International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(22s) 2026 pp. 1853-1882
https://doi.org/10.70917/ijcisim-2026-5795
Article

Secure Implementation and Evaluation of IoT Networks using PKC and Hash-Based Blockchain Framework

Sahar Ali Laskar[1], Md Afsarul Hoque[2] , J K M Sadique Uz Zaman*[3]

1,3 Department of Computer Science & Technology, University of North Bengal, Siliguri, West Bengal, India
2 North Bengal St. Xavior's College, University of North Bengal, Jalpaiguri, West Bengal, India
1 saharlaskar1991@gmail.com, 2 ma.hoque.pqc@gmail.com, 3 jkmsadique@gmail.com

Abstract: Internet of Things (IoT) is growing in its scope of sensors, industrial automation, transport, healthcare, agriculture and smart-city services, with edge devices having limited memory, computation, energy and communications. While blockchain can offer an audit trail and the source of device transactions and data provenance, traditional designs often have communication, storage, cryptographic and consensus costs that are not suitable for the constrained nodes. This article develps a simulation-based implementation methodology of a light weight public-key-cryptography and SHA-256 hash based blockchain system that is suitable for Class 1 and Class 2 IoT environment. The architecture proposed delegates sensing and transaction signing tasks to thin clients, places ledger maintenance and PBFT consensus at the gateway or validator nodes and separates large sensor payloads from the ledger, only recording their SHA-256 commitment in the ledger. The implemented architecture is evaluated with three well recognized public key cryptography scheme Ed25519, ECDSA P-256 and RSA-2048 . The implementation methodology is a mix of ns-3 for packet level network simulation and Contiki-NG / Cooja for the constrained operating-system behavior, while the independent cross-check is specified as OMNeT++ / INET for selected scenarios. Each scenario projection for latency, energy, throughput, storage and IDS behavior is created using a separate parameterized virtual-simulation model. The values are intended to be model generated and not real measurements using CC2538, ESP32 or Raspberry Pi hardware. The resulting framework offers a repeatable route from conceptual architecture to simulator implementation and identifies the measurements that need to be taken on the upcoming hardware validation.
Keywords: Blockchain, Internet of Things, public-key cryptography, Ed25519, ECDSA, RSA, SHA-256, PBFT

1. Introduction

Private and Consortium Blockchains provide the required security for edge nodes with few resources, while Lightweight Aggregate Signatures and Threshold-based Consensus reduce the transaction latency from seconds to the 80–150 ms range, as demonstrated in recent advances of the hybrid blockchain systems for cooperative IoT networks [1]. More importantly, comprehensive lightweight blockchain architecture as COSIER demonstrates the significant improvement in latency, blockchain build time and security against attacks in IoT applications by optimizing the blockchain architecture, authentication, consensus and storage together. Optimised implementations for ARM Cortex-M4 processors have demonstrated that high-speed, side-channel-resistant signatures are feasible on the very microprocessors which are found on the IoT devices of Class 1 and Class 2 platforms, requiring only few hundred thousand cycles for key generation, signing and verification [2], [3]. In 2023, a taxonomy of lightweight blockchain solutions was published in order to organize the five main dimensions of design architecture, authentication, cryptography, consensus and storage to help define the design space that any new constrained-IoT ledger will traverse [4]. Consensus mechanisms such as machine learning and feather-light blockchain infrastructures for resource-constrained edge devices in 2022 already demonstrated that removing the classical proof-of-work paradigm and

replacing it with a hierarchical or DAG-based one can make blockchain within the energy and latency budget of sensor networks [5], [6].

The Internet of Things has come a long way from the initial concepts of connected sensors to a new form of computing that involves complex and diverse objects constantly producing, sharing and reacting to digital data. The industry is projecting that there will be over 29 billion connected “Internet of Things” devices by 2030. Many of the devices that are expected will be low-cost, low-power devices and the computing and storage resources are intentionally cribbed. This is why endpoints are so desirable – they are simple and efficient to deploy, making them hard to secure with desktop, server and cloud security measures.

Security is then an architectural issue in the process of obtaining an encryption problem. When devices are communicating via a network that can be lossy or intermittently connected, IoT deployments need to establish device identity, authenticate messages, ensure data integrity, prevent replay attacks, limit access and maintain trusted records. Sicari et al. [7] pointed out that the requirements of confidentiality, authentication, access control, privacy and trust are interrelated in IoT systems. Similarly, Granjal et al. demonstrated that the overhead and complexity of the conventional Internet security protocols are becoming non-trivial impediments in many constrained settings [8]. Frustaci et al. later pointed out that the variety and constraints of IoT platforms make it possible to exploit a wide array of devices and lessen the effectiveness of applying security measures to various types of devices [9].

When the Internet Engineering Task Force (IETF) constrained-node terminology is included, the resource problem is even more evident. RFC 7228 defines the following two classes of devices: Class 1, about 10 KiB of RAM and around 100 KiB of code space; Class 2, about 50 KiB of RAM and around 250 KiB of code space [10]. The numbers are not necessarily hardware parameters but are offered as an engineering language. Doing this because the cryptographic primitive used is modern is not a good reason to consider it to be lightweight because it relies on a general purpose operating system, a large certificate chain, a persistent ledger replication, or a repeated and expensive consensus operation. The complete path of the protocol is important.

This is a situation where blockchain is appealing, as it shifts the trust paradigm. A distributed ledger can replace the need to have a single server to keep a single authoritative history of device events: the entries can be cryptographically hashed and the consensus protocol can determine who accepts them. The approach can be used for data provenance, auditability, distributed authorization and accountability. It does not in itself ensure that the sensor reading is accurate. A ledger may be used to record evidence of the report of a device but not be able to determine if the physical phenomenon was measured accurately. This is a key part of a defendable design of IoT and blockchain.

There are already a number of architectural solutions to the incompleteness of conventional blockchains for limited devices in the literature. Dorri et al. proposed an optimized blockchain architecture for IoT, which alleviates the burden caused by traditional public blockchains [11]. Novo has created a distributed architecture based on a blockchain system that would solve the problem of scalability of the centralized authorization system in the IoT environment [12]. Fernández-Caramés and Fraga-Lamas examined the potential applications of blockchain in IoT and listed the issues in the field regarding resources, scalability, privacy, interoperability and consensus [13]. These studies prove that it's not necessary or desired to directly transplant a cryptocurrency network. The more useful design question is what to do about distributing responsibility between the different types of tiers.

The present research is a continuation of a line of research related to the secure implementation of networks in the Internet of Things (IoT) field, based on the application of public-key cryptography and hash-based blockchain technology. In this paper we have two key goals: First, to leverage public-key cryptography for authenticity and security and second, to embrace hash-based blockchain technology for data integrity in small IoT-based devices. Thought we study and find out what the cryptographic primitive is, investigate blockchain operations and its limitations, look at the statistical properties of potential hash functions and finally create an implementation approach for constrained IoT [14]**.** In 2024, a paper on the working principles and applications of blockchain explored the structure of blocks, consensus mechanisms, hashing, digital signatures, smart contracts, applications and implementation issues [14]. A follow-on paper extended the application and security exploration, extending to the

additional domains and readdressing implementation issues [15]. The overall take-home message is simple: the studies had made the conceptual argument for blockchain in a limited setting, but not yet provided a sound blueprint for doing so.

A hash-function study was conducted with the Strict Avalanche Criterion (SAC) and Bit Independence Criterion (BIC) against messages of sizes ranging from 2 bytes to 1024 bytes. It consumed up to 8000 samples in small number of samples and decreased the sample number in large number of samples [16]. It was observed that the reported standard deviations have converged to the theoretical standard deviations (SAC) of 5.657 for MD5 and 8.000 for SHA-256 [17] as the message increased in length and the largest positive and negative correlations approached 10-3 level. The same work rightly cautions against the use of good SAC/BIC behavior to save MD5 from collision attacks [16]. It's a helpful methodological observation for this paper: statistical diffusion isn't necessarily a security verdict and get SHA-256 is secure.

In this paper we present a virtual implementation methodology. The basic concept is not to substitute some sort of guesswork in the form of numbers for experimentation. Rather, the framework classifies three types of evidence. The first is that the protocol and resource constraints are defined by standards and published literature. Secondly, there are already empirical cryptographic observations that exist in a previous SAC/BIC study. Third, the new system-level values are produced as parameterized simulation projections which explicitly state their assumptions, distributions and limitations. Here the following questions may be arised.

- How a blockchain framework combines PKC and SHA-256 in a virtual constrained-IoT environment
- How latency, throughput, energy and storage vary with network scale and load?
- How RSA, ECDSA and Ed25519 differ under resource-constrained computational models?

This paper has four main contributions. Firstly, it provides a complete three-tier implementation architecture where constrained devices are thin clients and gateway nodes manage the ledger and consensus burden. Second, it offers an implementation plan based on a simulator approach with cross-validation between ns-3, Contiki-NG/Cooja and optional OMNeT++/INET. Thirdly, it supports interchangeable cryptographic modules, such as RSA, ECDSA and Ed25519 and uses SHA-256 as the ledger commitment function. Fourth, it provides a clear virtual testing procedure and model predictions that are clearly separated from what a physical device would measure.

It is not only a numerical contribution, but also a methodological one. The paper does not state that a simulation can simulate all the properties of an actual IoT network. It proposes that by carefully parameterize a virtual testbed it can minimize the implementation gap, reveal protocol bottlenecks, enable repetitive sensitivity analysis and, finally, determine what should be measured when testing with hardware.

The rest of the article is organized as follows. A related work provided in section 2, Section 3 describes the proposed three tier architecture, virtual simulation methodology is given in Section 4, Section 5 provide brief Results and discussion, research implicationand limitation in Section 6. Future scope in Section 7, and the conclusion in Section 8.

2. Related Works

IoT security begins with an uncomfortable engineering fact: the endpoints that need protection are often the least capable components in the system. Sensor nodes can be low memory, low clock, low power radio and duty cycle for preserving the battery. The security protocol must coexist with sensing, routing, retransmission, application logic and operating system state. Thus, the asymptotic complexity of an algorithm is not a good indicator of security cost. The size of the code, the depth of the stack, the expansion of the packets, the time spent with the radio and the number of cryptographic operations per transaction all come into play.

Sicari et al. [7] presented a perspective on IoT security based on the following principles: confidentiality, authentication, access control, privacy, and trust. Granjal et al. [8] studied protocols like TLS, DTLS, and other

protocols in constrained environments, and studied the importance of considering characteristics of the devices and networks involved for protocol selection. RFC 7228 offers a useful classification of constrained devices, and explicitly acknowledges that Class 1 nodes may not have the ability to easily implement the Internet protocol suites that are normally utilized by conventional Internet hosts [10]. The take-home lesson for blockchain is clear: you don't expect a device to store a complete copy of the blockchain just because it can create a digital signature.

Frustaci et al. [9] also associate the shortcomings of IoT with the heterogeneity of the devices and limited implementation conditions. In reality, the attack surface encompasses weak credentials, vulnerable update processes, unprotected network services, implementation flaws, and authentication/authorization shortcomings. While Blockchain can help ensure auditability and integrity, vulnerabilities in the device's operating system or radio interface remain. Therefore, a simulation methodology should not assume that the blockchain is the universal security mechanism but that it is just one layer of the security of a larger system.

2.1 Blockchain Architectures for IoT

The first generation of blockchain systems was conceived based on assumptions that are significantly different from those in IoT. In Bitcoin's architecture [18], a replicated ledger is used, with participants communicating via a peer-to-peer network, and proof-of-work is applied. The design is intentionally permissionless, and resistant to some types of adversarial behavior, but it is not a natural fit with the computational and communication model of low power sensor nodes.

This mismatch was identified by Dorri et al. [11] who proposed an optimised architecture of blockchain for IoT. They split up resource-heavy tasks from limited devices, and they employ a “local blockchain structure” to alleviate a ton of overhead. Instead, Novo [12] considered access management to demonstrate how blockchain can be used to achieve distributed authorization without relying on a central authority. Fernández-Caramés and Fraga-Lamas [13] later surveyed the field and concluded that the typical problems were resource consumption, scalability, privacy, interoperability, and consensus.

This is a development that has been continued in more recent literature. In Erukala et al. [1] the authors show that a hybrid blockchain of fog and consortium cloud ledgers can provide end-to-end authentication and secure storage, and consequently utilize aggregate signatures to handle constrained nodes in 2025. COSIER [2] has combined four light-weight features (architecture, authentication, consensus and storage) into a hierarchical design, which is demonstrated to have lower latency and network overhead. In 2023, Mershad and Cheikhrouhou [4] divided the design space into five orthogonal dimensions, and showed that there are not many solutions that optimize all of them at once. Previously, there were some machine learning consensus and feather-light infrastructures [5], [6] that demonstrated the possibility of non-PoW consensus and hierarchical ledgers on edge hardware.

A design pattern that arises from these studies is the use of heterogeneous resources in an asymmetric fashion. A sensor may generate an authenticated event, a gateway may sign and participate in the consensus, and a storage service may store big payloads. This type of split is not a sign of reduced decentralization of the blockchain, however, because the decentralization of trust does not have to mean equality of computation at each node.

The following framework is based on this edge-assisted principle, with a special cryptographic comparison and an evaluation plan for simulators. A compact metadata and hash is stored on the ledger and full sensor payloads are off ledger. The architecture also makes consensus a responsibility of the validators. This decreases the state and computation necessary from Class 1 and Class 2 clients and makes the resource model virtual IoT-operating system compatible.

2.2 Cryptography on Constrained Devices

Public-key cryptography is central to the proposed design because blockchain transactions need attributable authorship, not merely confidentiality. RSA provides a useful historical baseline, but its large modulus and computationally expensive private-key operations can be burdensome on constrained processors. Gura et al. [19]

demonstrated the practical advantages of elliptic-curve cryptography over RSA on 8-bit CPUs, establishing an important design principle for embedded security: equivalent security levels need not have equivalent implementation costs.

Hutter and Schwabe have created an elliptic-curve primitive for 8-bit AVR microcontrollers that shows that modern elliptic-curve primitives can be realized on very constrained processors [20]. This trend has continued in more recent Curve25519 and Ed25519 implementations for ARM Cortex-M processors [21], [22] that include techniques to optimize arithmetic and make implementations constant-time. The side-channel resistant implementation of Ed25519 by Owens et al. [3] further shows that the Keygen, Sign and Verify routines can be efficiently implemented on a Cortex-M4, making the use of the Ed25519 suitable for the present Class 1/Class 2 profiles.

The above architecture makes Ed25519 appealing for two reasons. First, its signature format and key size are small compared to that of RSA. Secondly, to prevent some failures related to substandard random-number generation in ECDSA that are present in the original ECDSA algorithm, EdDSA as defined in RFC 8032 [23] employs deterministic nonce derivation in the signature process. ECDSA is still relevant in that it is a widely implemented and useful baseline, and RSA also is useful for comparison with the previous thesis work.

The framework thus enables the cryptographic algorithm to be used as a pluggable module. The simulator does not make any assumptions about the universal superiority of Ed25519 in any implementation. Rather, the evaluation is carried out against a number of different parameters like the time taken to operate, the energy required to operate, the size of the signature, and the cost of verification. Measured parameters can subsequently be used in place of those parameters in a later hardware study without altering the model of the network.

The foundation for the public-key aspects of the RSA study is given in [24]. The methodological importance of it to the present paper is the more important one: Cryptographic primitives should be evaluated not only on the basis of labels, but also on the basis of their real transformation and implementation behavior. The system-level background is provided by the blockchain papers [14] and [15]. The statistical basis is given by the MD5/SHA-256 study [16]. It is used to report the SAC and BIC for 10 different message lengths.SAC and BIC can set up useful statistical properties, but they are not able to overcome known cryptanalytic weaknesses like MD5 collisions [25], [26].

RSA is now a comparison baseline, ECDSA is a popular alternative to RSA (elliptic curve), Ed25519 is the leading option, and the blockchain architecture is the system where the cost of the cryptographic operation is measured. The novelty is that each component is new in itself. The key contribution is providing details on how the various components would be implemented and assessed as an entire virtual IoT system, while assuming that some measurements of the hardware would not exist. A network simulator is used to simulate a network. A virtual testbed is a software that simulates a network.

In networking, simulation is popular because researchers can manipulate the network topology, traffic, radio conditions, number of nodes, and protocol parameters in a controlled manner and systematically add custom application and protocol components to the simulator via its modular architecture, as is done in the discrete-event network simulator ns-3 [27]. IEEE 802.15.4 scenarios can be naturally based on the lr-wpan model, and energy models can represent the effects from batteries.

Cooja and Contiki offer an alternative view. The Contiki operating system is designed as a small operating system for the very small sensors in the networks [28] and Contiki-NG provides support for IPv6, RPL, CoAP, energy monitoring, and simulation workflows using Contiki's Cooja. The present Contiki-NG documentation allows for simulation with Cooja, and features tools for energy estimation like Energest [29]. This is especially helpful when the research question is not just about moving packets, but also the behaviour of an IoT Operating System.

Another discrete-event simulation environment is OMNeT++ [30], which is also based on components. Selected IP and transport scenarios can be cross checked with INET. There is no intention to perform all the tests in three simulators. This would lead to an increase in effort, but not necessarily be an increase in validity. A better

approach is to simulate the constrained node behavior with ns-3, the network behavior with Cooja/Contiki-NG, and run a few independent sensitivity tests with OMNeT++/INET.

Ledger-level studies can be performed using blockchain simulators, like SimBlock or BlockSim, but these simulators alone cannot emulate constrained radio behavior. The suggested methodology is thus to directly incorporate the application of blockchain into the network simulation. This will ensure that the causal sequence in which the transactions are created, sent as packets, confirmed and settled is preserved.

2.3 Research Gap

There are plenty of IoT/blockchain architectures in the literature, cryptographic benchmarks and plenty of simulation studies of wireless sensor networks. The missing combination is thinner. The current research aims to find a single experimental model in which the behavior of constrained nodes, the public-key signing, the SHA-256 commitments, the transport of the transactions, the PBFT consensus, the off-chain storage and the monitoring of the security are represented together. The gap is not the absence of all individual components; it is the absence of an integrated and transparent simulation methodology which links the components.

The second gap is related to evidence discipline. The number produced by a simulation does not necessarily reflect a real device's measurement. This distinction is highlighted in the present paper. Literature-calibrated cryptographic parameters are not physical benchmarks. Latency and energy values are not obtained on CC2538 or ESP32 boards, but rather model-generated. This distinction is used to enhance the reproducibility. A researcher who has access to ns-3 can implement the algorithms and execute the scenarios and then obtain trace-derived statistics in place of each one of the tables. Then, an experimenter can recreate the same scenarios with real equipment. The manuscript is thus not an unverified simulation but rather a stepping stone from conceptual design to a testable implementation.

3. Proposed System Architecture

The proposed system architecture is discussed in following subsection.

3.1 Overview and Design Rationale

The proposed system consists of three-tiered IoT architecture where computationally heavy security, validation, intrusion detection and blockchain operations are offloaded from IoT constrained devices to more capable gateway and validator nodes. The architecture is also designed to be responsible for the distribution of responsibilities across the 3 layers in an asymmetric manner.

Tier 1 uses resource constrained devices, which are mainly tasked with least resource-intensive sensor data acquisition, light processing, transaction construction, cryptographic signing, counter of transactions, and checking the proof of blockchain inclusion. The Tier 2 consists of gateway and validator nodes, which are responsible for a transaction validation, cryptographic signature verification, intrusion detection, transaction aggregation, PBFT consensus, blockchain maintenance, and state management. Tier 3 stores off-chain relatively large sensor payloads that do not necessarily need to be stored in the blockchain.

Separation of responsibilities aims to maintain distributed trust without holding the entire blockchain or running costly consensus operations. As such, a constrained IoT device does not have to carry out all the same operations as a gateway or validator. The device instead generates a reliable transaction that is signed, and the gateway and validator layer does the computationally heavy work of verifying and agreeing to the transaction. The sensor data are acquired and preprocessed as shown at Tier 1 in figure 1, and the proposed transaction flow starts there. The raw payload is linked to a device identifier, timestamp, payload hash, an increasing transaction number (monotonic) and a payload reference. If off-chain storage is activated, the payload is stored on the Tier 3 and is referenced in the transaction with a Content Identifier (CID) and a cryptographic commitment. The transaction is then hashed and digitally signed with the private key of the device, and sent to Tier 2.

In Tier 2, the transaction packet received is structurally verified and validated by checking the digital signature. If the transaction information is valid, it is taken to payload retrieval and integrity verification. Tier 2 calls Tier 3 for the payload, re-computes a new digest of the payload using SHA-256, then compares the new digest with the integrity value for the transaction. If the transactions pass these checks successfully, they are added to the mempool for processing by PBFT. The validator layer subsequently validates the batches of transactions using the PBFT consensus. When the validators reach consensus to add the block and the block has achieved a quorum, it is saved and the state of the blockchain is changed. A block confirmation with the information of the block and Merkle proof is sent back to Tier 1. It's possible for the constrained device to then confirm that the transaction is included inside the committed block, though not retaining a full blockchain copy.

Also, the separation of on-chain transaction data and the off-chain sensor payloads helps to limit the growth of the blockchain. The transaction can only store the cryptographic commitment and the reference used to certify the integrity of the sensor payload, which can have hundreds of bytes. The blockchain, as a result, is mostly a source of integrity and trust, not a repository for many volumes of raw sensor data.

3.2 Tier 1 – IoT Device Thin Client

Tier 1 is for the edge devices that have limited resources used for sensing and transaction generation (IoT devices). The design reduces the amount of computation, memory, storage, and communication needed at the endpoint, yet retains enough cryptographic information to make transaction authenticity and later verify that the transaction is part of a blockchain.

The process of the system may be started with sensor data acquisition as shown in Figure 1. The device links to the newly gained data its own ID and preprocesses the data. The raw data created is the transaction payload.

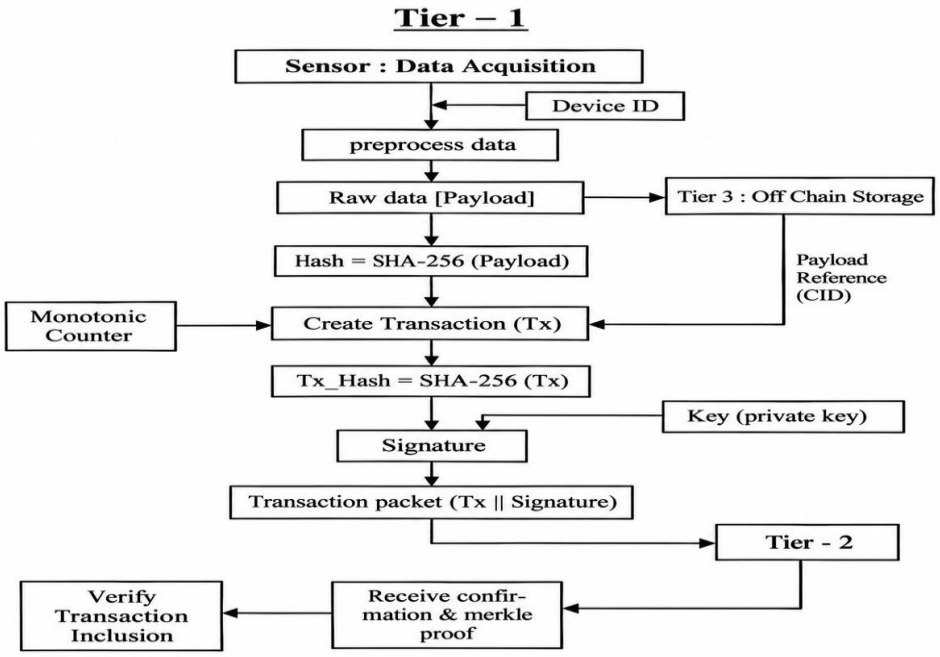

Figure 1: Tier-1 IoT device thin-client

A content digest is first computed using the SHA-256 algorithm. Each device maintains a monotonic transaction counter to ensure the transactions can be ordered and replayed transactions would not be accepted. The device then creates a canonical transaction that includes the metadata needed: the device identifier, counter, payload hash, payload reference (if available), and some selected contextual information, as well as a timestamp. If off-chain storage is used, the payload is stored off-chain. Tier 3 is required to generate a Content Identifier (CID) for the payload stored and

store it with that identifier. The identifier (CID) (payload reference) is included in the transaction. The transaction will then be serially and hashed using SHA-256. The digest of the transaction is then signed with the devices private key.

The proposed system is designed to be extensible to support an array of public-key algorithms for comparison of cryptographic overhead. Ed25519 is chosen as the default lightweight signature algorithm for its small sized signature generation and efficient public-key operations. The simulator also includes the ECDSA P-256 and the RSA-2048 algorithm(s) with support via an easy-to-use common cryptographic interface. This interface enables the system to evaluate and compare various metrics: key generation, signing, verification and computational cost, energy consumption and etc. The resulting object is the transaction packet, which is comprised of the transaction and a digital signature:

Transaction Packet= Tx || Signature

The packet is transmitted from Tier 1 to the Tier-2 gateway for validation. The IoT device does not carry a full chain with it. On the contrary, it stores only some recent block headers or other form of cryptographic commitments. Tier 2 responds with a confirmation message that contains the block information, Merkle proof, and after a transaction has been processed and included in a committed block. Using this information, the IoT device can check whether the transaction has been included in the blockchain without downloading or saving the entire blockchain. The Tier-1 functionality can therefore be summarized as:

- Sensor data acquisition
- Data preprocessing
- Device identification
- Payload hashing
- Monotonic transaction counter
- Transaction construction
- Digital signature generation
- Transmission of the signed transaction
- Storage of a limited rolling blockchain-header window
- Merkle-proof-based transaction-inclusion verification

3.3 Tier 2 – Gateway and Validator

The tier-two layer proposes to be the gateway, edge-processing, validator, intrusion-detection, and blockchain-consensus layer. Tier 2: It's made up of nodes with relatively more resources, which can do more intensive operations like cryptographic and consensus, compared to Tier 1. As illustrated in Figure 2, the transaction packet, created by the Tier-1 device, is received via the gateway interface.

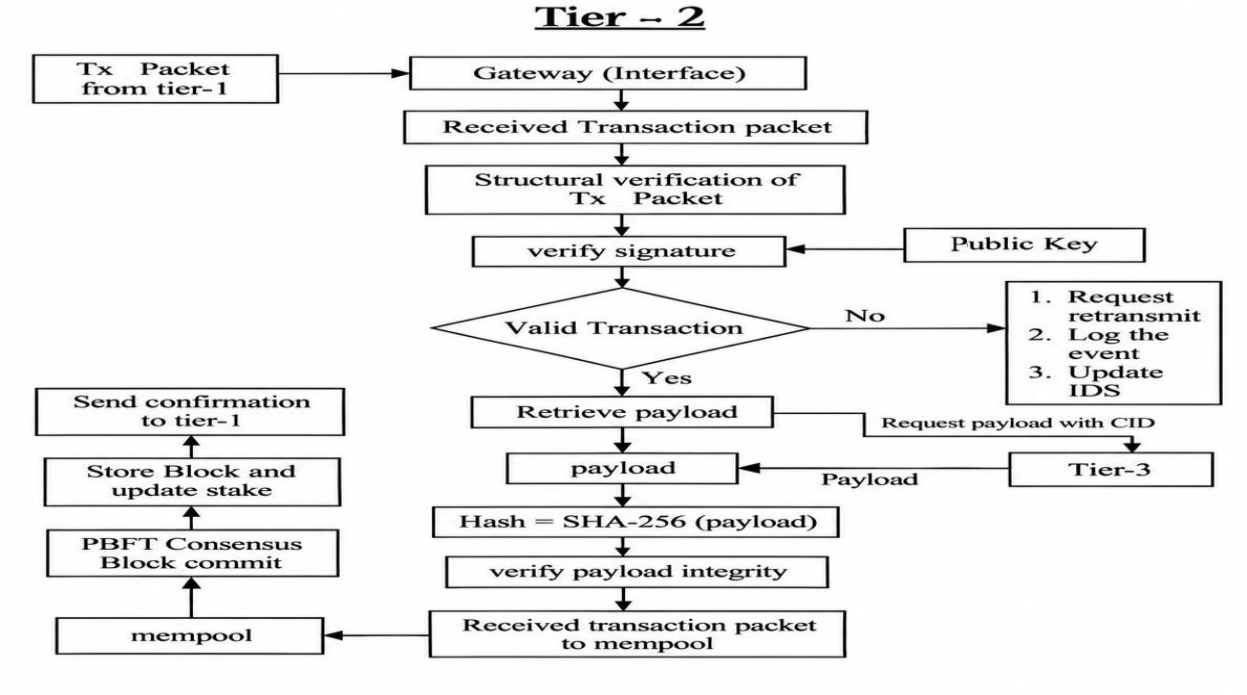

Figure 2: Tier-2 gateway and validator

The first step in receiving the transaction is to check its structure. The gateway verifies the following required transaction fields - transaction format, canonical serialization, device identifier, transaction counter, timestamp or acceptable time window, payload reference, etc. Then it is verified using the respecting public key by Digital-sig. In invalid/duplicate transactions, the transaction is rejected and recorded. If a transaction is invalid, duplicate or fails authentication it is rejected and recorded. Security events are also passed to the inherent intrusion-detection mechanisms with appropriate security events. If the transaction is valid, Tier 2 gets the corresponding sensor payload from Tier 3. The request is done by specifying the CID in the transaction. Tier 2 then uses the same data to calculate:

$H' = SHA\text{-}256(\text{retrieved payload})$

The newly created digest is compared to the payload integrity value in the transaction. If the value is found to be the same, the payload is deemed to be unaltered, and the received transaction packet is put into the mempool. If the verification is not successful, the transaction is denied. It is important to note that this separation is essential as the full sensor payload is not directly stored on the blockchain. Rather, the transaction will contain cryptographic data to help verify that the payload returned from off-chain storage is actually the same payload that was committed by the IoT device.

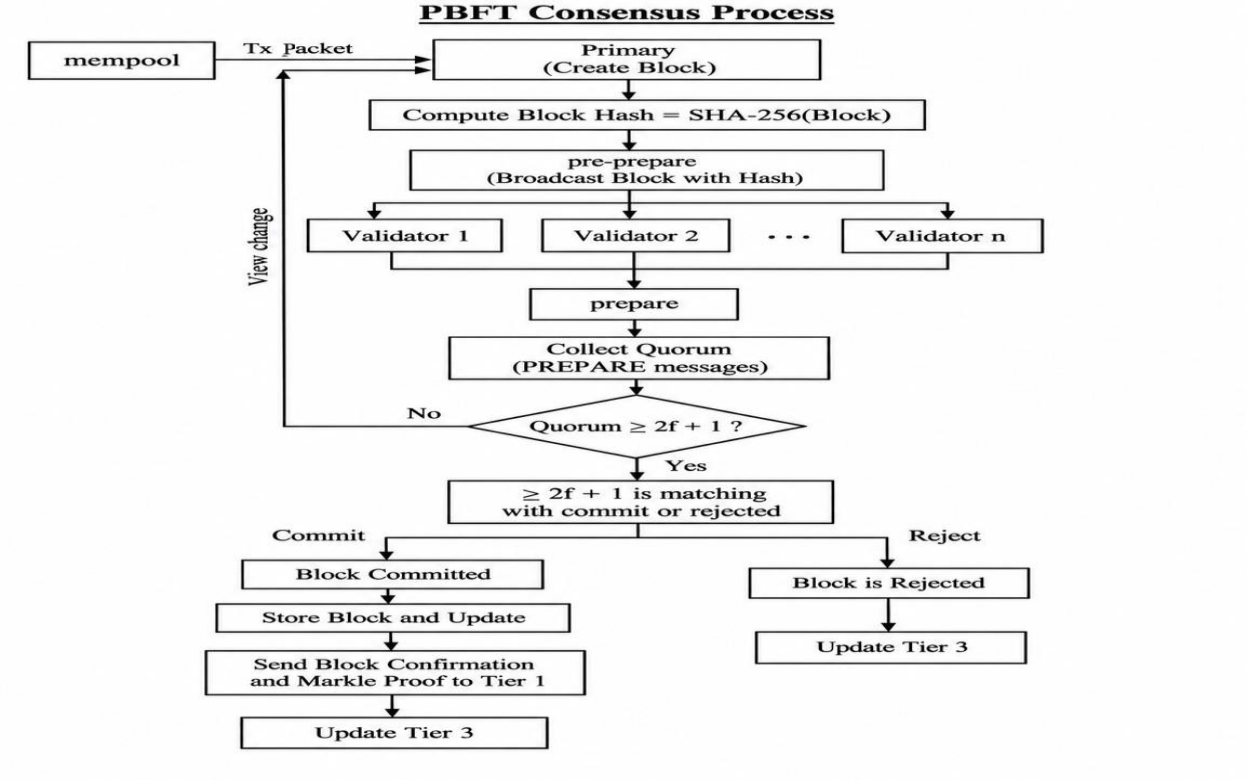

Figure 3: The consensus process of PBFT

The IDS is embedded into Tier 2, as validator/gateway nodes can monitor the transaction and consensus behavior of every device. The proposed IDS contains the following detection mechanisms for: Equivocation, Censorship behavior, Replay attacks and Sybil-related behavior.

Most of the time the censorship detector is able to detect abnormal behaviour from withholding or rejecting transactions from transactions with a configurable threshold. Replay detection is implemented with transaction counters, transaction identifiers and previously seen transactions. The equivocation detection considers conflicting validator behaviour or conflicting information related to the same consensus operation. Auditable security events are created when any type of security event is detected by the IDS. This enables the detection to be added to the blockchain based trust infrastructure, and not just a temporary log at the gateway.

Those transactions able to be accepted by the gateway and payload-integrity check go into the mempool. The validator layer then bundles up the valid transactions into blocks and carries out PBFT consensus. The proposed permissioned validator network is based on PBFT, as there is an assumption that there are a relatively small and known number of validators. The PBFT process is represented independently in Figure 3 and comprises block creation, computation of the block hash, PRE-PREPARE dissemination, PREPARE processing, quorum collection, and commitment.

The primary validator constructs a candidate block that includes the transactions that are in the mempool and calculates the hash of the block using SHA-256. The main broadcasts the PRE-PREPARE message, which includes the proposed block and the proposed block's hash. The validator replicas take part in PREPARE and corroborate the proposal. The PREPARE messages are collected to see if the quorum is achieved. The first quorum condition is: Quorum ≥ 2f+1. When the necessary quorum is not present, the protocol goes through the View-Change protocol.

If sufficient matching validator messages are obtained, the validators determine whether the required (2f+1) agreement supports committing or rejecting the proposed block. When the commit condition is satisfied, the block is committed, stored, and the blockchain state is updated. A block confirmation and Merkle proof are then returned toward Tier 1, while the corresponding Tier-3 state can also be updated. If the validators reach the rejection decision, the block is rejected and the relevant Tier-3 state is updated accordingly.

3.4 Tier 3 – Off-Chain Payload Storage

Tier 3 is the off-chain storage layer where sensor payloads are stored. It is mainly designed to avoid too large sensor data items to consume storage resources on the blockchain. The Tier-3 processing flow is presented in Figure. 4.

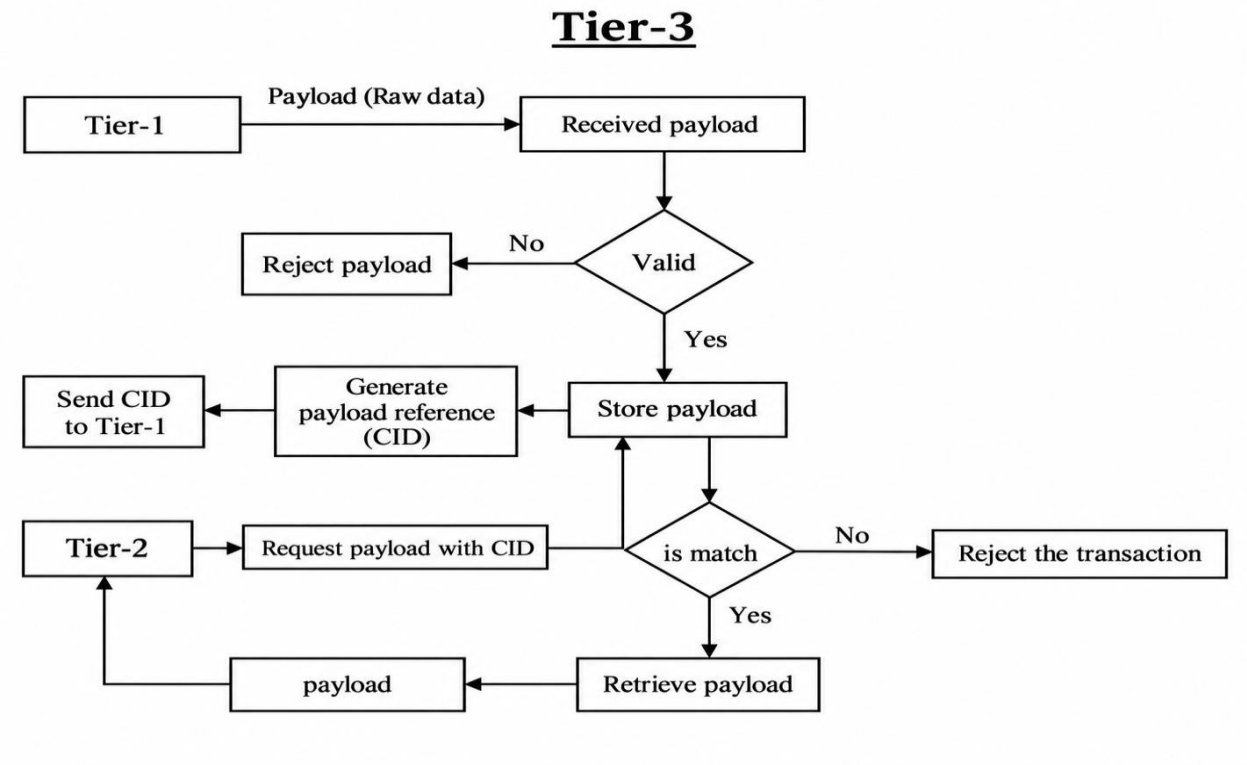

Figure 4: Tier-3 off-chain storage

If the sensor data is created at Tier 1, the original payload is sent to the Tier-3 storage service. The payload received is checked for validity prior to its storage in the addressable repository. Successfully stored Tier 3 produces a Content Identifier (CID) or other content-addressable reference. The CID goes back to tier 1 and is then put into the transaction created by the device. The resulting architecture separates the payload from its blockchain representation: Blockchain transaction → [CID + integrity commitment], where CID→actual sensor payload.

Tier 2 then sends a request for the payload to Tier 3 from the CID upon validation of the transaction. Tier 3 fetches the appropriate payload and passes it back to Tier 2. Tier 2 then works out a new digest of the transaction with SHA-256 and checks it against the integrity commitment for that transaction. When the calculated and the expected values are matching, the payload is assumed to be legitimate and has not been changed. Otherwise, if they do not line up, they will be rejected.

The simulation proposed doesn't need to deploy a complete IPFS system, instead using an IPFS/DHT-like abstraction. The abstraction models include payload retrieval, capacity in storage, retrieval delay and content addressable storage. This allows the validation of the proposed architecture, without increasing the complexity of the deployment of a full distributed storage infrastructure. The abstraction of storage can later be changed to a concrete IPFS node, distributed hash table, or other concrete implementation of content-addressable storage.

Large sensor payloads do not need to be periodically replicated over all the blockchain validators, thus limiting the increase in on-chain storage. Validators rather keep track of the transaction metadata and cryptographic commitments that are required to verify integrity.

Tier 1 acquires sensor data, preprocesses the data, obtains or associates the payload reference, calculates the payload hash, creates the transaction, and digitally signs it. The signed transaction is transmitted to Tier 2, where the gateway performs structural validation and signature verification. For a valid transaction, Tier 2 requests the corresponding payload from Tier 3 using the CID. Tier 3 returns the payload, and Tier 2 verifies its integrity by recalculating the SHA-256 digest. After successful integrity verification, the transaction packet is added to the Tier-2 mempool. The validator network then performs PBFT consensus. A block is committed only when the required agreement is achieved. The committed block is stored and the blockchain state is updated. Finally, a confirmation containing the relevant block information and Merkle proof is sent toward Tier 1, allowing the resource-constrained device to verify that its transaction has been included in the blockchain without maintaining the complete ledger.

3.5 Security mechanisms and threat considerations

It is a collection of complementary mechanisms that are proposed as security architecture, instead of relying on one cryptographic primitive are as follows:

- Digital signatures are used to make a transaction authentic, and connect it to the device it came from. The default signature mechanism is Ed25519, which are followed by ECDSA P-256 and RSA-2048 for comparative testing.
- The cryptographic commitments for sensor payloads, transactions, and blocks are provided by SHA-256. The payload hash is used for Tier 2 to ensure that the data returned from Tier 3 is the same as what is committed by the transaction, in the case of a need.
- The use of monotonic transaction counters mitigates the risk of replay attacks, since the validators are able to identify that a previously submitted/replayed transaction or that a transaction is out of order.
- The offchain storage mechanism for CID is a separation of large sensor payload files and transaction data relative to the blockchain, but also a cryptographically verifiable reference of the payload.
- The built-in IDS system is designed to detect inappropriate behavior in the transaction and validator. The use of transaction histories and validator observations can detect or identify equivocation, censorship, replay, Sybil-related behavior and more. Any events detected may be logged as security events that can be audited.
- Lastly, PBFT (Practical Byzantine Fault Tolerant) consensus is a consensus reached by permissioned validators assuming that the threshold of Byzantine Validators is respected. The protocol gives deterministic commitment if a quorum is reached.

The architecture is however, aware of some restrictions. If an attacker gets his or her hands on a device private key, it's an enormous security concern, as he or she can create seemingly valid signatures. Likewise, a bad sensor can provide false or fake readings and “sign” them. Cryptographic authentication can be used to determine who created the data but does not necessarily determine its physical validity. Also, maintaining the required Byzantine fault threshold is a part of the PBFT security guarantee. When the number of malicious validators becomes more than protocol tolerates, protocol security properties might not be guaranteed.

The off-chain storage architecture adds to the availability dependency by recording only the payload commitment and CID in the blockchain, not the entire payload. A payload may then be temporarily or permanently unavailable, while the blockchain commitment is still valid. Lastly, transaction meta data—including device identifiers, time stamps, counters, transaction links and relationships—can provide information about device activity. Thus, the issues of privacy need to be addressed in conjunction with the integrity, authentication and availability requirements of the proposed architecture.

4. Virtual Simulation Methodology

To implement the above mentioned three-tier framework in a virtual environment, its different approaches, the tools used in it and the different algorithms are described as follows.

4.1 Simulation-based approach

A simulation-based approach is chosen to analyze the proposed three-layered IoT architecture due to the presence of constraints related to the resource-constrained nature of IoT devices, wireless communication, cryptographic authentication, off-chain data storage, intrusion detection, and PBFT consensus algorithm. A virtual test environment allows for varying the number of devices and validators, rate of transactions, cryptographic protocols, network parameters, data storage parameters, and attacks.

The simulation is aimed at modeling the behavior of the proposed architecture rather than replicating it with a real-life environment. This is why processor execution, radio propagation, and energy consumption are modeled explicitly. The obtained outcomes should be analyzed as comparative outcomes since the results depend on specific conditions. Particularly, RSA-2048, ECDSA P-256, and Ed25519 should be compared under the same conditions.

4.2 Simulation Tools and Roles

The simulation environment combines network simulation, constrained-device behavior, cryptographic operations, and statistical analysis.

Table 1: Simulation Tools and their Role

Tool	Role	Implementation focus	Evidence produced
ns-3	Primary network simulator	IEEE 802.15.4, IP/UDP, queues, packet transmission and loss	Latency, throughput, PDR, packet traces
Contiki-NG/Cooja	Constrained-device simulation	IoT processes, application behavior, radio activity and energy estimation	Node events, timing, modeled energy
OMNeT++/INET	Cross-check	Selected network scenarios	Sensitivity and implementation comparison
OpenSSL/Crypto++	Cryptographic implementation	RSA-2048, ECDSA P-256, Ed25519 and SHA-256	Signing/verification time, operation results, key/signature sizes
Python/R	Analysis	CSV processing, statistics and visualization	Tables, graphs and statistical results

ns-3 is the primary network simulator, while Contiki-NG/Cooja represents constrained IoT-node behavior. A common cryptographic interface is used so that RSA-2048, ECDSA P-256, and Ed25519 are evaluated using the same transaction-processing procedure.

4.3 Resource-Constrained Device Model

Two virtual device profiles are considered. Class 1: approximately 32 MHz processor, 32 KiB RAM, and 512 KiB flash and Class 2: approximately 240 MHz processor, 520 KiB SRAM, and 4 MiB flash.

These values are used as resource-model anchors rather than measurements from specific physical devices. They allow the effect of processor capability on signing, verification, transaction processing, and energy consumption to be investigated.

4.4 Cryptographic Algorithms and Cost Model

The cryptographic evaluation compares: (1) RSA-2048, (2) ECDSA P-256 and (3) Ed25519, all three algorithms use the same transaction structure and experimental workload. RSA-2048 provides a conventional public-key signature baseline. ECDSA P-256 provides an elliptic-curve alternative with smaller keys and signatures than RSA-2048. Ed25519 is the default algorithm for the proposed architecture because of its compact keys and signatures and deterministic signing procedure, which is suitable for resource-constrained IoT devices.

1. The simulator separately records signing time at Tier 1 and verification time at Tier 2. This allows the computational cost imposed on constrained devices and validators to be compared independently. When hardware-specific execution timing is unavailable, the operation time is modeled as:

$$Top = Cop \,//\, f$$

where Cop is the modeled cycle count and f is the processor frequency. The corresponding processor energy is estimated as:

$$Eop = Pactive \times Top ,$$

where Pactive is the modeled active power.

4.4.1 Algorithm 1. IoT Device Transaction Creation

1. payload_ref ← storeOffChain(sensor_data)
2. data_hash ← SHA256(sensor_data)
3. tx ← encode(device_id, counter, timestamp, payload_ref, data_hash)
4. tx_hash ← SHA256(tx)
5. signature ← Sign(private_key, tx_hash)
6. signed_tx ← tx || signature
7. transmit(signed_tx)
8. counter ← counter + 1
9. record(signing_time, signing_energy, tx_size)

4.4.2 Algorithm 2. I*Transaction Verification and Payload Integrity*

1. receive signed_transaction
2. verify transaction structure, device ID, timestamp and counter
3. extract transaction and signature
4. start verification timer
5. valid ← Verify(public_key, tx_hash, signature)
6. stop verification timer
7. record verification_time

8. if valid = FALSE:

8.1. reject transaction

8.2. log event and update IDS

8.3. return rejected_transaction

9. payload using CID from Tier 3

10. receive payload

11. calculated_hash ← SHA256(payload)

12. if calculated_hash ≠ expected data_hash:

12.1. reject transaction

12.2. log integrity event

12.3. return rejected_transaction

13. add received transaction packet to mempool

24. return valid_transaction

4.4.3 Algorithm 3. Payload Storage and CID Generation

1. receive sensor_payload

2. validate payload

3. store payload

4. CID ← generateContentIdentifier(payload)

5. index(CID, payload)

6. return CID

4.4.4 Algorithm 4. Payload Retrieval and Integrity Verification

1. receive CID request

2. locate payload

3. retrieve payload

4. calculated_hash ← SHA256(payload)

5. if calculated_hash = expected_hash:

5.1. return payload

6. else:

6.1. reject payload

4.4.5 Algorithm 5. PBFT Consensus Round

1. if local_validator = primary(view, sequence):

1.1. B ← assembleBlock(mempool, batch_limit)

1.2. h ← SHA256(serialize(B))

1.3. broadcast(PRE-PREPARE, view, sequence, h, B)

2. upon valid PRE-PREPARE:

2.1. verify block structure and primary information

2.2. broadcast(PREPARE, view, sequence, h)

2.3. collect matching PREPARE messages

3. if prepare quorum is reached:

3.1. broadcast(COMMIT, view, sequence, h)

4. collect matching COMMIT messages

5. if commit quorum ≥ 2f + 1:

5.1. commit B

5.2. store block and update blockchain state

5.3. generate Merkle proof

5.4. send block confirmation and Merkle proof to Tier 1

5.5. update Tier 3

6. if rejection quorum is reached:

6.1. reject B

6.2. update Tier 3

7. if timeout:

7.1. broadcast VIEW-CHANGE

7.2. advance view

4.4.6 Algorithm 6. Intrusion Detection System

1. receive transaction or consensus event

1.1. if event is PRE-PREPARE:

1.2. key ← (validator_id, view, sequence)

1.3. if key exists in proposal_history:

1.4. if stored_hash ≠ incoming_hash:

1.5. generate EQUIVOCATION alert

1.6. record evidence

1.7. return IDS_alert

1.8. else:

1.9. proposal_history[key] ← incoming_message

2. if event is a transaction:

2.1. c ← transaction.counter

2.2. cmax ← highest_accepted_counter[device_id]

2.3. if c ≤ cmax:

2.4. generate REPLAY alert

2.5. record evidence

2.6. return IDS_alert

2.7. update device transaction history

3. if abnormal device identity or key behavior is detected:

3.1. generate SYBIL alert

3.2. record evidence

3.3. return IDS_alert

4. if valid transaction remains excluded from consecutive block opportunities:

4.1. missed ← update_censorship_counter(transaction)

4.2. if missed ≥ k:

4.3. generate CENSORSHIP alert

4.4. record evidence

4.5. return IDS_alert

5. record normal event

6. return NULL

4.5 Experimental Factors and Scenarios

Table 2: Simulation Factors and Scenarios

Factor	Experimental levels	Purpose
IoT devices	10, 50, 100, 250, 500	Scalability
Validators	4, 7, 10, 13	PBFT sensitivity
Signature algorithm	RSA-2048, ECDSA P-256, Ed25519	Algorithm comparison
Topology	Star, Mesh	Network sensitivity
Traffic	1, 5, 10 tx/min/device	Load sensitivity
Radio	IEEE 802.15.4, 250 kbps	Wireless baseline
Block batch size	128, 256, 512, 1024	PBFT batching
Run duration	60, 120, 300 min	Performance analysis
Attack	Replay, censorship, equivocation, Sybil	Security evaluation

Each cryptographic algorithm is evaluated using the same device count, traffic rate, payload size, topology, validator configuration, and simulation seed.

5. Results and Discussion

The results obtained from proposed framework are discussed in two parts namely - Performance evaluation and security evaluation.

5.1 Performance evaluation of proposed framework

The evaluation will consider three public key cryptographic (PKC) schemes, such as ECDSA P-256, Ed25519, and RSA-2048, considering various numbers of IoT devices. The metrics that have been selected include signing time, verification time, average energy consumption, average transaction latency, and transaction throughput.

5.1.1 Cryptographic Signing Time

The signing time is the time taken by an IoT device to create a digital signature for the transaction prior to sending it to the Tier-2 gateway. The transaction signing happens at Tier-1, which has limited resources, so it would be better if it could be signed quicker. In Table 3, it is shown that the signing time of Ed25519 for 10 devices is 0.2213 ms while ECDSA takes 0.4966 ms and RSA takes 4.1144 ms. As a result, the Ed25519 requires about half the signing time, as well as significantly less than RSA.

From Table 3, for 20 devices, Ed25519 records a signing time of 0.2567 ms, compared with 0.3875 ms for ECDSA and 4.1370 ms for RSA. Similarly, for 50 devices, the signing time is 0.2620 ms for Ed25519, 0.4104 ms for ECDSA, and 4.4466 ms for RSA. From Table 3, for 100 devices, Ed25519 requires 0.2747 ms, while ECDSA and RSA require 0.4052 ms and 4.7952 ms, respectively. For 200 devices, the corresponding signing times are 0.2907 ms for Ed25519, 0.4170 ms for ECDSA, and 3.8942 ms for RSA. Lastly, from Table 3, Ed25519 has the lowest signing time for all of the configurations evaluated, while ECDSA shows the highest signing overhead, and RSA the highest signing time.

Table 3: Different performance metrics to evaluate Propose frame work

No. of device	Metric	ECDSA	ED25519	RSA
10	Signing time (ms)	0.4966	0.2213	4.1144
	Verification time (ms)	0.5103	0.4495	0.2882
	Avg. Energy Consumption (mJ)	0.0318	0.014	0.2575
	Avg. Latency (ms)	0.6673	0.5793	0.4087
	Throughput (Tx/Second)	993.19	1490.7	227.14
20	Signing time (ms)	0.3875	0.2567	4.137
	Verification time (ms)	0.5113	0.5545	0.2828
	Avg. Energy Consumption (mJ)	0.0252	0.0163	0.2638
	Avg. Latency (ms)	0.6854	0.7509	0.4034
	Throughput (Tx/Second)	1112.69	1232.68	226.25
50	Signing time (ms)	0.4104	0.262	4.4466
	Verification time (ms)	0.5498	0.5416	0.2955
	Avg. Energy Consumption (mJ)	0.0263	0.0166	0.2808
	Avg. Latency (ms)	0.8904	0.7445	0.4295
	Throughput (Tx/Second)	1041.44	1244.31	210.88

100	Signing time (ms)	0.4052	0.2747	4.7952
	Verification time (ms)	0.5406	0.5615	0.3021
	Avg. Energy Consumption (mJ)	0.0259	0.0174	0.3034
	Avg. Latency (ms)	0.9107	0.8428	0.4528
	Throughput (Tx/Second)	1057.26	1195.93	196.18
200	Signing time (ms)	0.417	0.2907	3.8942
	Verification time (ms)	0.5557	0.5717	0.2606
	Avg. Energy Consumption (mJ)	0.0264	0.0184	0.248
	Avg. Latency (ms)	1.1896	1.1912	0.6949
	Throughput (Tx/Second)	1028.05	1159.61	240.69
500	Signing time (ms)	0.333	0.245	10.6411
	Verification time (ms)	0.4552	0.4766	0.5328
	Avg. Energy Consumption (mJ)	0.0211	0.0156	0.6771
	Avg. Latency (ms)	3.7943	3.1521	65.3179
	Throughput (Tx/Second)	1268.75	1385.83	89.49

The graph of the three PKC algorithms has been shown in Figure 5 . Based on Figure 5, it can be seen that Ed25519 is by far the slowest signing time for the curves tested across the populations of devices tested and that large differences can be seen in the signing time of the various curves on RSA. ECDSA is still in the range between Ed25519 and RSA. Hence, as seen from Table 3 and Figure.1, Ed25519 is the most efficient algorithm for signing transactions in the Tier-1 IoT device layer.

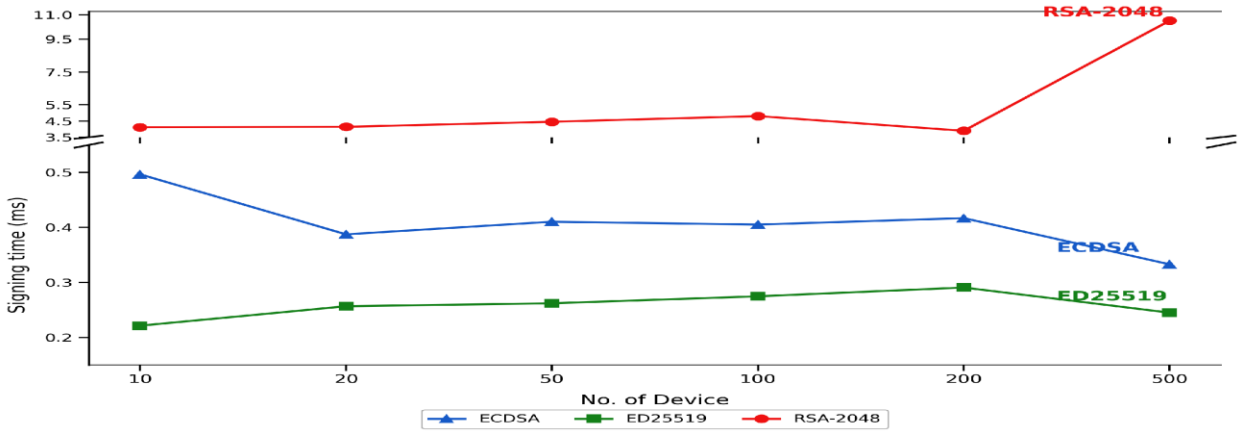

Figure 5: Signing Time comparison (ms)

5.1.2 Cryptographic Verification Time

Verification time is the amount of time needed for the Tier-2 gateway or validator to verify the received digital signature from the IoT device. The gateway has to be able to process a number of transactions simultaneously, necessitating efficient verification.

From Table 3, for 10 devices, the verification time is 0.5103 ms for ECDSA, 0.4495 ms for Ed25519, and 0.2882 ms for RSA. Thus, the lowest verification time is realized with RSA in this configuration. From Table 3, for 20 devices, ECDSA requires 0.5113 ms, Ed25519 requires 0.5545 ms, and RSA requires 0.2828 ms. Similarly, for 50 devices, the verification times are 0.5498 ms for ECDSA, 0.5416 ms for Ed25519, and 0.2955 ms for RSA. From Table 3, for 100 devices, the verification time is 0.5406 ms for ECDSA, 0.5615 ms for Ed25519, and 0.3021 ms for RSA. For 200 devices, the corresponding values are 0.5557 ms, 0.5717 ms, and 0.2606 ms for ECDSA, Ed25519, and RSA, respectively. From Table 3, for 500 devices, ECDSA takes 0.4552 ms, Ed25519 takes 0.4766 ms and RSA takes 0.5328 ms. Therefore, in most of the small and medium-size devices, the verification time is minimum for RSA but not in the case of 500 devices.

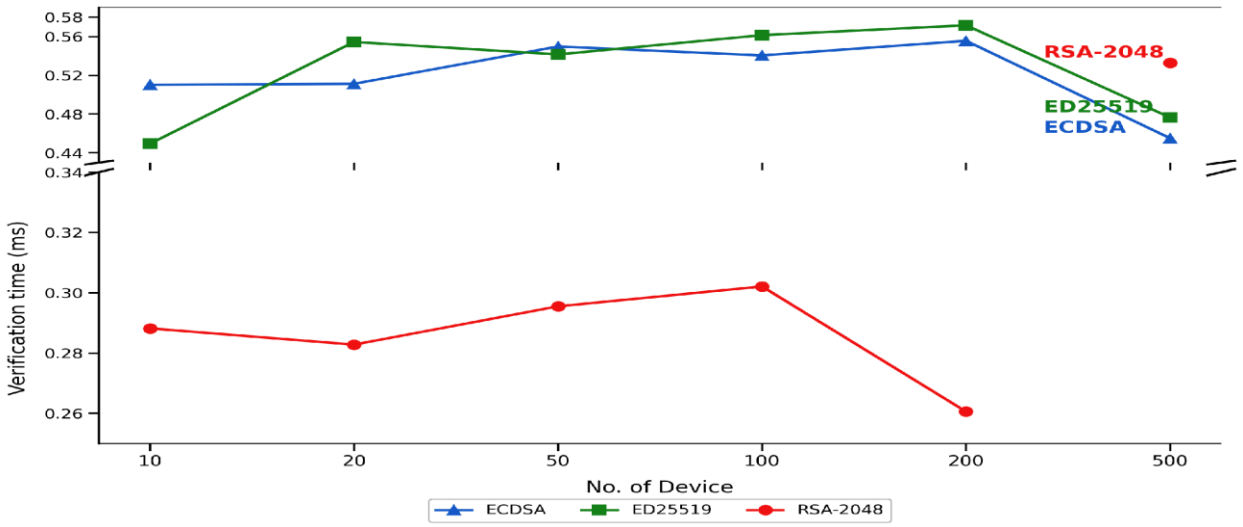

Figure 6: Verification Time Comparison (ms)

The result of the verification-time comparison is shown in figure 6. The verification time for ECDSA and Ed25519 is close to each other in the examined network sizes and for RSA signature the time remains relatively short in most network sizes (Figure 6). Unfortunately, in IoT scenarios, where the communication devices are resource constrained, the most suitable PKC algorithm is not defined by the verification time, since the signing is done on the resource constrained IoT devices.

Hence, from Table 3 and Figure 6, it can be seen that the length of time for verification is an edge that favors RSA in some configurations, but when all factors are taken into account, such as signing cost, energy consumption, latency, and throughput, then Ed25519 is more appealing to the proposed framework.

5.1.3 Average Energy Consumption

The supply of energy resources can be limited for IoT devices and therefore average energy consumption is an important measure for the proposed framework. Reducing the cryptographic energy consumption can also decrease stress on Tier-1 devices.

For 10 devices, it can be seen from Table 3 that the average energy of Ed25519 is 0.0140 mJ, while the average energy of ECDSA is 0.0318 mJ and the average energy of RSA is 0.2575 mJ. Consequently, Ed25519 requires much less energy than ECDSA and RSA. From Table 3, for 20 devices, Ed25519 consumes 0.0163 mJ, compared with 0.0252 mJ for ECDSA and 0.2638 mJ for RSA. For 50 devices, the energy consumption is 0.0166 mJ for Ed25519, 0.0263 mJ for ECDSA, and 0.2808 mJ for RSA. From Table 3, for 100 devices, Ed25519 consumes 0.0174 mJ, whereas ECDSA and RSA consume 0.0259 mJ and 0.3034 mJ, respectively. For 200 devices, the corresponding values are 0.0184 mJ, 0.0264 mJ, and 0.2480 mJ. From Table 3, the average energy consumption of Ed25519 is 0.0156mJ

while for ECDSA and RSA, it requires 0.0211mJ and 0.6771mJ respectively for 500 devices. This gap is greater in the case of RSA at 500 devices.

This average energy consumption of the 3 algorithms is presented in Figure 7. In Figure (7), the extrema of the energy consumptions of all public-key schemes are about the same, with the lowest result being from Ed25519 and the highest from RSA.

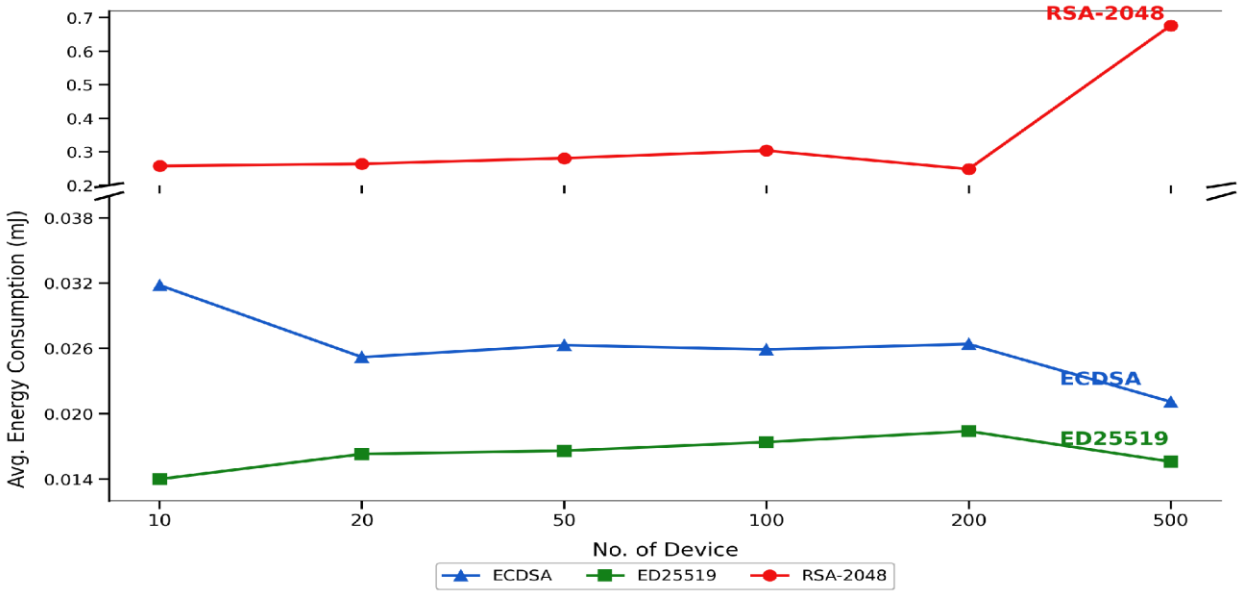

Figure 7: Avg. Energy Consumption Comparison (mJ)

From the above (Table 3 and Figure 7) it is observed that the energy consumption for the crytocrafted block using Ed25519 is the lowest among the above algorithms. The result is useful for the use of Ed25519 on Tier-1 IoT devices which have limited ways to compute on or power. The energy values should however be considered as simulation/model based values instead of hardware values, in order to respect that distinction between energy values and battery measurements as explicitly recommended by the provided methodology.

5.1.4 The Average Transaction Latency

Average transaction latency is the average time taken for a transaction to go through the proposed framework. Cryptographic processing, communication, queueing, gateway processing, and other network operations can impact the latency.

From Table 3, for 10 devices, the average latency is 0.6673 ms for ECDSA, 0.5793 ms for Ed25519, and 0.4087 ms for RSA. As a result, this configuration has the lowest latency in this context because it is the only one without the transmission of the PCS file. For 20 devices, the average latency of ECDSA is 0.6854 ms, that of Ed25519 is 0.7509 ms, and that of RSA is 0.4034 ms from Table 3. For 100 devices, the reported latency is 0.9107 ms for ECDSA, 0.8428 ms for Ed25519, and 0.4528 ms for RSA. From Table 3, for 200 devices, the average latency is 1.1896 ms for ECDSA, 1.1912 ms for Ed25519, and 0.6949 ms for RSA. At 500 devices, the latency increases to 3.7943 ms for ECDSA, 3.1521 ms for Ed25519, and 65.3179 ms for RSA. The 50 device values shown in Table 3 must be confirmed prior to publication of the final values. The reported values for ECDSA (1041.44ms), Ed25519 (1244.31ms) and RSA (210.88ms) differ significantly from the other per-configuration values. These values may be due to a formatting or decimal placement problem but they should not be assumed to be correct for the purposes of correcting them with the output of the original simulation.

The comparison of latencies of the three PKC algorithms is shown in figure 8. As seen from Figure 8, latency tends to rise with the number of devices, reflecting the impact of the growing load and contention in the network. The very high RSA latency with 500 devices points to a high scalability restriction with reported configuration.

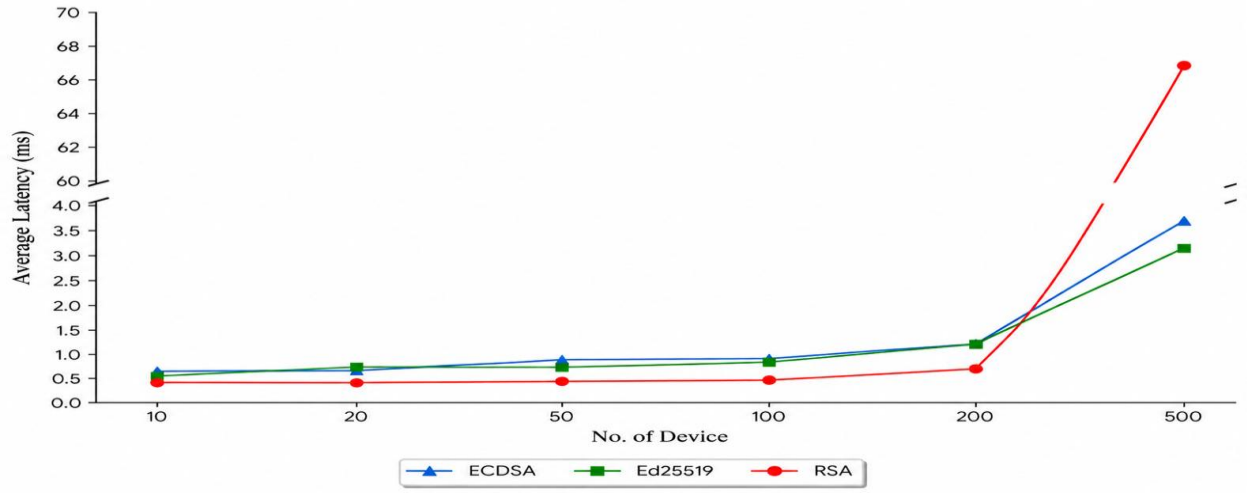

Figure 8: Average Latency Comparison (ms)

Thus, as can be seen in Table 3 and Figure 4, for larger numbers of devices ED25519 has relatively low latency, whereas RSA suffers from a significant performance deprecation at 500 devices. This illustrates that the network and processing effects need to be taken into account when assessing cryptographic selection as well as cryptographic execution time. The provided methodology also focuses on the latency component of the analysis in terms of processing, network, propagation, and queuing.

5.1.5 Transaction Throughput

Throughput is the number of transactions the proposed framework can process per second. A higher throughput means that the framework is able to process more transactions.

Table 3 indicates that 10 devices can achieve 1490.70 Tx/s with Ed25519, 993.19 Tx/s with ECDSA and 227.14 Tx/s with RSA. Hence, the 10-device configuration has the highest throughput with Ed25519. From Table 3, for 20 devices, the throughput of Ed25519 is 1232.68 Tx/s, compared with 1112.69 Tx/s for ECDSA and 226.25 Tx/s for RSA. From Table 3, for 100 devices, Ed25519 achieves 1195.93 Tx/s, while ECDSA achieves 1057.26 Tx/s and RSA achieves 196.18 Tx/s. For 200 devices, the throughput values are 1159.61 Tx/s for Ed25519, 1028.05 Tx/s for ECDSA, and 240.69 Tx/s for RSA. Finally, from Table 3, for 500 devices, Ed25519 achieves 1385.83 Tx/s, ECDSA achieves 1268.75 Tx/s, and RSA achieves only 89.49 Tx/s. For 50 devices, there is no throughput value for this generally given in Table 3, and so, no value is assumed.

Figure 9 shows the throughput comparison. Overall, Ed25519 has the highest throughput on all the reported configurations compared to ECDSA and RSA (which tends to be significantly slower).

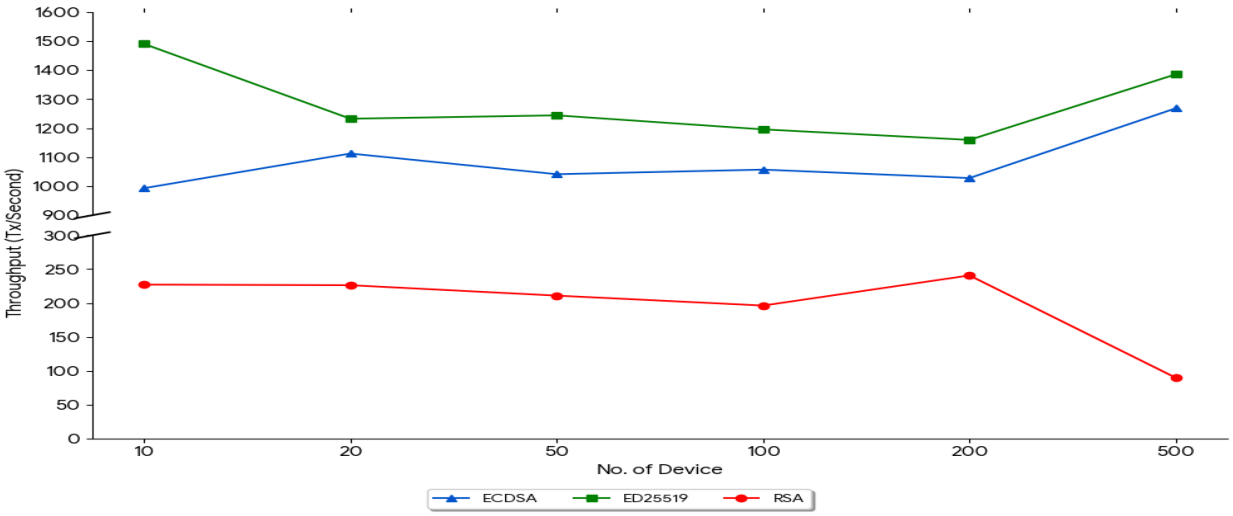

Figure 9: Throughput Compression

Thus, based on Table 3 and Figure 9, Ed25519 represents the best performance in coping with transactions among the three tested PKC operations. The low signing time and its energy usage make it more suitable for the proposed IoT–blockchain framework, as it has a higher throughput.

5.1.6 Performance Summary

Table 3 shows that the performance of the proposed framework is different based on the PKC algorithm used. It has the lowest signing time and average energy consumption among all the configurations evaluated, as Ed25519. While RSA offers low verification time in a few configurations, ECDSA offers relatively competitive signing and verification performance with significantly higher signing and energy overhead. The signing-time overheads in Figure 5 show that Ed25519 is least susceptible to such overheads, making it more suited to the resource-constrained Tier-1 devices. RSA offers lower verification time for the majority of the smaller and medium-sized configurations as shown in Figure 6, but this drops off at 500 devices. From the above figure, it can be seen that Ed25519 is the least energy consuming one while RSA is the highest energy consuming among all. The fact that the transaction signing functionality is executed on constrained devices is important in the context of IoT environments.

The latency results indicate from Figure 8 that having more devices can cause the transaction processing delay to increase because there are more devices to handle the transaction as well as network contention. On the 500-devices-under-test case, the RSA latency is very high compared to ECDSA and Ed25519. Figure 9 shows that Ed25519 has the highest throughput and is followed by ECDSA, with the results for RSA showing significantly lower throughput in most device populations.

From the overall results in Table 3 and Figures 5–9, it can be concluded that the proposed framework has the best performance considering the use of Ed25519. Its benefits in signing time, energy consumption and throughput plays a crucial role especially for the Tier-1 resource constrained layer of the IoT. ECDSA is a good alternative as it has relatively low computational requirements and has good throughput. While RSA-2048 is beneficial in verification time in a number of configurations, it has higher signing costs, consumes more energy and shows poor scalability when performing the workload reported.

The result also confirms the proposed three-tier architecture, where Tier 1 devices are used to create transactions and Tier 2 is comprised of a gateway and a validator layer to carry out the heavier computation for verification and PBFT. By decoupling the validator (central) layer from the constrained devices, the amount of consensus communication per IoT population can be decreased.

5.2 Security Evaluation of Proposed Framework

The proposed blockchain-based framework for resource-constrained IoT networks under different device populations and representative attack scenarios. The evaluation considers RSA, ECDSA, and Ed25519 as interchangeable cryptographic configurations and examines throughput, transaction success rate, signing time, verification time, energy consumption, and transaction size. In parallel, the IDS is evaluated against equivocation, censorship, replay, and Sybil flooding using true-positive rate (TPR), false-positive rate (FPR), F1-score, and detection latency.

The results presented in Table 4 demonstrate the scalability and cryptographic cost of the framework from 10 to 500 devices. Table 5 summarizes the attack-specific IDS performance. Together, the results indicate that the proposed architecture can maintain high transaction reliability under increasing device populations while providing effective detection of the considered security threats. The results also show that the choice of cryptographic algorithm has a significant effect on computational and energy overhead, whereas IDS performance is primarily determined by the characteristics of the attack and the evidence available to the detection mechanism.

5.2.1 Threat Model

The threat model assumes that an adversary can observe wireless communication, inject or modify traffic, replay previously observed transactions, compromise device credentials, or control a subset of validator nodes. The adversary may therefore attempt to compromise data integrity, availability, or consensus. The PBFT security assumption is maintained such that the adversary does not control more than one-third of the validator committee. The security evaluation focuses on authentication, integrity, replay resistance, attack detection, and consensus resilience. Confidentiality is considered separately because blockchain provides integrity and traceability rather than confidentiality of ledger contents.

The attack surface is divided into four layers. At the device layer, threats include key compromise, ECDSA nonce reuse or nonce-generation bias, malicious sensor values, and replay. At the network layer, packet injection, flooding, and selective interference are considered. At the consensus layer, delayed proposals, repeated view changes, censorship, and equivocation are considered. At the storage layer, the framework considers payload unavailability and inconsistencies between off-chain data and on-chain SHA-256 commitments. This layered model enables the framework to associate security events with their source while also considering their impact on other layers. For example, a compromised device credential can result in authenticated but malicious transaction injection, while validator censorship can affect transaction availability without producing an invalid block.

5.2.2 ECDSA Nonce Reuse

ECDSA requires a unique and sufficiently unpredictable nonce for each signature. Nonce reuse or significant bias can expose the private signing key and therefore represents an important device-level threat. The proposed framework considers this vulnerability by monitoring abnormal signature-generation behavior and providing a mechanism for suspicious devices to be quarantined. The purpose of the nonce-reuse experiment is to verify whether the framework can identify the conditions associated with insecure ECDSA signing rather than to claim a specific real-world key-recovery probability without physical implementation. The evaluation can record the number of signatures generated before an anomaly is identified, whether an IDS alert is produced, and whether subsequent transactions are accepted or rejected.

Ed25519 provides an important alternative because its deterministic signing process removes the same dependence on externally generated per-signature randomness. The performance results in Table 4 further support this choice. At 500 devices, Ed25519 requires only 0.240 ms for signing compared with 0.312 ms for ECDSA and 11.350 ms for RSA. Thus, Ed25519 provides both a reduced nonce-generation dependency and substantially lower signing overhead. The energy results reinforce this observation. At 500 devices, Ed25519 consumes approximately 0.015 mJ per transaction, compared with 0.020 mJ for ECDSA and 0.721 mJ for RSA. These characteristics make Ed25519 particularly suitable for resource-constrained IoT endpoints. Future experiments should compare deterministic

Ed25519 with ECDSA using hardware-backed entropy to evaluate the effects of ideal and degraded randomness under realistic device conditions.

Table 4: Different Performance metrics for security evaluation of IDS

No. of Devices	Algorithm	Throughput (tx/s)	IDS detection Success Rate (%)	Signing Time (ms)	Verification Time (ms)	Energy/Tx (mJ)	Tx Size (B)
10	RSA	21.13	87.38	4.51	0.417	0.283	285.35
	ECDSA	20.81	86.03	0.324	0.413	0.02	100.39
	Ed25519	18.23	74.98	0.222	0.479	0.014	93.42
20	RSA	41.16	92.95	3.55	0.266	0.223	285.36
	ECDSA	42.21	92.91	0.38	0.501	0.024	100.41
	Ed25519	38.45	85.97	0.243	0.513	0.015	93.4
50	RSA	104.49	97.19	4.02	0.269	0.256	285.36
	ECDSA	105.09	97.12	0.378	0.505	0.024	100.38
	Ed25519	103.01	94.25	0.236	0.495	0.015	93.4
100	RSA	208.46	98.53	4.08	0.28	0.259	285.37
	ECDSA	213.14	98.45	0.37	0.498	0.024	100.4
	Ed25519	207.52	96.96	0.242	0.499	0.015	93.38
200	RSA	422.98	99.16	3.33	0.24	0.211	285.39
	ECDSA	419.38	99.23	0.355	0.486	0.023	100.38
	Ed25519	424.15	98.48	0.244	0.499	0.015	93.4
500	RSA	782.88	99.75	11.35	0.546	0.721	285.18
	ECDSA	**1048.55**	**99.74**	0.312	0.418	0.02	100.38
	Ed25519	1037.08	99.37	**0.24**	**0.4**	**0.015**	**93.38**

5.2.3 Replay Protection

Replay protection is based on a monotonically increasing transaction counter associated with the device signature. The gateway maintains the highest accepted counter value for each registered device. Transactions containing a counter value lower than or equal to the stored value are rejected unless they belong to an explicitly defined recovery procedure.

The IDS results in Table 5 indicate strong replay-detection performance, with a 94.1% TPR, 2.6% FPR, and 94.0% F1-score, with a detection latency of 135 ms. These results indicate that the combination of transaction sequencing and IDS monitoring can identify the majority of replay attempts while maintaining a relatively low false-positive rate. The remaining undetected events demonstrate that replay protection should not rely solely on the IDS. The transaction counter provides a preventive mechanism, while the IDS provides an additional detection and response layer. This distinction is important because a replayed transaction may be rejected successfully even when the IDS does not classify the event as an attack.

Future experiments should therefore report separately the number of replay packets rejected by the counter, replay packets accepted by the system, IDS alerts, false alarms, and detection latency. This separation would provide a clearer distinction between attack prevention and attack detection.

Table 5: Security Metrics of propose framework

Attack	TPR	FPR	F1	Detection latency
Equivocation	97.8	0.8	98.5	110
Censorship k=2	86.9	6.4	85.4	410
Censorship k=3	80.7	4.1	82.6	610
Replay	94.1	2.6	94	135
Sybil flood	91.5	3.8	91	260

5.2.4 Byzantine Equivocation and Censorship

Equivocation is modeled as a Byzantine validator producing two different signed proposals for the same view and sequence number. Honest replicas compare the authenticated proposals and their associated block hashes. Conflicting proposals provide strong evidence of Byzantine behavior because the validator cannot plausibly claim that both proposals represent the same consensus state.

As shown in Table 5, equivocation achieves the strongest IDS performance among the evaluated attacks, with a 97.8% TPR, only 0.8% FPR, and an F1-score of 98.5%. The detection latency is also the lowest among the evaluated attacks at 110 ms. This performance is expected because equivocation generates explicit cryptographic evidence that can be verified by the honest validators. Censorship is more difficult to detect because the absence of a transaction does not necessarily indicate malicious behavior. Packet loss, congestion, queue overflow, routing changes, and delayed communication can produce similar observations. The proposed framework therefore uses a threshold k to identify repeated missed transaction opportunities.

For k=2, the IDS achieves an 86.9% TPR, 6.4% FPR, and 85.4% F1-score, with a detection latency of 410 ms. Increasing the threshold to k=3 reduces the FPR to 4.1%, but the TPR decreases to 80.7%, the F1-score decreases to 82.6%, and detection latency increases to 610 ms. This demonstrates the expected sensitivity-specificity trade-off. Therefore, k=2 provides faster and more sensitive censorship detection, whereas k=3 provides greater tolerance to transient network conditions. The appropriate threshold should consequently be calibrated using benign traces containing realistic packet loss, congestion, route changes, and delayed transactions.

5.2.5 Sybil and Flooding Attacks

The Sybil threat assumes that an adversary attempts to create or control multiple logical identities. The permissioned registration mechanism reduces this risk by requiring device authentication before an identity is admitted to the system. A lightweight computational puzzle or rate limiter may additionally be used to increase the cost of large-scale identity creation and transaction flooding without imposing excessive computational requirements on constrained endpoints.

The IDS results in Table 5 show a 91.5% TPR, 3.8% FPR, and 91.0% F1-score for Sybil flooding, with a detection latency of 260 ms. These results indicate that the proposed framework can identify a substantial proportion of identity-based and resource-exhaustion anomalies while maintaining a relatively low false-positive rate. The simulator should vary both the number of malicious identities and the transaction generation rate to determine the point at which the validator mempool becomes unstable. Future evaluation should additionally record legitimate identities, malicious identities, transactions per identity, accepted and rejected malicious transactions, IDS alerts, and detection latency. These measurements would allow the Sybil and flooding evaluation to be reproduced using event-level confusion matrices.

5.2.6 Cryptographic and Resource-Efficiency Trade-off

The results in Table 4 demonstrate a clear relationship between cryptographic selection and resource consumption. At 500 devices, ECDSA achieves the highest throughput at 1048.55 tx/s, followed closely by Ed25519 at 1037.08 tx/s, while RSA reaches 782.88 tx/s. Transaction success remains high for all algorithms, reaching 99.75% for RSA, 99.74% for ECDSA, and 99.37% for Ed25519.

Ed25519 provides the strongest overall resource-efficiency characteristics. Its signing time at 500 devices is 0.240 ms, compared with 0.312 ms for ECDSA and 11.350 ms for RSA. Its verification time is also the lowest at 0.400 ms, compared with 0.418 ms for ECDSA and 0.546 ms for RSA. Energy consumption shows an even larger difference. Ed25519 requires only 0.015 mJ per transaction, whereas ECDSA requires 0.020 mJ and RSA requires 0.721 mJ. Transaction size also favors Ed25519, with approximately 93.38 bytes, compared with 100.38 bytes for ECDSA and 285.18 bytes for RSA.

These results indicate that Ed25519 provides the best overall cryptographic configuration for constrained IoT endpoints. ECDSA remains a practical alternative where compatibility with existing systems is important and provides the highest throughput at the largest evaluated device population. RSA remains useful as a baseline but introduces considerably higher computational, energy, and communication overhead.

5.2.7 Overall Security Assessment

The combined results demonstrate that the proposed framework provides a balanced security and performance architecture for constrained IoT environments. The three-tier design limits resource-intensive blockchain operations at the endpoint while allowing gateway validators to perform verification and consensus functions.

The IDS results show particularly strong performance for equivocation and replay detection. Equivocation achieves a 97.8% TPR and 98.5% F1-score, while replay achieves a 94.1% TPR and 94.0% F1-score. Sybil flooding also achieves strong performance with a 91.5% TPR and 91.0% F1-score. Censorship presents the main detection trade-off, where increasing kk reduces false positives but also reduces detection sensitivity and increases latency. From the performance perspective, Ed25519 provides the most favorable resource-security trade-off. At 500 devices, it combines 1037.08 tx/s throughput, 99.37% transaction success, 0.240 ms signing time, 0.400 ms verification time, 0.015 mJ/transaction, and 93.38-byte transactions. ECDSA achieves slightly higher throughput but with somewhat greater signing and energy costs. RSA has the highest cryptographic and communication overhead. Overall, the results support the use of Ed25519 as the preferred signature scheme for resource-constrained endpoints, while the modular architecture allows RSA and ECDSA to remain available for comparison and compatibility. The security results further demonstrate that attack-specific evaluation provides more useful information than a single aggregate IDS accuracy value because each attack has different evidence, detection difficulty, and response requirements.

5.3 Interpretation of the Framework

The evaluation confirms that blockchain integration in constrained IoT networks is a systems-engineering problem involving cryptography, communication, consensus, storage, and security monitoring. Compact cryptographic signatures reduce endpoint overhead, while PBFT transfers consensus processing to more capable gateway validators. Off-chain storage further reduces ledger size while introducing an additional availability dependency. The results support Ed25519 as the preferred signature configuration for constrained devices. Its low signing time, low energy consumption, small transaction size, and near-ECDSA throughput provide a favorable balance between efficiency and security. RSA and ECDSA remain useful as interchangeable comparison schemes and preserve compatibility with existing cryptographic infrastructures.

SHA-256 provides a compact commitment mechanism for transactions, payloads, and blocks. Its use complements the authentication provided by the public-key layer and allows off-chain data integrity to be verified through on-chain commitments.

5.4 Performance Trade-offs

The performance results show that cryptographic cost is only one component of end-to-end system performance. As the number of devices increases, communication, queueing, retransmissions, and consensus processing become increasingly important. At 500 devices, ECDSA and Ed25519 exceed 1000 tx/s, while RSA remains below 800 tx/s, demonstrating the increasing impact of cryptographic and communication overhead at scale.

Energy consumption shows a similar trade-off. Ed25519 significantly reduces computational energy compared with RSA, but total IoT energy consumption also depends on wireless communication, retransmissions, and gateway processing. Consequently, optimization should consider the complete system rather than cryptographic operations in isolation.

The reported results should be interpreted according to the evidence source associated with each metric. The cryptographic and system-level values in Table 4 represent the evaluated framework configuration, while the attack-level values in Table 5represent IDS scenario results. Neither table should be interpreted as physical battery or radio measurements unless validated on hardware. External validity remains limited because real IoT deployments introduce wireless interference, propagation effects, processor-specific behavior, temperature variation, clock drift, and implementation-dependent cryptographic costs. The current evaluation therefore provides a controlled basis for comparison rather than a complete representation of a physical deployment.

5.1 6. Research Implications and Limitation

The framework demonstrates that constrained IoT devices can participate in blockchain-based systems without performing the complete workload of conventional blockchain nodes. Lightweight endpoint processing combined with gateway-based consensus provides a practical approach for reducing resource requirements. The modular cryptographic and IDS architecture also provides a flexible platform for evaluating alternative signature schemes and security mechanisms without redesigning the complete network. This is particularly valuable for future studies involving post-quantum authentication, alternative consensus mechanisms, and adaptive IDS techniques.

The primary limitation is the absence of complete physical testbed validation. The system-level performance results should therefore not be interpreted as direct battery-life, processor-cycle, or radio measurements from specific IoT hardware. The evaluation is also dependent on the assumptions of the communication, energy, queueing, and consensus models. PBFT may exhibit different scalability characteristics from other consensus mechanisms. Similarly, the threshold-based censorship detector may require recalibration under different wireless conditions or adaptive attack strategies.

A further limitation is that the IDS results represent attack scenarios summarized through TPR, FPR, F1-score, and detection latency. Future work should validate these results using independently generated benign and malicious traces and event-level confusion matrices.

5.2 7. Future Scope

Future work will extend the framework from the current evaluation environment toward a reproducible ns-3 and Cooja/Contiki-NG implementation. The same device populations, traffic rates, topologies, validator configurations, and attack scenarios should be reproduced to establish trace-level experimental validation. Realistic wireless parameters, including path loss, CSMA/CA contention, retransmissions, RPL routing, interference, and packet loss, should then be introduced. This will enable evaluation of their effects on throughput, latency, energy consumption, and IDS accuracy.

The consensus layer can be extended beyond PBFT to mechanisms such as Raft and HotStuff to compare scalability and communication overhead. The cryptographic layer can also be extended to post-quantum algorithms such as ML-DSA and Falcon, including hybrid authentication configurations.

Finally, hardware validation using representative constrained devices and gateway platforms should be performed. Measured cryptographic execution time, radio energy, packet loss, and end-to-end latency can then be compared with the current results. IDS evaluation should additionally use multiple independent random seeds and

report confidence intervals and effect sizes. These extensions will provide a direct path from the current framework evaluation to empirical validation.

5.3 8. Conclusion

This study presented a three-tier blockchain framework for resource-constrained IoT networks that combines lightweight IoT clients, gateway-based validation and PBFT consensus, off-chain storage, public-key authentication, SHA-256 commitments, and an intrusion detection mechanism. The framework was evaluated using RSA, ECDSA, and Ed25519 across device populations ranging from 10 to 500 nodes, considering both resource-performance metrics and representative security attacks.

The performance evaluation demonstrates clear differences among the cryptographic schemes. At 500 devices, ECDSA achieved the highest throughput of 1048.55 tx/s, closely followed by Ed25519 at 1037.08 tx/s, while RSA achieved 782.88 tx/s. Transaction success remained above 99% for all three configurations. Ed25519 provided the strongest resource efficiency, requiring only 0.240 ms signing time, 0.400 ms verification time, 0.015 mJ per transaction, and 93.38-byte transactions. In comparison, RSA required 11.350 ms for signing, 0.721 mJ per transaction, and approximately 285 bytes per transaction. The performance evaluation demonstrates that the choice of cryptographic algorithm has a significant impact

The security evaluation also demonstrates strong IDS performance across the considered attacks. Equivocation achieved the highest performance with 97.8% TPR and 98.5% F1-score, while replay achieved 94.1% TPR and 94.0% F1-score. Sybil flooding achieved 91.5% TPR and 91.0% F1-score. Censorship showed the expected threshold trade-off: k = 2 provided higher detection sensitivity; whereas, k = 3 reduced false positives at the cost of lower detection rate and increased latency.

Overall, the results indicate that Ed25519 provides the most suitable overall security-performance trade-off for resource-constrained IoT endpoints, while ECDSA offers the highest observed throughput at large scale and RSA remains a useful baseline. The proposed three-tier architecture reduces the need for constrained devices to perform blockchain consensus directly and provides a flexible basis for integrating cryptographic and IDS mechanisms.

REFERENCES

1. S. B. Erukala et al., “A secure end-to-end communication framework for cooperative IoT networks using hybrid blockchain system,” Sci. Rep., vol. 15, Art. no. 11077, Apr. 2025, doi: 10.1038/s41598-025-96002-w.
2. K. Mershad, “COSIER: A comprehensive lightweight blockchain system for IoT networks,” Comput. Commun., vol. 224, pp. 125–144, Aug. 2024, doi: 10.1016/j.comcom.2024.06.007.
3. D. Owens, R. El Khatib, M. Bisheh-Niasar, R. Azarderakhsh, and M. Mozaffari Kermani, “Efficient and side-channel resistant Ed25519 on ARM Cortex-M4,” IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 71, no. 6, pp. 2674–2686, Jun. 2024, doi: 10.1109/TCSI.2024.3384414.
4. K. Mershad and O. Cheikhrouhou, “Lightweight blockchain solutions: Taxonomy, research progress, and comprehensive review,” Internet of Things, vol. 24, Art. no. 100984, Dec. 2023, doi: 10.1016/j.iot.2023.100984.
5. A. T., S. Babu, and B. S. Manoj, “A Machine Learning Consensus Based Light-Weight Blockchain Architecture for Internet of Things,” in Proc. 14th Int. Conf. COMmunication Systems & NETworkS (COMSNETS), 2022, pp. 1–6, doi: 10.1109/COMSNETS53615.2022.9668487.
6. D. Reijsbergen, A. Maw, S. Venugopalan, D. Yang, T. T. A. Dinh, and J. Zhou, “Protecting the integrity of IoT sensor data and firmware with a feather-light blockchain infrastructure,” in Proc. IEEE Int. Conf. Blockchain and Cryptocurrency (ICBC), 2022, doi: 10.1109/ICBC54727.2022.9805485.
7. S. Sicari, A. Rizzardi, L. A. Grieco, and A. Coen-Porisini, “Security, privacy and trust in Internet of Things: The road ahead,” Comput. Netw., vol. 76, pp. 146–164, Jan. 2015, doi: 10.1016/j.comnet.2014.11.008.
8. J. Granjal, E. Monteiro, and J. Sá Silva, “Security for the Internet of Things: A survey of existing protocols and open research issues,” IEEE Commun. Surveys Tuts., vol. 17, no. 3, pp. 1294–1312, 2015, doi: 10.1109/COMST.2015.2388550.
9. M. Frustaci, P. Pace, G. Aloi, and G. Fortino, “Evaluating critical security issues of the IoT world: Present and future challenges,” IEEE Internet Things J., vol. 5, no. 4, pp. 2483–2495, Aug. 2018.
10. C. Bormann, M. Ersue, and A. Keränen, “Terminology for constrained-node networks,” RFC 7228, IETF, May 2014.
11. A. Dorri, S. S. Kanhere, and R. Jurdak, “Towards an optimized blockchain for IoT,” in Proc. IEEE/ACM IoTDI, Pittsburgh, PA, USA, 2017, pp. 173–178, doi: 10.1145/3054977.3055003.

12. O. Novo, “Blockchain meets IoT: An architecture for scalable access management in IoT,” IEEE Internet Things J., vol. 5, no. 2, pp. 1184–1195, Apr. 2018, doi: 10.1109/JIOT.2018.2812239.
13. T. M. Fernández-Caramés and P. Fraga-Lamas, “A review on the use of blockchain for the Internet of Things,” IEEE Access, vol. 6, pp. 32979–33001, 2018.
14. S. A. Laskar and J. S. Zaman, “Working Principle and Applications of Blockchain with its Challenges,” Fuzzy Syst. Soft Comput., vol. 19, no. 02(IV), pp. 195–205, Jul.–Dec. 2024.
15. S. A. Laskar, S. J. Abbas, and . S. Zaman, “A Deep Dive into Blockchain Applications, Security Enhancements and Challenges,” Commun. Appl. Nonlinear Anal., vol. 31, no. 8s, pp. 1041–1058, 2024, doi: 10.52783/cana.v31.5766.
16. S. A. Laskar, M. A. Hoque, and J. S. Zaman, “Comparative Analysis of MD5 and SHA-256 Algorithms using SAC and BIC Measurement,” accepted in an International Conference ICCTE -2025, eBook will be published in October, 2026
17. National Institute of Standards and Technology, “Secure Hash Standard (SHS),” FIPS PUB 180-4, Aug. 2015, doi: 10.6028/NIST.FIPS.180-4.
18. S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf
19. N. Gura, A. Patel, A. Wander, H. Eberle, and S. C. Shantz, “Comparing elliptic curve cryptography and RSA on 8-bit CPUs,” in Proc. CHES, Cambridge, MA, USA, 2004, pp. 119–132, doi: 10.1007/978-3-540-28632-5_9.
20. M. Hutter and P. Schwabe, “NaCl on 8-bit AVR microcontrollers,” in Proc. AFRICACRYPT, 2013, pp. 156–172, doi: 10.1007/978-3-642-38553-7_9.
21. H. Fujii and D. F. Aranha, “Efficient Curve25519 implementation for ARM microcontrollers,” in Proc. SBSeg Estendido, 2018, doi: 10.5753/sbseg_estendido.2018.4142.
22. H. Fujii and D. F. Aranha, “Curve25519 for the Cortex-M4 and beyond,” in Progress in Cryptology – LATINCRYPT 2017, LNCS 11368, 2019, pp. 109–127, doi: 10.1007/978-3-030-25283-0_6.
23. S. Josefsson and I. Liusvaara, “Edwards-Curve Digital Signature Algorithm (EdDSA),” RFC 8032, IETF, Jan. 2017.
24. J. S. Zaman and S. A. Laskar, “RSA Cryptosystem: Block Cipher or Stream Cipher,” in Proc. 4th Int. Conf. Computing and Communication Systems (I3CS), Shillong, India, 2023, pp. 386–392, doi: 10.1109/I3CS58314.2023.10127232.
25. X. Wang and H. Yu, “How to break MD5 and other hash functions,” in Advances in Cryptology – EUROCRYPT 2005, R. Cramer, Ed. Berlin, Germany: Springer, 2005, pp. 19–35.
26. V. Klima, “Finding MD5 collisions on a notebook PC using multi-message modification,” IACR Cryptology ePrint Archive, Report 2005/075, 2005.
27. T. R. Henderson, M. Lacage, G. F. Riley, C. Dowell, and J. Kopena, “Network simulations with the ns-3 simulator,” ACM SIGCOMM Demonstration, 2008.
28. A. Dunkels, B. Grönvall, and T. Voigt, “Contiki – a lightweight and flexible operating system for tiny networked sensors,” in Proc. IEEE LCN, Tampa, FL, USA, 2004, pp. 455–462, doi: 10.1109/LCN.2004.38.
29. G. Oikonomou, S. Duquennoy, A. Elsts, J. Eriksson, Y. Tanaka, and N. Tsiftes, “The Contiki-NG open source operating system for next generation IoT devices,” SoftwareX, vol. 18, Art. no. 101089, 2022, doi: 10.1016/j.softx.2022.101089.
30. A. Varga, “OMNeT++,” in Modeling and Tools for Network Simulation, K. Wehrle, M. Günes, and J. Gross, Eds. Berlin, Germany: Springer, 2010, pp. 35–59, doi: 10.1007/978-3-642-12331-3_3.