to store the approved transaction hashes in an array as a linear hash chain instead of the entire merkle tree to minimise storage capacity and hash verification complexity. Devices are assigned an initial reputation by manufacturers as a measure of trust, which the base station verifies during transaction verification. Upon

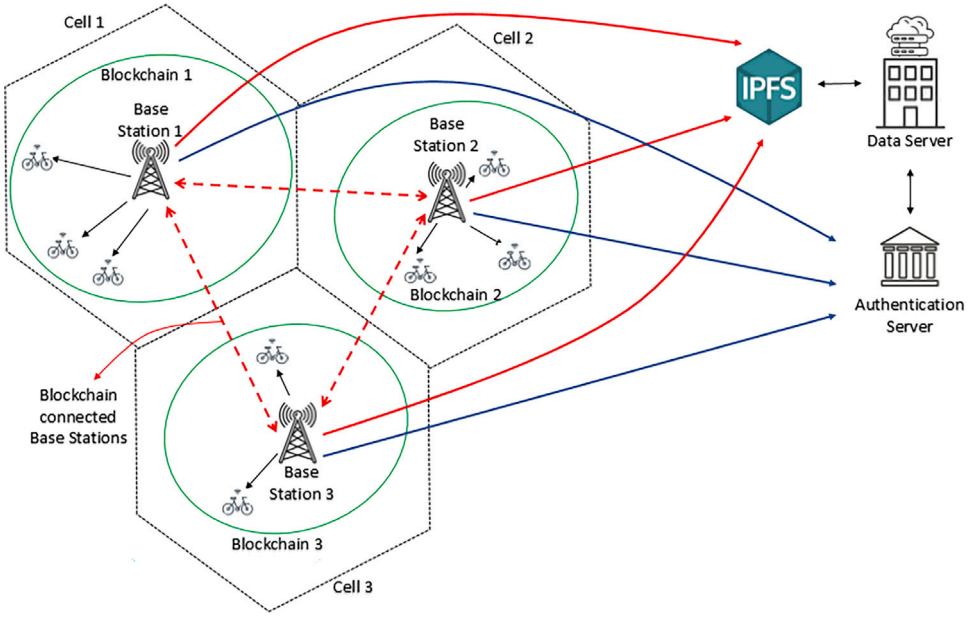

Fig. 1. Blockchain framework.

successful block transaction verification, reputation is incremented by *'1'* by the base station as a reward. As devices travel from one cell to another, they send a *'Leave Request'* and a *'Join Request'* to home and visited base stations, respectively, to avoid connection to fake base stations and ensure authenticity. Devices encrypt messages with *'Encryption Secret'* using non-interactive ZKP shared by base stations to ensure message integrity when joining that particular cell. As NB-IoT is secured using a hybrid blockchain, device secrecy is ensured by limiting device transparency and block details to other blockchain nodes.

4.3. Consensus algorithm

In conventional blockchain frameworks, devices generate blocks with the previous block's hash, merkle root, unique ID etc. Blocks are then added to the consensus pool to be verified by other blockchain participants. In such frameworks, when the number of participants is minimal, an attacker can guess the contents of the next block, impersonate and transmit zero-value blocks and blocks with fake device IDs to delay verification and fork the blockchain. These attacks may lead to a breach of device confidentiality. We propose implementing non-interactive ZKP as the consensus algorithm to overcome the limitations mentioned above, eliminate group consensus and limit block content visibility.

Non-interactive ZKP requires the device to send a complex mathematical equation to the base station to be solved to prove the device's identity. The algorithm reduces the number of interactions initiated by the base station to the device to retrieve its details to solve the equation, thus, reducing phishing attacks. Usage of this algorithm removes the requirement of block approval by other participants in the blockchain resulting in faster block approval. Transactions therefore approved, are updated by base stations in IPFS. The approved transactions' hash and updated reputation are transmitted to the devices.

4.4. Unique ID generation

The unique ID is a pseudonym used by IoTs as an identifier in a blockchain. Unique ID protects device addresses (hash of the private key) from being visible to other participants and authenticates devices during block verification. The public–private key pairs used to define the node's permanent address and protect the block's integrity are generated using Elliptic Curve Digital Signature Algorithm (ECDSA) shown in Eq. (1). As defined in the Bitcoin use-case (Anon, 2017a), *'a'* and *'b'* are assigned 0 and 7, simplifying the ECDSA curve to Eq. (2), which is used to derive the public and private keys as described by Genç and Afacan (2021).

$$y^2 = x^3 + ax + b(mod\ \ M) \tag{1}$$

$$y^2 = x^3 + 7(mod\ \ p) \tag{2}$$

where,
a, b = Coefficients in the elliptic curve equation,
M = Prime number defining the finite field

Despite the usage of public and private keys, device address tracking and creating fake identities are possible in low-traffic scenarios. These attacks may lead to fake block generation, thus reducing device confidentiality and trust. To prevent these attacks, we design base stations to generate a 32-bit Unique ID using *'Universal Unique Identifier (UUID) library'* each time a device enters a new cell to prove its identity. UUID library appends the *'Old Unique ID'*, i.e., the unique ID possessed by the device in the home base station, the *'Encryption Secret'* shared by the visiting base station during *'Device Join'* procedure and a *'Nonce'* to form the *'Seed'* as shown in Eq. (3). The *'uuid.UUID()'* function converts the *'Seed'* into a 8-bit integer which is then used to generate the unique ID as shown in Eq. (4).

$$Seed = Old\ Unique\ ID \parallel Encryption\ Secret \parallel Nonce \tag{3}$$

$$Unique\ ID = uuid\_UUID(random.getrandbits(Seed)) \tag{4}$$

Handshake protocols involved in authenticating the device to obtain the Unique ID from the *'Data Server'* upon joining a new cell are shown in Fig. 2. The process of sharing the *'Encryption Secret'* by the base station to the device is shown in Fig. 3. Symbols used in the figures are explained as follows.

N_A, N_B, N_C, N_D = Nonce generated by A, B, C and D respectively,
T_A, T_B, T_C, T_D = Timestamps generated by A, B, C and D respectively

4.5. Device join procedure

Devices aiming to join a base station authenticate themselves at the Authentication server using their existing Unique ID. Upon successful

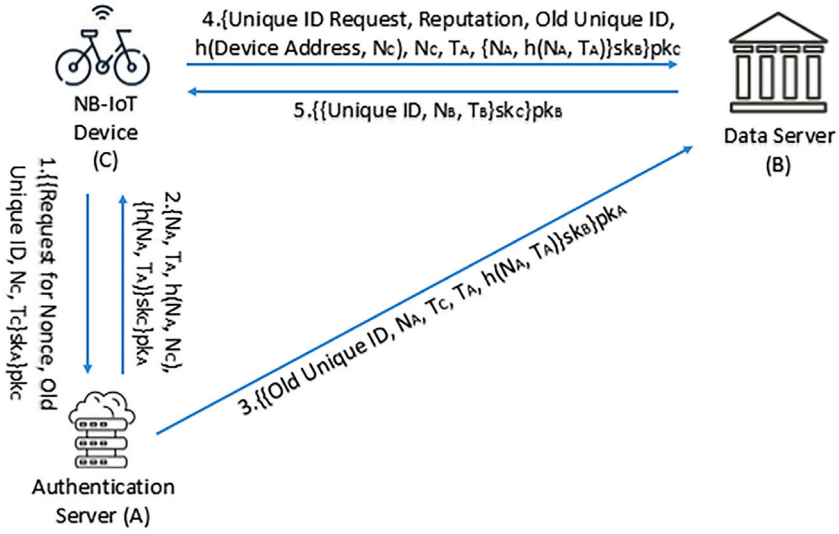

Fig. 2. Unique ID generation.

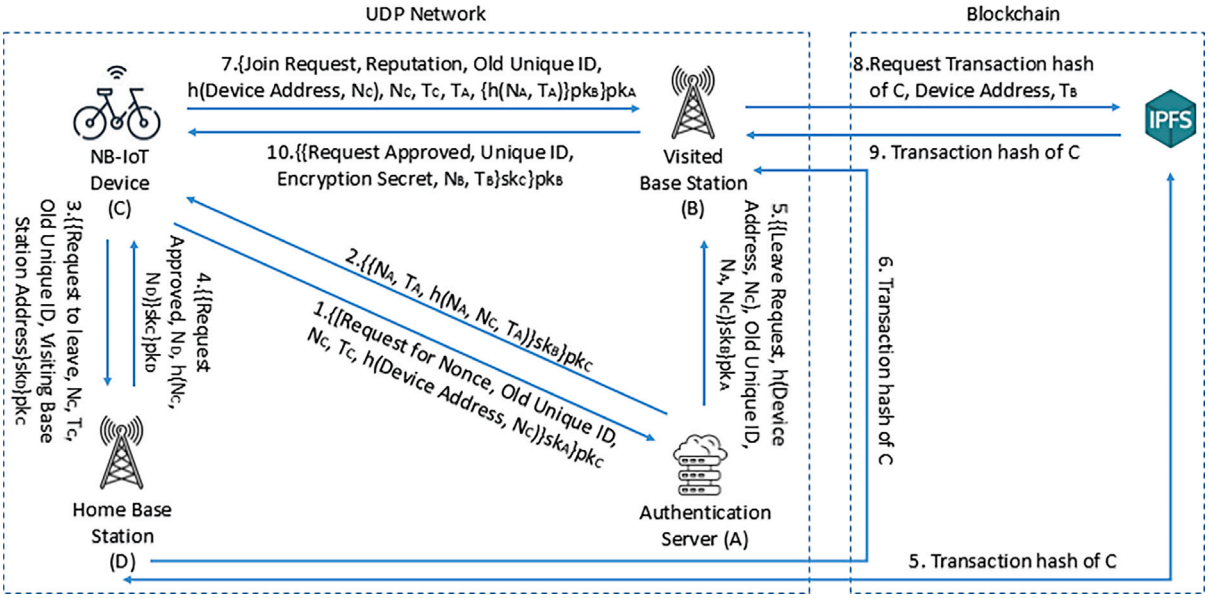

Fig. 3. Device join procedure.

authentication, the devices are designed to send a *'Leave Request'* to the current base station along with the address of the visited base station it intends to join. In this paper, we assume that upon reaching the edge of a cell, devices receive connections to join in the form of addresses from surrounding base stations to ensure seamless connectivity. On approval of the *'Leave Request'*, the current base station uploads the hash of the device transactions on IPFS to be retrieved and verified by the visiting base station. The visiting base station retrieves the device information from IPFS and verifies the device's identity. Upon verification, an *'Encryption Secret'* and a new Unique ID are sent to the device, allowing it to join the visiting base station. *'Encryption Secret'* and new Unique ID are used to encrypt and authenticate messages during transaction approval using non-interactive ZKP. The device join procedure is shown in Fig. 3.

4.6. Device reputation updation

The peer nodes verify device details and authenticate the block in a conventional blockchain framework. Upon successful authentication, a reward in the form of bitcoins or device resources is allocated to the device. However, this procedure may allow malicious nodes to reject blocks to delay the verification process and deplete other nodes of their battery resources required to verify the blocks.

To overcome these block verification limitations, we propose using non-interactive ZKP as the consensus algorithm, eliminating the requirement of peer nodes for block verification. To detect and eliminate malicious nodes, we introduce reputation to determine device integrity. In our design, we set reputation on a scale of *1* to *10*, where *1* represents a device with a low reputation and *10* represents a device with a high reputation. We set *4* to be the minimum device reputation. Devices with a reputation less than *4* are classified as malicious and transactions initiated by them are not verified. Each device is assigned an initial reputation by the manufacturer as defined by the data server. Upon each successful block approval, IPFS updates the device's reputation by one and sends it to that particular device. Devices update their reputation when in the *'Active state'*. If transactions are rejected due to an error in verifying device details, reputation is reduced by one.

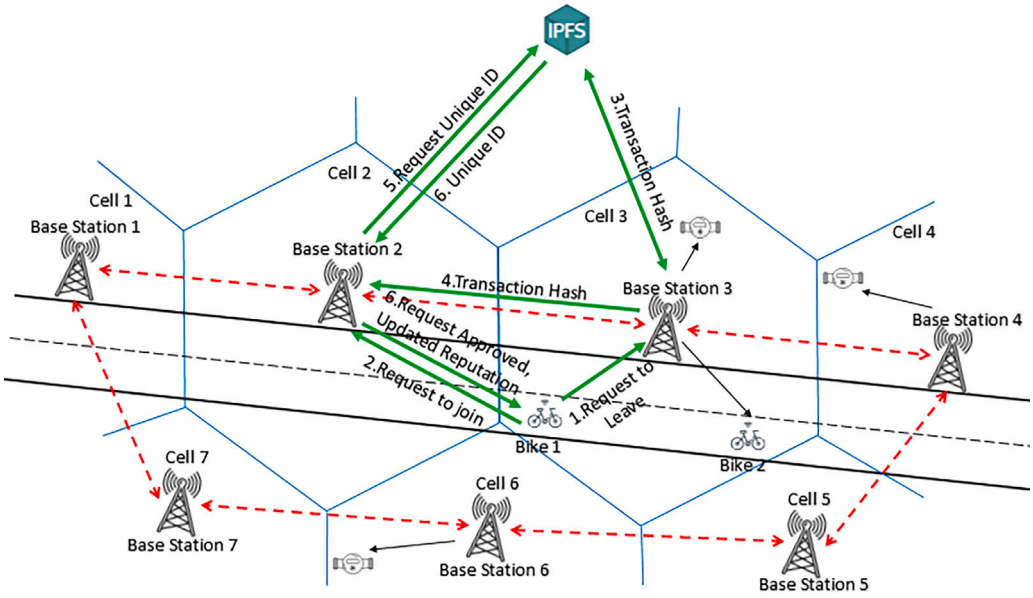

Fig. 4. Device authentication procedure.

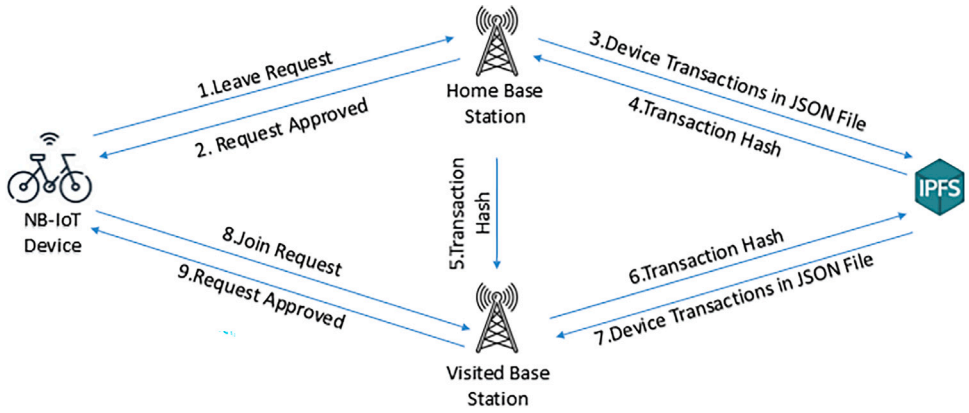

Fig. 5. Hash storage and retrieval from IPFS.

As IoT devices are mobile, they are required to authenticate themselves when joining a visited base station. The authentication process requires the visited base station to verify the reputation and unique ID of the node with the corresponding information retrieved from IPFS. The procedure for device authentication is shown in Fig. 4.

4.7. Hash storage and retrieval from IPFS

Base stations update the transaction hash in IPFS upon receiving a *'Leave Request'* from the devices. Base stations save the deployed smart contract and transactions executed by the devices in JSON format using the 'Save' option under the 'Transactions Recorded' section in Remix IDE. The file is then uploaded on IPFS. The resulting file hash is shared with the visited base station to retrieve and verify the device details from IPFS, as shown in Fig. 5. File hash is entered in the tab under 'Load file from IPFS' to retrieve the device transactions.

4.8. Transaction approval using non-interactive zero-knowledge proof

In non-interactive ZKP, base stations solve a computationally intensive mathematical challenge sent by the devices to authenticate the devices and approve transactions. However, this process would require base stations to have a high computational capability, resulting in block-approval delay. Hence, we modify the challenge–response mechanism in non-interactive ZKP to have low computational complexity and reduce the number of interactions between the device and base stations to avoid the probability of phishing attacks.

Devices generate a HMAC using SHA256 with a timestamp, nonce, unique ID, reputation, message and Encryption Secret as shown in Fig. 6. The resulting hash is sent to the base station for block approval along with Timestamp, Nonce, Unique ID, Reputation, message to be sent and hash. Upon receiving the request, the base station generates a verification hash (v_{hash}) using the device details along with the device's *'Encryption Secret'* generated and stored at the time of *'Device Join'* procedure. The resulting v_{hash} is compared with the hash sent by the device. On successful verification, the transaction is appended to the base station's merkle tree and the resulting block hash is shared with the device as shown in Algorithm 1. As the smart contract is designed to permit only devices to generate HMAC and initiate transactions and only base stations to approve them, fake block generation, delayed block approval and device resource depletion are eliminated. Since base stations only possess the *'Encryption Secret'* and can view the transaction requests alone, impersonation and phishing attacks are prevented.

The mathematical equation describing the hash generation process in the modified non-interactive ZKP is given in Eq. (5).

$$hash = SHA256(T, N, UniqueID, r, Message, e) \tag{5}$$

Algorithm 1 Device Smart Contract

Define *Device = [Unique ID, Device Address, Reputation, Linear hash chain]*
Define *Block = [Message, Nonce, Unique ID, Timestamp, Block Address]*
Function: AddDevice (Only Base Station)
Verify *Device(Unique ID, Address, Reputation, Latest Block Hash)*
if (*Reputation* < *4*) **then**
    Classify Device as malicious
    Break
end if
if (*Device Verified*) **then**
    Add Device to Blockchain
    if (*Reputation* < *10*) **then**
        Reputation = Reputation + 1
    end if
else
    Reputation = Reputation - 1
end if
Function: Add Block (Only Base Station)
if ($hash\_flag = 1$) **then**
    Base Station[Merkle root] += Block(Message, Nonce, Unique ID, Timestamp, Block Address)
    Base Station(Device[Linear hash chain]) += Block(Message, Nonce, Unique ID, Timestamp, Block Address)
end if
Device[Linear hash chain] ← *Base Station[Transaction hash of approved Block]*
Function: Construct Non-interactive Zero-Knowledge Proof (Only Device)
hash = SHA256(Timestamp, Nonce, Unique ID, Reputation, Message, Encryption secret)
Base Station ← *Device[Request to approve block, Timestamp, Nonce, Unique ID, Reputation, Message, hash]*
Function: Verify Non-interactive Zero-Knowledge Proof (Only Base Station)
v_{hash} = *SHA256(Timestamp, Nonce, Unique ID, Message, Reputation stored in Base Station, Encryption Secret stored in Base Station)*
if (v_{hash} == *hash*) **then**
    $hash\_flag = 1$
end if

Algorithm 2 Base Station Smart Contract

Define *Device = [Unique ID, Device Address, Reputation, Linear hash chain]*
Define *Addresses of Base Stations*
Function: SendHash (Only Base Station)
if *Device[Leave Request] received* **then**
    Store *Device[Transaction and Smart Contract]* in *JSON file*
    Upload file in IPFS
    Transaction_hash ← *IPFS*
    Visited Base Station ← *Transaction_hash*
end if
Function: GetHash (Only Base Station)
if *Device[Join Request] received* **then**
    Transaction_hash ← *Home Base Station*
    Request JSON file from *IPFS* using *Device[Transaction_hash]*
    Deploy and *Run Transactions* using *Device[Transaction_hash]*
end if

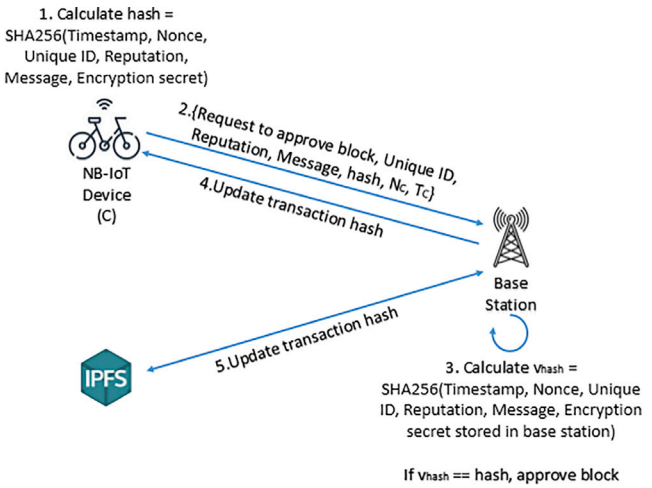

Fig. 6. Transaction approval using zero-knowledge proof.

The *'hash'* thus generated is transmitted to the base station along with Unique ID, message, reputation, nonce (N) and timestamp (T) to verify the device identity and approve the block. The base station calculates the verification hash (v_{hash}) using the Eq. (6) and approves the block only if v_{hash} = hash.

$$v_{hash} = SHA256(T, N, UniqueID, r', Message, e') \tag{6}$$

where,
T = Timestamp,
N = Nonce,
r = Reputation sent by the device in the block verification request,
r' = Reputation stored at the base station,
e = Encryption secret sent by the device in the block verification request,
e' = Encryption secret stored at the base station

4.9. Computational complexity

In this sub-section, we compare the performance of the proposed approach with that of conventional blockchain architecture in terms of memory and execution time using complexity equations. In the proposed hybrid blockchain architecture, devices store the generated hashes along with the previous hashes as a linear hash chain upon successful transaction verification. The equations to calculate the hash of individual blocks are shown in Eq. (7), (8) and (9).

$$\text{Hash of Block}_1\ (H_1) = h(\text{Genesis Block},\ \text{Message}_1) \tag{7}$$

$$\text{Hash of Block}_2\ (H_2) = h(block_1, \text{Message}_2) \tag{8}$$

$$\text{Hash of Block}_n\ (H_n) = h(block_{n-1}, \text{Message}_n) \tag{9}$$

Intermediary hashes are calculated by repeatedly hashing the previous block hash until the merkle tree is obtained as shown in Eq. (10).

$$\therefore \text{Merkle tree} = h(h(...h(h(H_1)))) \tag{10}$$

This design results in the number of hashes in each linear hash chain to be much lower than the total number of hashes in the merkle tree of the same height. Therefore, the memory requirement for checking the authenticity of transactions reduce from $O(2^n)$ (*'n'* number of transactions in merkle tree) to $O(m)$ (*'m'* number of hashes in a linear hash chain).

Base stations being central nodes, store the merkle tree related to the transactions in their particular cells. Base stations store the merkle

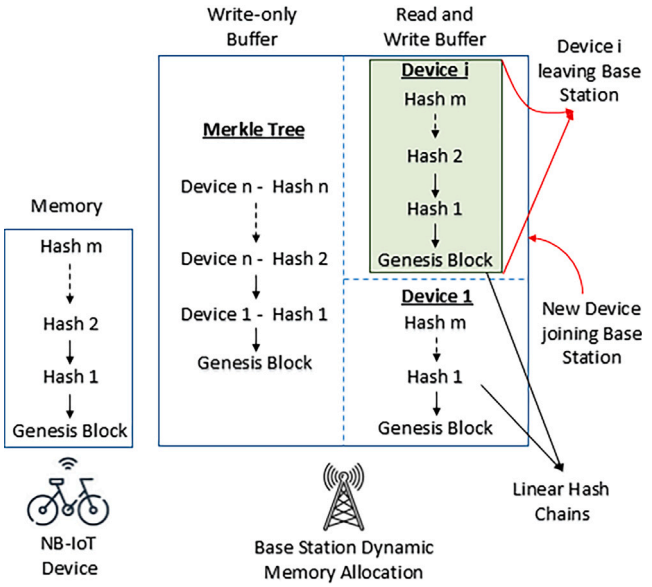

Fig. 7. Memory partition in base stations.

root and dynamically allocate the rest of the memory using *'Next-fit memory allocation'* algorithm (Puaut, 2002) to store hash chains and information, such as unique ID, Encryption Secret etc., of devices within its range. *'Next-fit memory allocation'* algorithm stores the merkle tree using a significant part of the memory. It then searches for the remaining memory and allocates it equally to all the devices present in the range of the base station. The part of the memory containing the merkle tree is *'write only'* where base stations append the approved hashes but do not traverse through them during device verification. In contrast, the dynamically allocated portions are *'read and write'*. Base stations write the device hashes in the partitions assigned explicitly to them and verify them during *'Device Join'*.

When base stations receive a *'Leave Request'* from a device, the hash chains stored in the dynamically allocated memory are converted to a JSON file and sent to the IPFS to be accessible by the visited base station. This architecture eliminates the base station checking the merkle tree during *'Leave Request'* and *'Device Join'*, thus reducing the execution time from $T(\log_2 n)$ to $T(m)$. $T(\log_2 n)$ is the time taken to verify *'n'* hashes in merkle tree and $T(m)$ represents the time taken to check *'m'* individual device transactions in a linear hash chain, where, $m << n$. While executing transaction verification, base stations only check the validity of the latest device hash as a part of the consensus protocol, thus reducing time complexity to $T(1)$.

The proposed framework enhances blockchain scalability by allocating new devices and storing more individual device transactions. Fig. 7 illustrates the memory partition in base stations. Mathematical equations supporting memory and time complexity computations are derived below.

4.9.1. Conventional blockchain

Memory complexity:. To derive memory complexity in the case of a conventional blockchain, let us assume that the merkle tree is a perfect binary tree (Garewal, 2020). Let the number of data blocks linked to the leaf nodes be *'n'*. The resulting number of tree nodes is derived using complexity equations (Fu et al., 2006) are given by the following equations.

$$Total\ number\ of\ nodes = n + \frac{n}{2} + \frac{n}{4} + \cdots + 1 \tag{11}$$

Since each block has a branching factor of *2*, the total number of blocks at each level can be represented by 2^k. Hence, we replace n with 2^k in Eq. (11) to get,

$$\text{Total number of nodes} = 1 + 2 + 2^2 + \cdots + 2^k \tag{12}$$

$$\therefore \text{Total number of nodes} = \frac{2^k - 1}{2 - 1} = 2^k - 1 \tag{13}$$

In total, number of nodes along with the data blocks sum up to $2^k + 2^k - 1 = 2^{k+1} - 1$, resulting in a complexity of $O(2^n)$.

Time complexity:. To calculate the time complexity of an *'n'* node perfect merkle tree, we start with computing the time complexity required to traverse through each node level (Fu et al., 2006), as shown below.

$$1 - node\ binary\ tree = log_2(1) + 1 = 1\ (time\ unit) \tag{14}$$

$$2 - node\ binary\ tree = log_2(2) + 1 = 2\ (time\ units) \tag{15}$$

$$4 - node\ binary\ tree = log_2(4) + 1 = 3\ (time\ units) \tag{16}$$

$$\therefore For\ n - node\ binary\ tree = [log_2(n) + 1]\ (time\ units) \tag{17}$$

Hence, time complexity for a conventional blockchain is given by $T(\log_2 n)$.

4.9.2. Proposed framework

Memory complexity:. In our proposed blockchain framework, base stations store, traverse and retrieve device hashes in the memory partitions allocated to specific devices as a linear chain. Assuming there are *'m'* leaf nodes, the total number of nodes will be,

$$Total\ number\ of\ nodes = 1 + 1 + \cdots m\ times = m \tag{18}$$

Since, the memory complexity to verify each hash is *1*, the resulting computational complexity is $O(m)$.

Time complexity:. Hash chains of individual devices stored in dynamically allocated memory in base stations are structured to contain one hash at each level to form linear hash chains. Hence, the number of nodes in individual hash chains (m) is much lower than those present in merkle tree (n) of the same height, i.e., $m << n$. The time complexity to traverse through each transaction hash (node) in hash chains is derived as follows.

$$Traversal\ time\ for\ 1\ node = 1\ (time\ unit) \tag{19}$$

$$Traversal\ time\ for\ 2\ nodes = 2\ (time\ units) \tag{20}$$

Therefore, traversal time T for *m nodes = time(1st node) + time(2nd node) + ... + time(mth node) = 'm' (time units)*. Therefore, time complexity for the proposed architecture is $T(m)$.

In the *'Device Join'* algorithm, the base station verifies all the transactions stored in the JSON file retrieved from IPFS. Hence, time complexity amounts to $T(m)$. However, during transaction verification, the base station checks the validity of the latest device transaction as a part of the non-interactive ZKP consensus. Thus, time complexity reduces to $T(1)$.

5. Experimental prototype and evaluation

In this section, we implement and evaluate the performance of NB-IoT connected using a blockchain-based framework. Performance is measured in terms of execution time, energy and memory consumption on Remix IDE in Ethereum and Raspberry Pi 4 platforms. A brief description of performance metrics is as follows.

Power - Power on Remix IDE was measured by observing CPU power consumption using Powerstat Anon (2015b). Powerstat is a CPU analyser in Linux OS that displays power consumption. While implementing NB-IoT on a Raspberry Pi 4, K2901 A Source Unit (Anon,

Table 2
Device Specifications.

Parameters	Value
Bandwidth	180 kHz
Battery Specifications	1000 mAh, 3.6 V (Anon, 2019a)
Tx Power	36 W
Rx Power	24 W
Tx Gain	70
Rx Gain	40
Tx Current	21.7 μA
Transmission Mode	Frequency-Division Duplexing
Distance between Tx and Rx	100 m
Packet Size - Uplink	2–125 B (Anon, 2017c)
Packet Size - Downlink	2–85 B
Peak Data Rate - Uplink	66 kbps (Anon, 2019d)
Peak Data Rate - Downlink	26 kbps

2011) was used to power the device at 5.45 V and 3 A. Values were recorded by observing power variations during run-time.

Memory - Memory on Remix IDE was measured using s-tui (Anon, 2022b), a CPU analyser in Ubuntu displaying core temperature, operating frequency and memory consumption. In Raspberry Pi implementation, 'Task Manager' was utilised to record the memory consumed.

Execution Time - Execution time in Remix IDE was measured using a millisecond clock as solidity does not support the execution time measurement function. In Raspberry Pi 4, execution time was measured by computing the difference between the start and stop time, as shown in Eq. (21).

$$T_{execution} = T_{stop} - T_{start} \tag{21}$$

5.1. Implementation on remix IDE

In this sub-section, we propose implementing smart contracts in Remix IDE on Ubuntu 19.06. We implement our model as two smart contracts: Device Smart Contract and Base Station Smart Contract. A detailed explanation of smart contracts is given below.

5.1.1. Device smart contract

Device smart contract deals with verifying and adding a device to the base station, block generation, block approval using non-interactive ZKP, incrementing device count upon successful device addition and viewing device details based on Unique ID as shown in Algorithm 1. The Device smart contract is defined to permit devices to access block generation and base stations to access the rest of the functions to limit the transparency of block details. The smart contract is designed based on the assumption that NB-IoT, being resourced constrained, is vulnerable to attacks and base stations being directly controlled by the service provider are immune to threats.

Verification and addition of devices:. Devices are designed as a structure containing the device address, reputation, unique ID and hashes of the transactions initiated by it connected as a linear chain. When a device requests to join a visited base station, it sends an authentication request and a *'Join Request'* to the Authentication server and base station, respectively, as shown in Fig. 3. Upon authentication, the base station executes the smart contract to verify the device details and reputation. Upon successful verification, devices are allowed to join the blockchain.

Block generation and verification:. Devices construct blocks to store messages or observations in IPFS. A block is designed to contain the message to be sent, nonce, Unique ID and timestamp. Non-interactive ZKP used as the consensus algorithm calculates a hash of the block parameters along with the *'Encryption Secret'* shared by the base station during *'Device Join'*. This hash is sent to the base station as a part of the block approval request. The base stations calculate v_{hash} using the device details received along with the device's *'Encryption Secret'* stored in the dynamically allocated memory. Upon successful verification, the transaction hash is appended to the merkle tree and the device's linear hash chain stored in dynamically allocated memory.

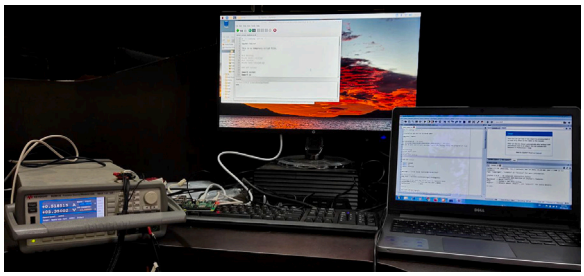

Fig. 8. Hardware setup.

View device details using unique ID:. As we propose to use a hybrid blockchain, only base stations are authorised to view device details. The smart contract ensures a device is callable by the Unique ID shared during the *'Device Join'* procedure. This design ensures device secrecy as Unique ID is known only to the device and the base station.

5.1.2. Base station smart contract

Base stations communicate with each other to upload and retrieve transaction hash upon receiving a *'Leave Request'* from the devices. Only base stations are authorised to execute the Base Station smart contract as shown in Algorithm 2 to avoid misuse of device transaction hashes. When base stations receive a *'Leave Request'*, device transactions are stored in a JSON format using *'SendHash'* function. The file is uploaded in IPFS and the resulting transaction hash is communicated with the visited base station for verification. The IPFS hash is downloaded by visited base station using *'GetHash'* function as shown in Fig. 3.

5.2. Implementation on Raspberry Pi 4

In this sub-section, we implement the *'Unique ID generation'* and *'Device Join'* algorithms discussed under Sections 4.4 and 4.5, respectively, as UDP Client–server programs on Raspberry Pi 4. Device parameters required to simulate NB-IoT are specified in Table 2. Raspberry Pi 4 and laptop were programmed as client and server, respectively and vice versa. Performance of each device, home and visited base stations, data servers and authentication servers in terms of power, memory and execution time was measured. A screenshot of the implementation setup is shown in Fig. 8.

5.3. Verification and validation using scyther

In this section, we formally verify the protocols implemented in the proposed blockchain framework using Scyther. Scyther is a formal verification tool to detect individual attacks and attack patterns across multiple protocols operating in parallel. It provides a user interface through Python code infrastructure. GUI in Scyther is written in *Python 3* using *wxPython GUI toolkit* and the backend is written in *C*. Protocols implemented in Scyther are written in SPDL (Security Protocol Description Language).

Scyther works based on a pattern refinement algorithm to provide a concise representation of infinite sets of traces to analyse various attacks and protocol behaviours. Scyther checks the correctness of protocols and displays a proof tree with the help of the backend in case of attacks. The tool operates for an unbounded number of sessions. However, this feature does not guarantee unbounded correctness. In such cases, Scyther displays 'No attacks within bounds'. The tool displays 'No attacks' if no specific attacks or attack patterns are found. Additionally,

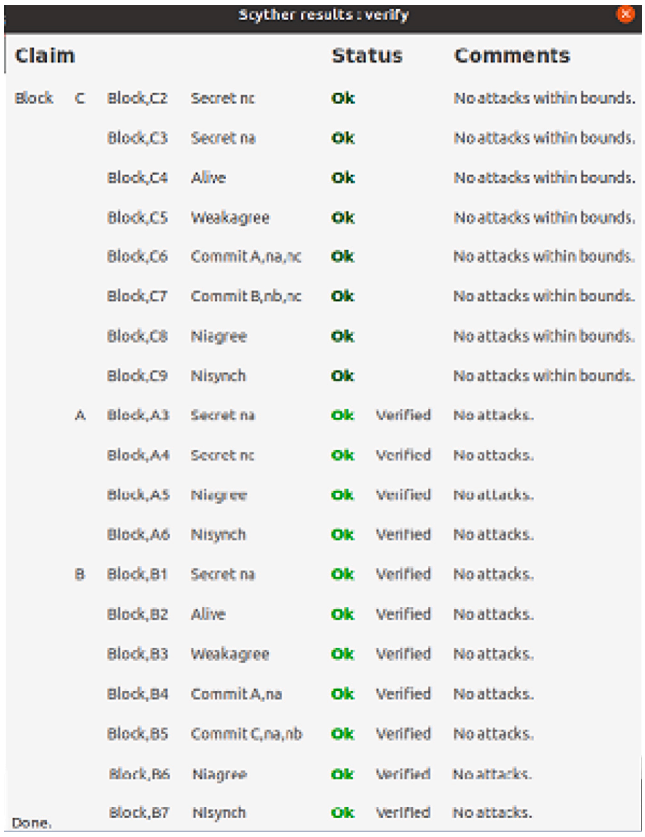

Fig. 9. Scyther verification report for device ID generation.

Scyther analyses multiple sub-protocols running in parallel. The tool verifies claims such as *Weakagree*, *Commit*, *Secret*, *Alive*, *Niagree* and *Nisync* described below.

- Niagree - Niagree shows that in a one-way authentication with a non-injective agreement, the sender believes that when the protocol is executed, the receiver is the responder.
- Nisynch - Nisynch indicates that the protocol operates with data consistency on all execution steps.
- Alive - Alive guarantees that after the protocol execution, the receiver is active and proves that he initiates the expected response.
- Weakagree - Weakagree ensures that all the participating entities are executing the protocol.
- Commit - Commit ensures that the protocol has asserted an effective claim.
- Secret - Secret ensures that the parameters embedded in protocols have been kept secret from eavesdroppers.

In this paper, we implement authentication protocols such as *'Unique ID Generation'* and *'Device Join'* operating between devices, authentication and data servers on a wireless medium outside the blockchain framework on Scyther to validate them against known attack patterns formally. Verification reports (Abraham and Jevitha, 2019) thus generated by Scyther are shown in the screenshots Fig. 9 and 10.

Other protocols, such as *'AddDevice'*, *'AddBlock'*, *'SendHash'* etc., are implemented as a part of the blockchain framework between devices and base stations on Remix IDE. The security of these protocols depends on the blockchain transparency and consensus algorithm implemented. Hence, they are not explicitly verified using Scyther.

5.4. Security analysis and mitigation strategies

In this section, we analyse the proposed blockchain framework's security features and mitigation strategies.

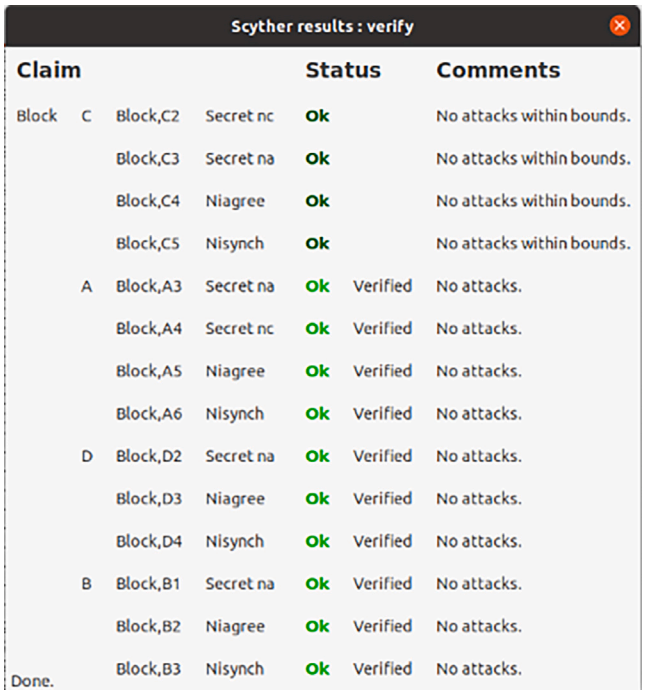

Fig. 10. Scyther verification report for device join.

5.4.1. Device anonymity

IoT devices are assigned a *'Device Address'* as a permanent ID by the Data Server when they join the blockchain for the first time. To ensure device anonymity, the Data Server generates a pseudonym in the form of a unique ID using *'python uuid library'* every time a device enters a new cell. The 'Seed' to generate the unique ID is *Old Unique ID* || *Encryption Secret* || *Nonce*. Encryption Secret is known only to the device, visiting base station and data server. During the *'Device Join Process'*, the Encryption Secret is encrypted by the device's private key and visiting base station's public key and then shared with the device. Hence, the attacker eavesdropping on the channel cannot decrypt it. Additionally, as the Data Server randomly generates the *Nonce*, it cannot be predicted by the attacker, thus, preserving device anonymity.

5.4.2. Message authenticity and integrity

Message authenticity is required to ensure that the message originated from the sender. This design requirement is achieved by using *Nonce* (N_C) and *Device ID*, i.e., the permanent ID of the devices as shown in authentication protocols of *'Unique ID Generation'* and *'Device Join Procedure'*. In these handshake protocols, the device sends a *'Join Request'* to the visiting base station along with the Old Unique ID, reputation, timestamp, nonce and a hash of Device ID and nonce as shown *[Join Request, Reputation, Old Unique ID, h(Device Address,* N_C*),* N_C*,* T_C*,* T_A*, [h(*N_A*,* T_A*)]*pk_B*]*pk_A*]*. As *Device ID* is stored only by the device and the Data Server, it is not known to other blockchain entities. Hence, *h(Device ID,* N_C*)* is transmitted to the receiver by the data server to validate *'Join Request'*. Therefore, an attacker will not be able to retrieve *Device ID* and calculate *h(Device ID,* N_C*)* by eavesdropping on the channel, thus maintaining message authenticity.

Message integrity ensures that the message has not been modified in transit. This can be achieved using *h(*N_A*,* T_A*)*. Receivers maintain a list of previous nonces used for authentication. When the receivers receive the *'Join Request'*, they check the nonce and calculate *h(*N_A*,* T_A*)* to ensure a valid and unique value. If the *h(*N_A*,* T_A*)* is invalid or the nonce has been used, the recipient rejects the message as potentially fraudulent or invalid.

5.4.3. Man-in-the-middle attack

In Man-in-the-Middle Attack, attackers can intercept and alter the message. To prevent the attack during device authentication, the device sends a *Request for Nonce* to the Authentication Server, which in turn responds with $[N_A, T_A, h(N_A, N_C, T_A)]$. N_A and T_A are the nonce and timestamp of the Authentication Server. N_C refers to the nonce of the device. $h(N_A, N_C, T_A)$ ensures that *Nonce* (N_A) to be sent to visiting base station by the device for verification has not been tampered. Meanwhile, the Authentication Server constructs $[[Leave\ Request, h(Device\ Address, N_C), Old\ Unique\ ID, N_A, N_C)]sk_B]pk_A$ to send the N_C and N_A to visiting base station to verify the identity of the device.

Upon receiving N_A and T_A, the device calculates $h(N_A, T_A)$ and sends $[Join\ Request, Reputation, Old\ Unique\ ID, h(Device\ Address, N_C), N_C, T_C, T_A, [h(N_A, T_A)]pk_B]pk_A]$ to the visited base station. The visited base station then calculates $h(N_A, T_A)$ using T_A and N_A received from the device and Authentication Server, respectively, to verify the authenticity of the device request.

5.4.4. Impersonation attack

Consider the case where an attacker can impersonate a prospective visiting base station during handover by publishing a fake address to the device. The devices send $[[Request\ to\ Leave, N_C, T_C, Old\ Unique\ ID, Visiting\ Base\ Station\ Address]sk_D]pk_C$ to the home base station to verify the visiting base station and the *'Leave Request'*. The home base station verifies the identity of the visiting base station from the list of blockchain participants. It sends the hash of the JSON file containing the device transactions to the IPFS and the visiting base station. The visiting base station verifies the authenticity of the hash and home base station with the help of IPFS and approves $[Join\ Request, Reputation, Old\ Unique\ ID, h(Device\ Address, N_C), N_C, T_C, T_A, [h(N_A, T_A)]pk_B]pk_A]$ from the device. This procedure eliminates the joining of devices to the malicious base stations and ensures the authenticity of the device and the visiting base station identity.

Suppose an attacker tries to impersonate a device and sends a 'Join Request' to the visiting base station. He cannot construct $h(Device\ Address, N_C)$ to prove his identity to the Authentication Server and the visiting base station since he does not possess *Device Address*. Thus impersonation attacks are prevented in our framework.

5.4.5. Blockchain fork

Blockchain fork occurs when some nodes do not upgrade their software intentionally or due to a lack of required resources per the updated block verification rules. In our proposed framework, the smart contract permits devices only to initiate transactions and base stations to verify them. Hence, an update in software and transaction verification rules will not require the devices to update software, thus eliminating blockchain forks.

5.4.6. Replay attacks

Since NB-IoT operates on low bandwidth, attackers can interfere and replay messages or a part of the message captured from the previous authentication protocol. To avoid such attacks, we design our protocols to include verification of nonces and timestamps. Consider the protocol used during Unique ID generation sent by the Authentication server to the device, $[N_A, T_A, h(N_A, N_C), [h(N_A, T_A)]sk_C]pk_A$. N_A and T_A are included as a part of authentication along with $h(N_A, T_A)$. Even if the attacker replays N_A and T_A, these values will not match the hash value, making the protocol resilient to replay attacks.

5.4.7. Distributed Denial of Service (DDoS)

Attackers launch DDoS to cause transaction flooding in the blockchain to delay the verification of benign transactions and cause network congestion, bloated ledger and node failures. To avoid such adversaries, we design authentication protocols to include nonces and timestamps. When attackers initiate numerous transactions, the receiver ignores them when the nonce verification fails and *h(N, T)* mismatch occurs. To prevent these attacks in the future, base stations blacklist the attackers and ignore the verification of transactions generated by them.

5.4.8. Sybil attack

In Sybil attacks, attackers manipulate a single node to operate under multiple fake identities to gain a majority and launch a 51% attack. To overcome sybil attacks, each device is given a Unique ID as a pseudonym when it enters visiting base station by the Data Server and is shared with the device, the visiting base station and the Authentication Server. Hence, multiple device identities are prevented within a particular base station.

When devices move from one cell to another, they send a *'Request for Nonce'* to the Authentication Server along with the existing Unique ID shown as $[[Request\ for\ Nonce, Old\ Unique\ ID, N_C, T_C]sk_A]pk_C$. Upon device authentication at the visiting base station, the home base station transmits the hash of the JSON file containing the hashes of the transaction initiated by the device through IPFS to the visiting station and removes a record of the file in its memory. This design prevents Sybil attacks across different base stations.

5.4.9. Resource depletion attacks

Resources in the blockchain framework can be depleted in three ways described below.

Malicious nodes:. Attackers can generate blocks with zero transaction value to drain base stations and other devices of resources in terms of memory and execution time. However, some transactions initiated by benign devices may get rejected due to the presence of information altered by noise signals on the wireless medium. To efficiently classify such nodes, base stations use reputation on a scale of *1* to *10* as a measure of trust. Reputation is increased by one if transactions are verified and decreased by one otherwise. If reputation is less than *4*, the device is classified as malicious and transactions initiated by them are not verified.

Fake block generation:. Attackers generate fake blocks to either blacklist benign nodes or to cause double spending to re-use the same digital currency and avail multiple services. Fake blocks aim to deplete resources on devices and base stations. To avoid such attacks, we eliminate group consensus and implement non-interactive ZKP as the consensus algorithm where the transaction initiated by a device is visible only to the base station where the device is present. This architecture limits transaction and device transparency to other devices. Thus fake block generation by peer nodes is limited.

Our design assumes that attackers can only manipulate devices and that base stations under service providers' supervision are attack-free. Additionally, base stations are only permitted to approve transactions and not initiate them by the smart contract employed in the hybrid blockchain framework. Hence, fake blocks are not generated by base stations.

Delayed block verification:. In conventional blockchains using decentralised consensus algorithms, malicious nodes can mine a block and publish the hash after an extended period to deplete resources in other nodes trying to find the hash less than the hash strength. Our proposed architecture mitigates this attack since non-interactive ZKP is used as the consensus algorithm and blockchain is implemented as a hybrid structure. This design eliminates group consensus and makes the transaction verification available only to the base station where the device is present.

5.5. Performance evaluation

In this section, we discuss the performance of smart contracts and device protocols implemented using Raspberry Pi 4 and Remix IDE. Further, we compare our results with performance values in some existing works.

Considering the scenario of NB-IoT, PoW (Anon, 2022a) may not be practically implementable due to constrained bandwidth and battery

Table 3
Performance values of device smart contract.

Parameter	Execution Time (s)	Memory (MB)	Power (W)
Add Device	1.69	6.2	0.58
Block Formation	1.53	5.6	0.55
Zero-Knowledge Proof	0.96	5.8	0.19
Retrieve Block	0.96	4.7	0.2
Retrieve Device Details	1.06	4.3	0.38
Read Device Count	0.88	3.1	0.17

Table 4
Performance values of base station smart contract.

Parameter	Execution Time (s)	Memory (MB)	Power (W)
SendHash	1.12	3.8	0.12
GetHash	1.12	3.4	0.21

Table 5
Performance values of unique ID generation.

Parameter	Execution Time (s)	Memory (MB)	Power (W)
IoT Device	0.235	0.3	1.30
Authentication Server	2.678	0.3	1.04
Data Server	5.159	0.35	1.02

Table 6
Performance values of device join procedure.

Parameter	Execution Time (s)	Memory (MB)	Power (W)
IoT Device	0.213	0.2	1.45
Authentication Server	1.836	0.2	0.84
Home Base Station	6.931	0.1	0.42
Visiting Base Station	7.932	0.05	0.18
Dynamic Memory Allocation at Base Station	1.39	1.0	0.02

requirements. Hence, non-interactive ZKP is used as the consensus algorithm. We measure performance for various smart contract operations such as *Add Device*, *Block Formation*, *SendHash*, *GetHash* and *Retrieve Device Details* and *Retrieve Block*. Values thus measured are recorded in Tables 3 and 4. Since *Retrieve Device Details* and *Retrieve Block* require the devices to enter *Unique ID* and *block hash* respectively and view details, low power and execution time overhead is incurred. Hence, the performance metrics of our approach were observed to be lower than others.

Similarly, we observe that *Read Device Count* requiring the user to query the variable counting the devices in the blockchain requires the lowest power, memory and execution time of 0.17 W, 3.1MB and 0.88s respectively. In contrast, the function, *Add Device* executed by the base stations verifies the device details and reputation. It cross-checks the latest block hash with the value stored in the dynamically allocated memory. Hence, the function requires the highest power, memory and execution time of 0.58 W, 6.2MB and 1.69s respectively. Other functions such as *Block Formation*, *Zero-Knowledge Proof*, *SendHash* and *GetHash* implementing fewer functions such as block formation, block verification and upload and retrieval of JSON file hash from IPFS require an average of 0.5 W, 5MB and 1.5s to execute.

As explained in Section 4.9, base stations dynamically allocate memory when devices join and leave the current cells. We observe that the memory required for this process is 1MB, much less than the memory-storage requirements in optimisation algorithms proposed in Wang et al. (2020), Qi et al. (2021). However, the time taken to read and write hash chains and device information into the memory buffer was 1.39s which is much higher than previous works. This increase in value is observed due to continuous write and flushing operations in memory buffers as devices enter and exit cells.

Further, we evaluate the performance of *'Unique ID Generation'* and *'Device Join'* algorithms implemented on Raspberry Pi 4 using a UDP client–server program on a Wi-Fi network. Values thus measured for the algorithms are recorded in Tables 5 and 6, respectively. From Table 5, we observe that the Authentication Server and Data Server require 2.678s and 5.159s to execute the smart contract. In practical scenarios, base stations receive numerous requests for authentication from devices requiring to join and leave cells. Hence, *Authentication Server* and *Data Server* require higher execution time to validate these requests. However, the memory and power required to process individual requests were low. Similar behaviour was observed in *'Home Base Station'* and *'Visiting Base Station'* shown in Table 6. As various authentication protocols are executed for checking message integrity, hash upload and retrieval from IPFS, an average execution time of 7s was recorded, observed to be higher compared to other functions executed. However, the memory and power to process each request were lower than those of the Device and Authentication servers.

The performance of *Device ID Generation* and *Device Join* algorithms were studied in terms of congestion, interference and throughput as shown in Fig. 11. Network congestion is measured in terms of Round Trip Time (RTT), interference in terms of packet drop rate and throughput in terms of the number of packets transmitted per second in the UDP channel for varying packet drop rates of 0, 0.25, 0.5, 0.75 and 1, respectively. The X-axis of the figure is set to the number of authentication requests sent simultaneously by a single NB-IoT node to the *'Authentication Server'* and *'Data Server'* in one *'Active'* state.

In Fig. 11a, we observe that RTT rises for increasing packet drop rate and increasing number of authentication requests. For example, if authentication requests are equal to eight, RTT is approximately 0.028s, 0.054s, 0.075s and 0.182s for drop rates of 0, 0.25, 0.5 and 0.75, respectively. For a drop rate of 0.75, the NB-IoT node achieves the peak RTT of 10000s for authentication requests totalling nine and ten, where all packets are dropped. Similar behaviour is observed for a drop rate of 1 for varying numbers of authentication requests simulated.

In Fig. 11b, we plot the interference by measuring the packet drop rate for varying numbers of authentication requests. We observe that as the number of authentication requests increases, the packet drop rate at the receiver reduces. This behaviour is observed because the NB-IoT device remains in *'Active'* state longer to process the increased authentication requests. Hence it receives the lost packets re-transmitted from the sender. For ten authentication requests sent, our model has a drop rate of 0.0209, approximately equal to zero. From Fig. 11a, we infer that for a packet drop rate of zero, an RTT of 0.029s is observed, which is within the range of 0 to 100ms (Anon, 2017e). Hence, our model provides good communication.

We study the NB-IoT node's throughput for varying authentication requests and drop rates. We see that throughput decreases continuously as the bandwidth-constrained nodes require higher execution time and memory to process the increased number of authentication requests. For a drop rate of 0, throughput is constant at 34 packets/s for varying numbers of authentication requests. Throughput progressively decreases for increasing drop rates and reaches 0 packets/s for a drop rate of 0.75 and the number of authentication requests being nine and ten. These values indicate that all packets were lost in transit. Similarly, for a drop rate of 1, throughput is 0 packets/s for a varying number of authentication requests, indicating all packets are lost. Further, *Device ID Generation* and *Device Join* algorithms were formally tested using Scyther. The resulting evaluation reports show that our protocols are immune to known attacks.

5.5.1. Comparison with existing approaches

As seen in Table 3, the row labelled *'Zero-Knowledge Proof'* involves block verification using non-interactive ZKP. We observe that power

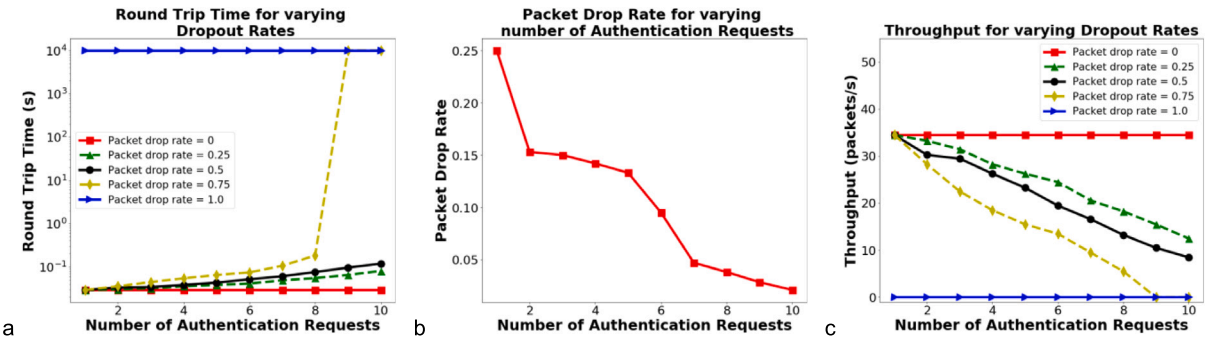

Fig. 11. Round trip time, packet drop rate and throughput of NB-IoT devices.

Table 7
Comparison of performance values of consensus algorithms.

Parameter	Execution Time (s)	Memory (MB)	Power (W)
Proof of Work	6.58	50	3.56
Proof of Stake	1.92	15	0.63
Proof of Authority	1.92	15	0.4
Byzantine Fault Tolerance	2.25	20	0.61
Zero-Knowledge Proof	0.96	5.8	0.19

Table 8
Performance values of frameworks in proposed approaches.

Proposed Approach	Execution Time (s)	Memory (MB)	Current (A)	Power (W)
Jiang et al. (2019)	3.80	–	–	–
Ile et al. (2019)	37.58	–	0.1744	–
Gao et al. (2018)	2	–	–	–
Javaid et al. (2020)	10	–	–	–
Wang et al. (2020)	0.001	100	–	–
Qi et al. (2021)	0.1	2	–	–
Our work	0.96	1	–	0.19

and execution time of 0.19 W and 0.96s were much lower than the values presented in most existing works. However, we observe that Wang et al. (2020) and Qi et al. (2021) require 0.001s and 0.1s, respectively, because these works focus on memory and time optimisation in a conventional blockchain framework and do not consider their applications in a resource-constrained NB-IoT scenario.

The power, memory and execution time required to authenticate the node request for varying consensus algorithms such as Proof of Work (PoW), Proof of Stake (PoS), Proof of Authority (PoA) and Byzantine Fault Tolerance (BFT) as recorded in Table 7 shows that PoW requiring to calculate the nonce for increasing difficulty in block mining consumes the highest power, memory and execution time of 3.56 W, 50MB and 6.58s respectively. BFT requiring the votes of two-thirds of the nodes to validate a block consumes the next highest power, memory and execution time. BFT is followed by PoS and PoA where the validator selected based on the stake held or the reputation has a more weighted vote in approving blocks. Hence, the performance values were observed to be in the same range. However, these values are much higher than those measured in non-interactive ZKP.

From Table 8, we infer Javaid et al. (2020) tested their model for high, medium-high and low arrival packet mining rates resulting in execution times of 0.1s, 1s and 10 s, respectively. Since NB-IoT has a low bandwidth, it supports lower packet arrival rates. Hence, we consider the case of *'Low arrival packet mining rate'* with an execution time of 10 s for our comparison. We observe that our framework consumes approximately 74.73% lower execution time than others. The model proposed by Ile et al. (2019) requires 0.1744 A to execute as shown in Table 8. In contrast, our model operating at 5.45 V requiring 0.19 W (Table 3) consumes 0.034 A, thus providing 80.50% increased energy efficiency.

6. Discussion

Blockchain architecture being distributed in nature makes the stored data vulnerable to security risks and regulatory issues discussed as follows.

6.1. Security risks

Some security risks in securely storing data are hacking, phishing and 51% attacks. In device hacking, attackers break into the device and extract sensitive information such as device ID and private key. Double-spending and zero-value attacks can be launched using these details to lure other nodes into sending cryptocurrency. In phishing attacks, individuals and organisations using blockchain are targeted to steal sensitive information. These attacks are executed in the form of fake websites, emails or messages that appear to be from a legitimate source. 51% attack is performed in a blockchain platform using consensus algorithms such as PoW, PoS etc., to control more than 50% of the hash rate in the network. Thus allowing the adversary to manipulate the blockchain by creating new blocks faster than the rest of the network, thereby controlling transaction authentication

In our proposed architecture, the smart contract permits devices to initiate transactions but does not permit them to approve. Since non-interactive ZKP is used as the consensus algorithm, the transparency and approval of transactions are limited to base stations, thereby eliminating the possibility of double spending and zero-value attacks. As non-interactive ZKP eliminates the requirement of peer nodes for block authentication, communication between devices is limited, thus, preventing phishing and 51% attacks.

6.2. Regulatory issues

The lack of a central point of control in blockchain architecture makes it challenging for governments to enforce laws and regulations, as they may need more authority to take action against violating entities. Additionally, as blockchain is in the initial stages of development, a well-defined set of rules governing the behaviour of nodes and data storage needs to be established.

In our proposed approach, we overcome these issues by defining the functions of devices and base stations in the smart contract. Devices are designed to initiate transactions, while base stations are permitted to verify device details, allow devices to join visiting base stations and approve transactions. If any attempts are made by the devices to impersonate peer nodes or generate zero-value transactions, their reputation is reduced by *1* by the base stations. If reputation is less than *4*, then the device is blacklisted and the base stations do not verify transactions initiated by them.

Each base station maintains individual merkle trees to store transactions initiated by devices present in their range. When devices move from one cell to another, base stations share the device transaction to the visited stations through IPFS as a JSON file, thus maintaining track of device activities. The service provider, in turn, controls base stations. IPFS holds the records of all the transactions initiated by the devices. This mechanism would facilitate the government to implement necessary regulations and take action against violating nodes.

6.3. Network interference and congestion

Interference and congestion in a blockchain network are caused due to increased transaction volume, a non-scalable blockchain model and attacks such as DoS, Man-In-the-Middle and 51% attacks. Network interference and congestion may result in slowed-down transaction processing, reduced blockchain scalability, centralised risks due to the concentration of mining power and transaction flooding.

In our proposed model, non-interactive ZKP is used as the consensus algorithm. Hence, it eliminates the requirement of peer nodes to authenticate the blocks, mining power centralisation and 51% attacks. As we store the verified block hashes of individual devices as a linear hash chain in dynamically allocated memory partitions in base stations, scalability to store hashes for an increased number of nodes is provided. As our design verifies the latest block hash instead of the entire Merkle tree, during the *Device Join* procedure, the volume of transaction verification in the network reduces, thereby increasing the transaction processing rate. Further as the authentication protocols in the *Device ID Generation* and *Device Join* procedures include nonce (N), timestamp (T) and $h(N, T)$ as explained in Section 5.4, DoS, Man-In-the-Middle and transaction flooding attacks are prevented.

7. Conclusion and future work

In this paper, we developed a hybrid blockchain framework to provide seamless network connectivity across cells in mobile NB-IoT applications. Base stations are interconnected through IPFS to provide mutual authentication and eliminate fake base station connections and the requirement of certificate transfer. IoT devices are designed to store transaction hashes as a linear chain instead of the entire merkle tree to reduce memory and computational complexity. Non-interactive ZKP is used as the consensus algorithm to provide scalability, reduce user and transaction transparency and eliminate group consensus to enable rapid block verification. Base station memory is partitioned to enhance scalability and reduce hash verification complexity. Further, we implemented the framework on Remix IDE and Raspberry Pi 4 and analysed energy-performance-security trade-offs with those of the existing works. The security of the proposed framework was formally validated using Scyther. We show that our framework consumed 80.50%, 74.73% and 50% lower power, execution time and memory, respectively, compared to existing approaches, thus, providing a lightweight alternative. We propose to extend this work to develop algorithms with effective memory management for varying NB-IoT performance requirements.

CRediT authorship contribution statement

Vamshi Sunku Mohan: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing – original draft, Writing – review & editing, Visualization. **Sriram Sankaran:** Conceptualization, Validation, Resources, Writing – review & editing, Visualization, Supervision, Project administration. **Priyadarsi Nanda:** Conceptualization, Formal analysis, Writing – review & editing, Visualization. **Krishnashree Achuthan:** Supervision, Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request

References

Abraham, Misha, Jevitha, Kp., 2019. Runtime verification and vulnerability testing of smart contracts. http://dx.doi.org/10.1007/978-981-13-9942-8_32.

Alkhateeb, A., Catal, C., Kar, G., Mishra, A., 2022. Hybrid blockchain platforms for the Internet of Things (IoT): A systematic literature review. Sensors 22 (4), 1304. http://dx.doi.org/10.3390/s22041304.

An, S., Routray, S.K., 2017. Issues and challenges in healthcare narrowband IoT. In: 2017 International Conference on Inventive Communication and Computational Technologies. ICICCT, pp. 486–489. http://dx.doi.org/10.1109/ICICCT.2017.7975247.

Anon, 2011. B2901A Precision Source/Measure Unit. https://www.keysight.com/in/en/product/B2901A/precision-source-measure-unit-1-ch-100fa-210v-3a-dc-10-5a-pulse.html.

Anon, 2012. Advantages and disadvantages of NB-IoT. https://www.rfwireless-world.com/Terminology/Advantages-and-Disadvantages-of-NB-IoT.html.

Anon, 2014. The Scyther Tool. https://people.cispa.io/cas.cremers/scyther/.

Anon, 2015a. IPFS. https://ipfs.io/.

Anon, 2015b. Powerstat. https://manpages.ubuntu.com/manpages/xenial/man8/powerstat.8.html.

Anon, 2017a. The mathematics behind blockchain. https://mathinvestor.org/2017/08/the-mathematics-behind-blockchain/.

Anon, 2017b. Narrowband IoT (NB-IoT). https://www.u-blox.com/en/narrowband-iot-nb-iot.

Anon, 2017c. NB-IoT - Data Rates and Latency. https://www.netmanias.com/en/post/blog/12609/iot-nb-iot/nb-iot-data-rates-and-latency.

Anon, 2017d. Remix IDE. https://remix-project.org/.

Anon, 2017e. What is RTT in networking? https://aws.amazon.com/what-is/rtt-in-networking/.

Anon, 2019a. Analyzing NB IoT and LoRaWAN Sensor Battery Life. https://tech-journal.semtech.com/analyzing-nb-iot-and-lorawan-sensor-battery-life.

Anon, 2019b. NB-IoT deployment guide to basic feature set requirements. https://www.gsma.com/iot/wp-content/uploads/2019/07/201906-GSMA-NB-IoT-Deployment-Guide-v3.pdf.

Anon, 2019c. Raspberry Pi 4. https://www.raspberrypi.com/products/raspberry-pi-4-model-b/.

Anon, 2019d. What is the difference in data throughput between LTE-M/NB-IoT and 3G or 4G? https://www.gsma.com/iot/resources/what-is-the-difference-in-data-throughput-between-lte-m-nb-iot-and-3g-or-4g/.

Anon, 2020. LoRaWAN vs NB-IoT: A Comparison Between IoT Trend-Setters. https://ubidots.com/blog/lorawan-vs-nb-iot/.

Anon, 2022a. Proof of work: An overview of PoW blockchains. https://komodoplatform.com/en/academy/proof-of-work/.

Anon, 2022b. The Stress Terminal UI: s-tui. https://github.com/amanusk/s-tui.

Anon, 2023a. Narrowband IoT (NB-IoT). https://www.thalesgroup.com/en/markets/digital-identity-and-security/iot/resources/innovation-technology/nb-iot.

Anon, 2023b. What are the 4 different types of blockchain technology??https://www.techtarget.com/searchcio/feature/What-are-the-4-different-types-of-blockchain-technology.

Aponte-Novoa, F.A., Orozco, A.L.S., Villanueva-Polanco, R., Wightman, P., 2021. The 51% attack on blockchains: A mining behavior study. IEEE Access 9, 140549–140564. http://dx.doi.org/10.1109/ACCESS.2021.3119291.

Cao, W., Wang, J., Zhao, Y., Huang, L., 2020. NBIoT optimization on massive devices access load control. In: 2020 International Conference on Intelligent Computing and Human-Computer Interaction. ICHCI, pp. 33–37. http://dx.doi.org/10.1109/ICHCI51889.2020.00015.

Dorri, A., Steger, M., Kanhere, S.S., Jurdak, R., 2017. BlockChain: A distributed solution to automotive security and privacy. IEEE Commun. Mag. 55, 119–125.

Du, Z., Qian, H.-f., Pang, X., 2023. PartitionChain: A scalable and reliable data storage strategy for permissioned blockchain. IEEE Trans. Knowl. Data Eng. http://dx.doi.org/10.1109/TKDE.2021.3136556.

Fu, FW., Niederreiter, H., Su, M., 2006. The characterization of 2n-periodic binary sequences with fixed 1-error linear complexity. In: Gong, G. Helleseth, Song, T., HY. Yang, K. (Eds.), Sequences and their Applications – SETA 2006. SETA 2006. In: Lecture Notes in Computer Science, vol 4086, Springer, Berlin, Heidelberg, http://dx.doi.org/10.1007/11863854_8.

Gao, F., Zhu, L., Shen, M., Sharif, K., Wan, Z., Ren, K., 2018. A blockchain-based privacy-preserving payment mechanism for vehicle-to-grid networks. IEEE Netw. 32 (6), 184–192. http://dx.doi.org/10.1109/MNET.2018.1700269.

Garewal, K.S., 2020. Merkle trees. In: Practical Blockchains and Cryptocurrencies. Apress, Berkeley, CA, http://dx.doi.org/10.1007/978-1-4842-5893-4_10.

Genç, Y., Afacan, E., 2021. Design and implementation of an efficient elliptic curve digital signature algorithm (ECDSA). In: 2021 IEEE International IOT, Electronics and Mechatronics Conference. IEMTRONICS, Toronto, on, Canada, pp. 1–6. http://dx.doi.org/10.1109/IEMTRONICS52119.2021.9422589.

Ghasempour, A., 2019. Internet of Things in smart grid: Architecture, applications, services, key technologies, and challenges. Inventions 4 (1), 22. http://dx.doi.org/10.3390/inventions4010022.

Hong, H., Hu, B., Sun, Z., 2019. Toward secure and accountable data transmission in narrow band Internet of Things based on blockchain. Int. J. Distrib. Sensor Netw. http://dx.doi.org/10.1177/1550147719842725.

Ile, C., Rafati, S., Schiller, E., 2019. Performance evaluation of LTE-M NBIoT in IoT to blockchain transmissions. https://files.ifi.uzh.ch/CSG/staff/Rafati/Ile-Cepilov-IS.pdf.

Innocent, A.A.T., Prakash, G., 2019. Blockchain applications with privacy using efficient multiparty computation protocols. In: 2019 PhD Colloquium on Ethically Driven Innovation and Technology for Society. PhD EDITS, pp. 1–3. http://dx.doi.org/10.1109/PhDEDITS47523.2019.8986954.

Jabbar, R., Kharbeche, M., Al-Khalifa, K., Krichen, M., Barkaoui, K., 2020. Blockchain for the internet of vehicles: A decentralized IoT solution for vehicles communication using ethereum. Sensors 20 (14), 3928. http://dx.doi.org/10.3390/s2014392.

Javaid, U., Aman, M.N., Sikdar, B., 2020. A scalable protocol for driving trust management in internet of vehicles with blockchain. IEEE Internet Things J. 7 (12), 11815–11829. http://dx.doi.org/10.1109/JIOT.2020.3002711.

Jiang, T., Fang, H., Wang, H., 2019. Blockchain-based internet of vehicles: Distributed network architecture and performance analysis. IEEE Internet Things J. 6 (3), 4640–4649. http://dx.doi.org/10.1109/JIOT.2018.2874398.

Kumar, Priyanka, Shah, Maharshi, 2020. To build scalable and portable blockchain application using docker. http://dx.doi.org/10.1007/978-981-15-4032-5_56.

Li, W., Guo, H., Nejad, M., Shen, C.-C., 2020. Privacy-preserving traffic management: A blockchain and zero-knowledge proof inspired approach. IEEE Access 8, 181733–181743. http://dx.doi.org/10.1109/ACCESS.2020.3028189.

Martín, A.G., Leal, R.P., Armada, A.G., Durán, A.F., 2018. NBIoT random access procedure: System simulation and performance. In: 2018 Global Information Infrastructure and Networking Symposium. GIIS, pp. 1–5. http://dx.doi.org/10.1109/GIIS.2018.8635738.

Mohammed, C.P., Chopra, S.R., 2023. Blockchain security implementation using python with NB-IoT deployment in food supply chain. In: 2023 International Conference on Emerging Smart Computing and Informatics. ESCI, Pune, India, pp. 1–5. http://dx.doi.org/10.1109/ESCI56872.2023.10100139.

Na, D., Park, S., 2022. IoT-chain and monitoring-chain using multilevel blockchain for IoT security. Sensors 22 (21), 8271. http://dx.doi.org/10.3390/s22218271.

Nguyen, L.D., Kalor, A.E., Leyva-Mayorga, I., Popovski, P., 2020. Trusted wireless monitoring based on distributed ledgers over NB-IoT connectivity. IEEE Commun. Mag. 58 (6), 77–83. http://dx.doi.org/10.1109/MCOM.001.2000116.

Nguyen, Lam, Leyva-Mayorga, Israel, Lewis, Amari, Popovski, Petar, 2021. Modeling and analysis of data trading on blockchain-based market in IoT networks. IEEE Internet Things J. PP, 1. http://dx.doi.org/10.1109/JIOT.2021.3051923.

Popli, S., Jha, R.K., Jain, S., 2019. A survey on energy efficient narrowband Internet of Things (NBIoT): Architecture, application and challenges. IEEE Access 7, 16739–16776. http://dx.doi.org/10.1109/ACCESS.2018.2881533.

Praptodiyono, Supriyanto, et al., 2020. Mobile IPv6 vertical handover specifications, threats, and mitigation methods: A survey. Secur. Commun. Netw. 2020, 5429630:1–5429630:18.

Puaut, I., 2002. Real-time performance of dynamic memory allocation algorithms. In: Proceedings 14th Euromicro Conference on Real-Time Systems. Euromicro RTS 2002, Vienna, Austria, pp. 41–49. http://dx.doi.org/10.1109/EMRTS.2002.1019184.

Qi, X., Zhang, Z., Jin, C., Zhou, A., 2021. A reliable storage partition for permissioned blockchain. IEEE Trans. Knowl. Data Eng. 33 (1), 14–27. http://dx.doi.org/10.1109/TKDE.2020.3012668.

Routray, S.K., Gopal, D., Javali, A., Sahoo, A., 2021a. Narrowband IoT (NBIoT) assisted smart grids. In: 2021 International Conference on Artificial Intelligence and Smart Systems. ICAIS, pp. 1454–1458. http://dx.doi.org/10.1109/ICAIS50930.2021.9395891.

Routray, S.K., Gopal, D., Pallekonda, A., Javali, A., Kokkirigadda, S., 2021b. Measurement, control and monitoring in smart grids using NBIoT. In: 2021 6th International Conference on Inventive Computation Technologies. ICICT, pp. 229–234. http://dx.doi.org/10.1109/ICICT50816.2021.9358604.

Routray, S.K., Sharmila, K.P., Akanskha, E., Ghosh, A.D., Sharma, L., Pappa, M., 2021c. Narrowband IoT (NBIoT) for smart cities. In: 2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks. ICICV, pp. 393–398. http://dx.doi.org/10.1109/ICICV50876.2021.9388513.

Samara, G., Aljaidi, M., Daoud, E.A., Al-Safarini, M.Y., Alsayyed, G.M., Almatarneh, S., 2022. Message broadcasting algorithm implementation using mobile long term evolution and narrow band Internet of Things over intelligent transportation system (ITS). In: 2022 International Arab Conference on Information Technology. ACIT, Abu Dhabi, United Arab Emirates, pp. 1–7. http://dx.doi.org/10.1109/ACIT57182.2022.9994179.

Sankaran, S., Sanju, S., Achuthan, K., 2018. Towards realistic energy profiling of blockchains for securing Internet of Things. In: 2018 IEEE 38th International Conference on Distributed Computing Systems. (ICDCS), pp. 1454–1459. http://dx.doi.org/10.1109/ICDCS.2018.00148.

Shahjalal, M., Islam, M.M., Alam, M.M., Jang, Y.M., 2022. Implementation of a secure LoRaWAN system for industrial Internet of Things integrated with IPFS and blockchain. IEEE Syst. J. 16 (4), 5455–5464. http://dx.doi.org/10.1109/JSYST.2022.3174157.

Sultania, A.K., Blondia, C., Famaey, J., 2021. Optimizing the energy-latency tradeoff in NB-IoT with PSM and eDRX. IEEE Internet Things J. 8 (15), 12436–12454. http://dx.doi.org/10.1109/JIOT.2021.3063435.

Wang, T., Zhu, W., Ma, Q., Shen, Z., Shao, Z., 2020. ABACUS: Address-partitioned bloom filter on address checking for uniqueness in IoT blockchain. In: 2020 IEEE/ACM International Conference on Computer Aided Design. ICCAD, pp. 1–7.

Zhang, H., Zhao, Y., 2022. Vehicle load monitoring method based on NBIOT. In: 2022 5th International Symposium on Autonomous Systems. ISAS, pp. 1–5. http://dx.doi.org/10.1109/ISAS55863.2022.9757347.

Vamshi Sunku Mohan received the Bachelor of Engineering (B.E.) degree in electrical and electronics engineering from the Bangalore Institute of Technology, Bangalore, and the Master of Science (M.S.) degree in electrical and computer engineering from the University of Massachusetts Lowell. He is currently a full-time Ph.D. Scholar at the Center for Cyber Security Systems and Networks, Amrita Vishwa Vidyapeetham. His research interests include machine learning, blockchain, mobile communication protocols, IoT security and cryptography.

Dr. Sriram Sankaran received the B.Tech(Hons.) from the Malaviya National Institute of Technology Jaipur, Jaipur, formerly known as the Regional Engineering College, and the M.S. and Ph.D. from the University at Buffalo, The State University of New York. He is an Associate Professor with the Center for Cybersecurity Systems and Networks, Amrita Vishwa Vidyapeetham, Amritapuri Campus, where he directs the Sustainable Computing Laboratory. He has published more than 50 papers all in the areas of mobile embedded computing systems and cybersecurity. His research interests include mobile embedded computing systems, cybersecurity, and the Internet of Things, with a particular focus on energy efficient computing, modelling, and simulation. He served as the Program Chair for the International Symposium on Embedded Computing and System Design (ISED) 2019 and also bagged the Best Paper Award at the IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS) 2020.

Dr. Priyadarsi Nanda received Ph.D. in Computing Science from the University of Technology Sydney (UTS), Australia. He is an academic at the University of Technology Sydney (UTS) and has more than 32 years of experience in Teaching and Research. He is a strong researcher in Cybersecurity, IoT security, Internet Traffic Engineering, wireless sensor network security and many more related areas. His most significant works have been in the area of Intrusion detection and prevention systems (IDS/IPS) using image processing techniques, Mitigation of cyber-attack using machine learning techniques in IOT based applications, Intelligent

firewall design, security in Intelligent Transportation System (ITS) etc. He has published over 120 high quality refereed research papers including Transactions in Computers, Transactions in Parallel Processing and Distributed Systems (TPDS), Future Generations of Computer Systems (FGCS) as well as many ERA Tier A/A* conference articles. In 2017, his work in cyber security research has earned him and his team the prestigious Oman research council's national award for best research. Dr. Nanda has successfully supervised 17 HDR at UTS (14 Ph.D. + 3 Masters), and currently, supervising 8 Ph.D. students.

Dr. Krishnashree Achuthan received the Ph.D. degree from Clarkson University, NY, USA. She currently heads the Center for Cybersecurity Systems and Networks and the Amrita Technology Business Incubator (Amrita TBI), Amrita Vishwa Vidyapeetham. She is also the Dean of P.G. Programs at the Amrita School of Engineering, Coimbatore. She is an Ardent Researcher with multi-disciplinary interests. She holds 29 U.S. patents and has published widely in highly acclaimed international journals.