International Journal of Scientific Engineering and Research (IJSER)
ISSN (Online): 2347-3878
SJIF (2024): 6.623

Blockchain-Based Secure Routing Protocol for IoT Networks

Thejiya V.

[1]Department of Computer Application, Krupanidhi Group of Institutions, Bangalore, India
Email: *thejiya22[at]gmail.com*

Abstract: ***The proliferation of the Internet of Things (IoT) has intensified the need for secure and reliable routing protocols that can operate within the severe resource constraints of IoT devices. Traditional IoT routing protocols (e.g., RPL and AODV) often struggle with security vulnerabilities such as blackhole and wormhole attacks, limited scalability, and trade-offs between overhead and performance. In this paper, we propose a theoretical design for a blockchain-based secure routing protocol tailored to IoT networks. The design integrates a lightweight distributed ledger (using platforms like IOTA's directed acyclic graph or Hyperledger Fabric's permissioned blockchain) to serve as a shared, immutable memory for routing information. We detail the architectural framework, including resource-aware consensus mechanisms and lightweight smart contracts to automate route validation and trust management. Our protocol leverages blockchain's immutability and decentralized consensus to authenticate routing messages, validate route paths, and record reputation metrics for nodes, all while minimizing computational and energy overhead on constrained devices. We present an in-depth comparison with RPL and AODV, demonstrating improved security (resilience against common attacks), enhanced trust and data integrity, and reasonable scalability for IoT deployments. Evaluation criteria are discussed qualitatively, showing that the proposed approach can achieve robust defense against blackhole and wormhole attacks, with manageable latency and energy consumption overhead. We also analyze the strengths and limitations of the design – highlighting how blockchain adds security and transparency at the cost of extra overhead – and outline future research directions to further optimize blockchain-based routing in IoT. The insights and analysis in this paper aim to pave the way for next-generation secure and scalable routing protocols suitable for the emerging IoT ecosystem.***

Keywords: IoT Routing, RPL, AODV, Blockchain, Secure Routing, Distributed Ledger, Smart Contracts, Consensus, IoT Security.

1. Introduction

The Internet of Things (IoT) connects vast networks of resource-constrained devices (sensors, actuators, smart appliances) that communicate to deliver innovative services. At the heart of IoT communication lies the routing protocol, which enables multi-hop data transfer across devices. In low-power and lossy networks (LLNs) typical of IoT, the IPv6 Routing Protocol for Low Power and Lossy Networks (RPL) has become a de facto standard. RPL builds a Destination-Oriented Directed Acyclic Graph (DODAG) topology optimized for energy efficiency and is widely used in applications such as environment monitoring and smart cities. Similarly, in mobile or ad-hoc IoT scenarios, classic MANET protocols like Ad hoc On-Demand Distance Vector (AODV) may be applied due to their reactive route discovery suitable for dynamic topologies. However, like other network protocols, RPL and AODV were not originally designed with strong security against malicious actors, leaving IoT networks vulnerable to a variety of routing attacks.

1.1 Security challenges in IoT routing

IoT devices often operate unattended and communicate wirelessly, making them easy targets for attackers. Routing attacks can degrade network performance or hijack traffic. For instance, RPL is susceptible to rank attacks (a malicious node advertises a fake optimal route by manipulating its rank value, tricking others into routing through it) and blackhole attacks (where a node intentionally drops all packets it forward). A successful rank attack can effectively create a sinkhole, diverting traffic through the attacker, while a blackhole causes denial of service by discarding packets. RPL can also suffer from wormhole attacks, version number attacks, and other exploits of its DODAG maintenance process.

AODV, on the other hand, can fall prey to classic MANET attacks: e.g., in a blackhole attack on AODV, a malicious node can respond to route requests with false RREP messages claiming a short path (often advertising a very low hop-count or high sequence number), then absorb or drop the passing traffic. Without additional security, AODV's route discovery trusts the first RREP, which an attacker can exploit. Similarly, wormhole attacks may occur when two colluding attackers tunnel routing messages between distant parts of the network to create a shortcut that attracts traffic, bypassing legitimate routes. These attacks not only disrupt IoT applications (causing data loss, added latency, or network partitioning) but also can be stepping stones to deeper intrusions (e.g., intercepting sensor data or injecting false data).

1.2 Limitations of traditional countermeasures

Conventional security solutions for routing (such as cryptographic authentication or intrusion detection systems) are often hard to directly apply in IoT settings. IoT nodes typically have limited CPU, memory, bandwidth, and energy supply, so lightweight protocols are required. RPL does define some security modes (pre-installed or authenticated keys for control messages), but in practice these add significant overhead and require key management that might be infeasible for large IoT deployments. Many RPL networks therefore run in unsecured mode, exposing them to attacks. Likewise, secure variants of AODV (e.g., SAODV) use digital signatures and hash chains to authenticate routing messages, but they incur extra transmission overhead and computational cost that battery-powered IoT nodes may not handle well. Research has explored trust-based or reputation-based routing enhancements (where nodes evaluate the trustworthiness of neighbors based on behavior), as well as dedicated detection algorithms for specific attacks. While these approaches can

mitigate certain threats, they often address one attack at a time and may rely on central authorities or assumed trust which counteracts the decentralized nature of ad-hoc IoT networks.

1.3 Blockchain as a potential solution

Blockchain technology offers a promising new foundation to enhance security and trust in distributed networks. A blockchain is essentially a distributed ledger maintained by a network of nodes that reach consensus on the ledger's state. Once data are recorded on the blockchain, they are extremely difficult to tamper with thanks to cryptographic linking of blocks and decentralized consensus. These properties align well with the needs of IoT routing security: immutability can ensure that once a routing decision or network event is logged, it cannot be maliciously altered or repudiated; decentralized consensus can remove reliance on any single trusted node (important since any IoT node could be captured or become malicious); and transparency of the ledger can enable auditability of routing behavior (making it easier to detect misbehavior). By providing a shared, secure "memory" for the network, a blockchain can effectively serve as a trust anchor and coordination mechanism among IoT devices.

Previous research indicates that Blockchain-Based Routing (BBR) frameworks can improve overall routing performance and security by securely storing routing decisions and updates, enabling automatic routing management via smart contracts, providing robust authentication, and even supporting reputation-based or onion-routing schemes. These features directly target the pain points of traditional protocols, suggesting that a well-designed integration of blockchain could yield a secure routing protocol resilient to common attacks.

Despite the appeal, integrating blockchain into IoT routing is non-trivial. Standard blockchain platforms like Bitcoin or Ethereum are far too resource-intensive (in terms of computation, energy, and bandwidth) for IoT devices – e.g. Proof-of-Work consensus would quickly drain battery-powered nodes. The challenge, then, is designing a lightweight blockchain framework and routing protocol co-design that preserves IoT-friendly efficiency while dramatically enhancing security. In this paper, we focus on a theoretical design of such a protocol, called BCR-IoT for convenience (Blockchain-based Secure Routing for IoT). We choose suitable blockchain technologies (e.g., IOTA or Hyperledger Fabric) that are amenable to IoT integration, and we incorporate consensus and smart contract mechanisms optimized for low-power devices.

We compare our proposed BCR-IoT design with traditional protocols (RPL and AODV) in terms of security, scalability, and communication overhead. Our analysis shows that BCR-IoT can significantly bolster security – preventing or mitigating attacks like blackholes and wormholes that easily defeat RPL/AODV – at the cost of a moderate increase in routing overhead. We justify that this trade-off is acceptable in many IoT scenarios, especially as devices become more capable, and that the improved trust and reliability can enable IoT networks to operate in untrusted environments (e.g., multi-party ad-hoc networks) where traditional protocols would be too risky. Finally, we discuss the strengths and limitations of the BCR-IoT approach and outline future research directions, such as further optimizations in consensus algorithms and real-world prototyping.

2. Background and Related Work

2.1 IoT Routing Protocols and Challenges

2.1.1 RPL – IPv6 Routing Protocol for LLNs

RPL is a distance-vector routing protocol optimized for low-power and lossy networks (LLNs), standardized by the IETF for IoT scenarios. RPL organizes nodes into a topology called a DODAG (Destination Oriented Directed Acyclic Graph), rooted at a border router (sink). Each node has a rank (an integer or floating metric) signifying its distance to the root (e.g., hop count, expected transmission count, or other objective function). RPL's operation is largely proactive: the root initiates the network by broadcasting DODAG Information Objects (DIO) that propagate through the network, allowing nodes to join the graph by selecting preferred parents (usually the neighbor with the lowest advertised rank). Periodic trickle-timer mechanisms govern the frequency of control messages (DIO, and DAO for downward routes) to keep overhead low in stable conditions. RPL's strengths include loop avoidance and local repair mechanisms, and as studies show, it tends to be power-efficient and scalable under static conditions – indeed, simulations indicate that RPL outperforms AODV in terms of power consumption, especially as network size increases. RPL's efficient DODAG maintenance yields stable energy usage even as more nodes join, making it suitable for large-scale, mostly-static IoT deployments.

However, RPL faces issues in dynamic or adversarial conditions. Under mobility, RPL's trickle timers and repairs may lag, leading to increased latency or temporary routing loops. More critically, RPL is vulnerable to a range of routing attacks. We highlighted rank attacks and blackhole attacks earlier; additionally, there are sinkhole attacks (malicious node impersonates a root or advertises an attractive route to draw traffic), wormhole attacks (out-of-band tunnel that confuses topology), DODAG inconsistency attacks (spoofed inconsistency messages forcing unnecessary repairs), and routing information replay or DIS flooding (exploiting the DODAG Information Solicitation mechanism to drain nodes' energy). Many of these are exacerbated by RPL's design choices: it trusts routing control messages within the network (unless operating in a secure mode which is rarely enabled due to complexity), and it lacks built-in verification of route optimality or consistency – a malicious node's claims are generally accepted by neighbors in the absence of an external trust mechanism. Research has proposed various countermeasures, such as intrusion detection systems and trust-based RPL enhancements. For example, trust-based RPL variants maintain a reputation score for neighbors and avoid those with low trust to mitigate blackhole/sinkhole attacks. These methods can improve security (one such model improved packet delivery and throughput by >35% while only modestly increasing power use), but they often require complex tuning and are not yet part of the RPL standard.

2.1.2 AODV – Ad hoc On-Demand Distance Vector

AODV is a classic routing protocol initially designed for mobile ad-hoc networks (MANETs), which has also been applied in IoT contexts (particularly IoT networks with dynamic topology or no fixed infrastructure). Unlike RPL, AODV is a reactive protocol: it discovers routes on-demand. When a source node needs a route to a destination, it broadcasts a Route Request (RREQ) which floods through the network. When either the destination or an intermediate node with a fresh route to the destination receives the RREQ, it responds with a Route Reply (RREP) that traces back to the source, establishing the forward path. AODV maintains routes as needed and times them out if unused, and uses Route Error (RERR) messages to notify nodes of link breakages. One key feature is AODV's use of sequence numbers to ensure loop-free and up-to-date routes (the sequence number indicates the freshness of routing information). Because of its on-demand nature, AODV can be more efficient than proactive protocols in networks where traffic is sporadic or the topology changes frequently – it avoids the overhead of maintaining routes that are never used. However, AODV may introduce higher latency for initial packet delivery due to the route discovery process.

In IoT scenarios (e.g., a swarm of drones or vehicles communicating, or a wireless sensor network that isn't easily structured into a tree), AODV can provide flexibility. But similar to RPL, vanilla AODV lacks intrinsic security. It is well documented that AODV is susceptible to routing misuses: a notorious example is the blackhole attack, where a malicious node answers RREQs with bogus RREPs claiming a very short path, then drops all data packets that arrive. Since AODV nodes typically choose the first or best RREP and have no knowledge of the network beyond their neighbors, they can be easily misled by such false replies. Cooperative blackhole (multiple colluding attackers) and gray hole (selective dropping) attacks further challenge AODV's reliability. Another issue is wormhole attacks, in which two distant attackers tunnel RREQ/RREP packets between them: this can create the illusion that two far-apart nodes are neighbors, resulting in routes that pass through the wormhole pair. AODV (and indeed any routing protocol without geographical or temporal checks) has difficulty detecting wormholes because the routing packets themselves do not appear malicious – they arrive with valid data but from unexpected locations. Standard AODV also doesn't authenticate the source of routing messages, making it vulnerable to impersonation or Sybil attacks (an attacker can fabricate multiple identities to influence routing). Researchers have proposed enhancements like Secure AODV (SAODV) which introduces cryptographic signatures for RREQ/RREP and hashing for hop counts. SAODV can thwart impersonation and ensure that route advertisements are authentic, but it still might not fully stop wormholes (as those could forward even authentic packets). More recently, innovative ideas such as the use of blockchain and trust in AODV have been explored. For example, Ran et al. (2021) proposed an improved AODV that uses a blockchain to store node state and deploy smart contracts to select routes meeting QoS constraints, effectively finding multiple disjoint paths (one main, one backup) to improve reliability. Their simulation results showed performance gains especially under malicious conditions, hinting at the potential of blockchain to harden AODV-like protocols.

2.2 Blockchain Technologies for IoT Integration

2.2.1 Blockchain fundamentals

A blockchain is a distributed ledger consisting of an append-only chain of blocks, where each block contains a set of transactions and a cryptographic hash of the previous block. A network of nodes maintains the blockchain through a consensus mechanism that ensures all honest nodes agree on the contents of the ledger. The absence of a centralized authority and the tamper-evident nature of the ledger make blockchains attractive for establishing trust in open networks. Beyond cryptocurrencies, blockchain systems can record arbitrary data with associated smart contracts – essentially code that executes on the blockchain to enforce custom rules or automate operations. Key properties of blockchain relevant to routing are: immutability (once a routing event is recorded, it cannot be altered unnoticed), decentralization (no single point of failure or trust), transparency (important events are visible to all participants, aiding audit and forensics), and authentication (transactions are signed by nodes' private keys, ensuring accountability for actions). These can directly enhance security: for instance, if routing control messages or metrics are stored on a ledger, an attacker cannot forge or tamper with them without detection. Additionally, distributed consensus means an attacker would need to subvert a majority of the network (often economically or computationally infeasible) to corrupt the routing information, dramatically raising the bar for potential attackers compared to targeting a single root or flooding a network with fake messages.

2.2.2 Lightweight and IoT-oriented ledgers

Traditional blockchains like Bitcoin's are too heavy for IoT (in terms of energy for Proof-of-Work and data volume). Fortunately, alternatives more suitable for IoT have emerged. IOTA is one prominent example – it's an open distributed ledger specifically designed for the "Internet of Everything" (IoE) and IoT context. IOTA eschews the conventional linear blockchain in favor of a Directed Acyclic Graph (DAG) called the Tangle. In IOTA, each transaction verifies two previous transactions via a small Proof-of-Work, resulting in a DAG of transactions rather than sequential blocks. This design yields feeless transactions (no miners to pay) and theoretically high scalability as more transactions lead to faster confirmation (through parallelism). For IoT devices, the lack of fees and the ability to perform tiny transactions (even a single sensor reading) are advantageous. However, IOTA's consensus (especially in its early iterations) relied on a Coordinator node for finality, raising questions of centralization – the IOTA Foundation has been working on "coordicide" to remove that. Still, IOTA remains attractive for IoT scenarios like sensor data integrity and lightweight trust, and it can support simple smart contracts through second-layer solutions.

On the other side of the spectrum, Hyperledger Fabric represents a permissioned blockchain approach suitable for controlled IoT environments (e.g., industrial IoT managed by a company or consortium). Hyperledger Fabric is an open-source enterprise-grade distributed ledger platform, which allows networks of known participants (permissioned identities). It features a modular architecture with pluggable

consensus and advanced privacy features like channels (subnets of the blockchain where a subset of nodes can conduct private transactions). Fabric does not require energy-intensive mining; instead, it can use crash fault tolerant or byzantine fault tolerant consensus algorithms (such as Raft or PBFT) to achieve finality in a matter of seconds or less. Smart contracts (chaincode) in Fabric can be written in general-purpose programming languages and are executed by endorsing peers. For IoT, Fabric's permissioned model means that devices or gateways are registered with cryptographic identities by an administrator, which immediately mitigates Sybil attacks (no unverified device can join pretending to be many nodes). Its channel mechanism can confine routing-related transactions to relevant participants, saving bandwidth. Fabric's privacy controls ensure that only authorized nodes see certain data – useful if routing information is sensitive (e.g., could reveal device roles or locations). There have been prototypes where Hyperledger Fabric is integrated with IoT networks for access control and secure data sharing, showing viability in practice. The trade-off is that running even a Fabric client or node may be heavy for a tiny sensor; thus, often IoT architectures with Fabric assume that IoT devices communicate with a more powerful gateway, which in turn interfaces with the blockchain network.

2.2.3 Related work on blockchain-based routing

In recent years, researchers have started exploring blockchain or distributed ledger systems to secure routing in various contexts. A survey by Wijesekara et al. (2023) analyzed dozens of Blockchain-Based Routing (BBR) proposals and found that about 20.5% of these were geared towards IoT networks. These BBR proposals often leverage blockchain in several ways: (1) to store routing information (routes, metrics, topology changes) immutably on a ledger; (2) to use smart contracts for automating routing decisions or enforcing forwarding agreements; (3) to provide inherent authentication and identity management for routers (every node has a blockchain identity, making it harder for an attacker to spoof another node); (4) to implement reputation systems where nodes' trustworthy behavior is recorded and evaluated on-chain; and (5) to create incentive mechanisms (often via micropayments or cryptocurrency) that reward honest routing behavior and penalize malicious acts. For example, Ramezan and Leung (2018) proposed a Blockchain-based Contractual Routing (BCR) protocol for IoT, using Ethereum smart contracts to handle route establishment as a contract between nodes. In their design, intermediate nodes would be compensated for forwarding packets as per the contract, and any misbehavior (like dropping packets) could be detected and lead to loss of reward – aligning economic incentives with truthful routing. Another work, called MARS (Monetized Ad-hoc Routing System), presented a position paper on using cryptocurrency payments to incentivize packet forwarding in MANETs. These approaches indicate that blockchain can introduce a notion of trust and accountability economically, which was never present in traditional ad-hoc routing.

Beyond incentives, purely technical enforcement has been explored. For instance, Chen et al. (2023) developed a blockchain-based RPL optimization to secure and stabilize IoT routing: their method recorded certain routing information on-chain and ensured data integrity and security while actually improving energy consumption and route stability in critical IoT applications. This suggests that, if designed carefully, blockchain integration need not come at a huge cost to performance; it can be leveraged to optimize as well (e.g., by providing a reliable global view of the network state to aid route selection). Additionally, blockchain has been combined with emerging techniques like machine learning for secure routing. A recent framework used Hyperledger Fabric with an ML-based consensus to filter out outlier (potentially malicious) routing data, thereby creating an outlier-aware secure consensus for IoT routing updates.

In summary, the literature highlights that blockchain can secure routing by making the network collectively uphold routing correctness rather than trusting individual nodes. However, a gap remains in fully specifying a practical, end-to-end design that is implementable on constrained devices. Many proposals are at conceptual or simulation stages, and real-world deployments are still rare. In the following sections, we build upon these ideas to describe our unified design for a blockchain-based secure routing protocol in IoT, explicitly addressing the issues of resource constraints, security attack coverage, and comparative performance with existing protocols.

3. Proposed Blockchain-Based Secure Routing Protocol (BCR-IoT)

In this section, we present the design of BCR-IoT, a Blockchain-Based Secure Routing Protocol tailored for IoT networks. The design is theoretical, but we ground it in realistic assumptions about IoT hardware and emerging lightweight blockchain frameworks. We begin with the architecture of the network and system, then detail the route discovery and validation mechanisms that intertwine with the blockchain ledger. We describe the chosen consensus mechanism and how it is optimized for resource-constrained settings, and we discuss the use of lightweight smart contracts to automate trust management and routing decisions. Our design goal is to maximize security and trustworthiness of routing with minimal additional overhead, by intelligently offloading heavy tasks to more capable nodes or distributing them over time.

3.1 Architecture Overview

The proposed architecture consists of three main layers of entities (illustrated in Figure 1): (1) the IoT Devices (sensor/actuator nodes that need routing), (2) the Miner or Validator Nodes (which maintain the blockchain ledger and run most of the blockchain logic), and (3) an optional Gateway/Edge layer that interfaces between the constrained devices and the blockchain network. This architecture leverages the heterogeneity often present in IoT networks – not all nodes are equally resource-constrained. We assume that at least a small subset of the network (it could be dedicated devices, cluster heads, or edge servers) has comparatively higher capabilities (energy, computation, storage) and can shoulder the burden of blockchain operations. Regular IoT devices participate in routing by sending and receiving packets and providing authentication (digital signatures) for their messages, but they do not perform expensive consensus algorithms. Instead, they rely on the miner/validator nodes to add records to the distributed ledger

on their behalf. This division of roles keeps the overhead on tiny devices low, while still achieving a decentralized trust model via the validators.

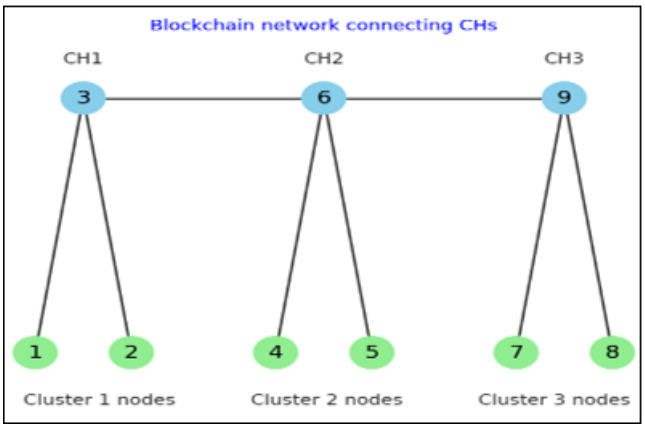

Figure 1: Proposed Architecture overview

We organize IoT devices into clusters, each managed by a more capable cluster head (CH) node (shown in blue). The CHs collectively form a peer-to-peer blockchain network to maintain the ledger of routing information. Regular IoT devices (green nodes) are connected to their local CH, which serves as their gateway to the blockchain. In this design, IoT devices do not communicate with the blockchain directly; instead, CHs verify device identities, collect routing information (such as route requests, acknowledgments, etc.), and publish transactions to the blockchain. The CHs reach consensus on the blockchain state (for example, using a Byzantine fault tolerant algorithm or an IOTA-like gossip), thereby ensuring that all CHs (and thus all clusters) share a consistent, tamper-proof view of network routes and node reputations. This hierarchical approach balances decentralization with efficiency: trust is distributed among multiple CH nodes, and there is no single point of failure, yet individual low-power nodes are spared the cost of running a full blockchain client.

IoT devices and identities: Each participating IoT device is provisioned with a unique cryptographic identity (e.g., an ECC public/private key pair). This identity is used to sign routing messages and transactions, binding actions to the device and preventing impersonation. The identity can be established by a one-time registration on the blockchain (perhaps via the gateway or CH): for example, when a new node joins, its public key and an identifier (like a hash of its MAC or a logical ID) are recorded on the ledger by an authorized entity (this could be done by a system administrator or via a self-registration smart contract if the network allows open joining with certain proofs). We assume a permissioned participation model for simplicity – meaning unknown devices cannot participate in routing until they register – but the consensus is decentralized among the known participants. This addresses Sybil attacks since creating a fake identity would require registering a new key on the blockchain, which the consensus can control or rate-limit. Once identities are set, all routing protocol messages (Route Requests, Replies, etc.) will be signed by the sender's private key. Neighboring devices and CHs will verify these signatures against the sender's public key (retrieved from the blockchain's identity registry). This cryptographic verification ensures authenticity of routing messages: a malicious node cannot masquerade as someone else, and injected fake messages (not signed by any known key) will be discarded as invalid.

Miner/Validator nodes: In our architecture diagram (Figure 1), the cluster heads (CH1, CH2, CH3) act as validator nodes that maintain the blockchain. They form a peer-to-peer network among themselves (shown by the horizontal connections between CHs) and run a consensus algorithm to agree on new blocks. Depending on deployment, these could be special IoT devices with extra resources, or simply a role assigned to a few normal devices. We might also consider that every IoT device above a certain capacity could run the blockchain software – the design is flexible to network size. For clarity, we depict a clustered scenario where each CH represents a group of devices. The blockchain maintained by the CHs contains entries such as: route discovery transactions, route validation records, and node behavior logs (trust scores, misbehavior reports). Because the blockchain state is shared, even if one cluster is under attack, other validators ensure the integrity of data – an attacker would have to corrupt a majority of CHs to subvert the ledger, which is much harder than attacking individual links or nodes as in traditional routing.

Gateway functionality: If a strict clustering is not present, a generic gateway concept can be applied. A gateway (which could coincide with a CH or be another edge device) handles communications between low-power nodes and the blockchain network. IoT devices might talk to a gateway using a lightweight protocol (like BLE, Zigbee, or CoAP/UDP) and the gateway translates and forwards relevant info to the blockchain network (over, say, TCP/IP). The gateway also buffers and aggregates updates to avoid overloading the ledger with frequent small transactions. In many IoT systems (like LoRaWAN or NB-IoT networks), such gateways are naturally present; our protocol can be embedded partly in gateway software to leverage that existing component.

3.2 Route Discovery with Blockchain Integration

3.2.1 Baseline routing approach

Our protocol follows a route-on-demand philosophy similar to AODV, adapted for blockchain support. We choose on-demand routing because it naturally fits an event-driven logging system (blockchain) – routes are discovered and can be logged when needed, rather than attempting to log every periodic update as would happen in a proactive protocol like RPL (which could overwhelm the ledger with constant data). However, we incorporate certain elements of RPL's approach for efficiency, such as preferring reliable links and maintaining some local route caching to reduce the frequency of discoveries.

When a source IoT device (let's call it Node S) has data to send to a destination device Node D and does not have a current route, it initiates a Route Request (RREQ). Node S constructs an RREQ packet containing: its own ID, the destination ID, a route request sequence number (to match replies and prevent replay), and possibly some metrics (e.g., current battery level or required QoS). Crucially, Node S signs the RREQ with its private key. This RREQ is then broadcast to S's neighbors (if multi-hop at device level) or simply sent to its cluster head/gateway (if a star topology per cluster). We

assume neighbors or the CH verify the signature and then process the RREQ.

In a pure peer-to-peer scenario, each receiving node (let's call one Node X) will verify the RREQ's signature (ensuring it indeed originated from S), then check if it has seen this RREQ before (via sequence number and source ID). If not, it records the RREQ (to avoid rebroadcast loops) and appends its own ID to the route record in the RREQ before forwarding it onward. This ID list constitutes the discovered path as the RREQ propagates. Node X then rebroadcasts the RREQ further (or passes it to its CH to broadcast to other clusters). This is similar to AODV's flooding mechanism, but one key difference is that each node signs the portion of the RREQ it forwards or at least signs a statement "I forwarded this RREQ at time T". We can achieve this without excessive overhead by having each node's signature on the accumulated route info or by separate small attestation transactions to the ledger (more on this shortly).

Eventually, the RREQ reaches either the destination D or an intermediate node that has a fresh route to D. Suppose it reaches D. Node D now prepares a Route Reply (RREP). In our design, the RREP will carry the complete route (the sequence of node IDs) from S to D that was accumulated. It also carries the original sequence number from S's request. Node D signs the RREP (ensuring authenticity of the reply), and unicasts it back towards S along the reverse of the discovered path. Since each node in the path knows from the RREQ which neighbor it received the request from, it can forward the RREP along that chain. Alternatively, and more securely in our case, Node D could create a transaction on the blockchain to publish the discovered route. For example, D (or the first CH that sees the RREP) submits a "RouteReply" transaction that includes the hash of the route (or the full route), the source, destination, and a timestamp, all signed by D. This acts as a ledger-record of the route.

3.2.2 Blockchain-based validation of route

Once the route is reported (either via transactions as it was discovered, or via the final RREP transaction), the validator nodes (CHs) engage to validate the route. Validation means checking that: (a) all nodes on the route are legitimate (registered in the blockchain's identity list), (b) the route does not contain any obvious inconsistency (like a loop or a forbidden node), and (c) each hop on the route was actually witnessed by the supposed intermediate node. Condition (c) is novel – how do we ensure each link was genuine? We implement a mechanism wherein intermediate nodes, upon forwarding the RREQ, also post a route witness transaction. Specifically, when Node X forwards a RREQ to Node Y, Node X can submit a tiny signed statement to the blockchain: "X heard RREQ S−>…−>XS->…->X and forwarded to Y at time t". Alternatively, Node X might simply sign the RREQ's content and include it in the RREQ itself (a cumulative signature approach), but that can bloat the packet. Offloading to blockchain means we separate concerns: the network propagates minimal info, and the blockchain carries the heavier audit trail. So, as the RREQ propagates through CHs, those CHs (on behalf of their member nodes) collectively add entries like "Node A -> Node B neighbor relation observed for RREQ #123" onto the ledger. This creates a chain of custody for the route discovery.

When Node D's RouteReply transaction is added, validators cross-check it against these prior "neighbor witness" transactions. If every pair of consecutive nodes (U,V) in the route had earlier logged that they have a direct communication for that RREQ, the route is confirmed consistent. If any link in the route was fabricated (e.g., an attacker injected a fake neighbor), the ledger would lack a corresponding witness from the supposed transmitter or receiver, and the validators would reject the RREP transaction as invalid. This approach thwarts wormhole attacks: in a wormhole, two far apart nodes (say M and N) might appear adjacent in the route without having direct radio contact. In our system, when M forwards to N via a wormhole tunnel, N can log a witness "N heard RREQ from M", but M and N are not real neighbors in the network topology – how would the system detect that? If M and N collude fully, they could both log witness transactions (M says "I forwarded to N", N says "I received from M"). To address this, we might integrate geographic or timing information: e.g., a witness transaction could include a timestamp and perhaps the signal quality of the heard RREQ. If M and N are distant, their transmission time and reception time might not line up with a single-hop radio propagation (especially if the wormhole transmission took longer). Also, other nearby honest nodes of M or N might dispute the neighbor relation (if N is not within radio range of M, normally N's neighbors wouldn't include M; a smart contract could flag if an alleged new neighbor relation pops up that contradicts known distance or coverage data). Admittedly, complete wormhole mitigation may require additional methods (like distance bounding or radio watermarking), but the blockchain provides a framework to store and analyze such evidence. For our theoretical design, we assume either the wormhole can be detected by inconsistent timing or the need for multiple colluders makes it detectable by unusual patterns in the ledger.

Once the route is validated, it is considered an agreed route and can be used for data forwarding. Node S can now send its data packets along the path S->…->D. We require that each data packet is marked with an identifier of the route (e.g., a route ID or the hash of the node sequence) and possibly sequence numbers, and that nodes along the way only forward it if it matches the route they agreed to forward. Because the route was established via a contract, intermediate nodes have essentially committed to forward packets for a session from S to D. This could be enforced by a smart contract that holds a security deposit for each node or a reputation stake; however, in a purely theoretical design, we can simply rely on the auditability: if an intermediate node drops the packet, the destination D won't see it and can trigger an alert (perhaps sending a "packet not received" event to the blockchain after a timeout). The contract then could decrement the reputation of the suspect node or mark that node as a potential blackhole in the ledger.

3.3 Consensus Mechanism for the Blockchain

The choice of blockchain platform (and thus consensus algorithm) is critical to ensure the system remains lightweight. We consider two scenarios aligned with the platforms mentioned: one using IOTA (DAG-based) and one using Hyperledger Fabric (permissioned BFT). Both avoid Proof-of-Work mining in the classical sense.

In an IOTA-based design, every validator (CH or capable IoT node) participates in the Tangle by issuing transactions. When a validator needs to add a new record (say a route witness or route reply), it selects two tip transactions to approve (per IOTA protocol) and does a small Proof-of-Work to attach the new transaction. The small PoW is sized such that even IoT-class processors (with perhaps tens of MHz CPU) can compute it in a fraction of a second, ensuring that adding a transaction doesn't stall the network. As transactions accumulate, the consensus emerges from the weight of cumulative approvals. The final confirmation time in IOTA can be on the order of seconds. One advantage of IOTA is scalability: as IoT devices generate more transactions (like many route updates), the network can theoretically handle more throughput since validation is parallelized. Another advantage is that IOTA has no miners or transaction fees – this is important because IoT nodes cannot handle complex reward mechanisms or micropayments with fees for something as frequent as routing updates. By using IOTA's DAG, every node that issues a transaction contributes a bit of work and helps confirm others, sharing the burden. However, pure IOTA (at least historically) had issues with partition tolerance and required a coordinator for finality; in our theoretical design, we'll assume a coordinator-free IOTA where a lightweight consensus (like tip selection and maybe a voting mechanism for conflict resolution) runs distributed among CHs. If conflicts occur (e.g., two different routes submitted at same time for the same source-dest), a rule is in place – perhaps the one with more cumulative weight wins or the earliest timestamp wins if no conflict in ledger references.

In a Hyperledger Fabric-based design, since the set of validator nodes (CHs) is known and permissioned, we can use an efficient consensus like Raft (Crash Fault Tolerant) or a lightweight Byzantine Fault Tolerance (like Istanbul BFT or Simplified PBFT) given the scale is relatively small (maybe tens of validator nodes). Fabric's approach would involve an ordering service: one or more CHs act as orderers to collect transactions, order them, and package them into blocks, which then get distributed to all CHs. The ordering service can be made crash fault tolerant (which is fine if we assume CHs are mostly honest but we want reliability), or if we suspect up to f of N CHs might be malicious, a BFT ordering service can be used to still guarantee consistency as long as a majority (or supermajority) are honest. The consensus in Fabric ensures immediate finality of blocks – once a block is agreed and committed, it's final (no forks). This is beneficial for a routing scenario because nodes can trust the route information as soon as it's committed, without worrying about chain reorganizations. The latency for consensus in a LAN environment can be around 50–500 ms for Fabric (depending on settings and number of nodes) which is low. We would configure Fabric's block time or block size to be small, to commit routing info quickly. For example, each RREQ or RREP transaction could be committed in the next block within a second. Fabric also allows us to implement the logic as chaincode: e.g., a chaincode that automatically checks those "witness" records and validates route transactions could run as part of transaction validation phase, making consensus decisions partly application-aware (smart contract-enabled consensus).

We emphasize that Proof-of-Work (PoW) is avoided in our protocol (except for IOTA's negligible PoW) due to energy concerns. We also avoid Proof-of-Stake (PoS) with token incentives in this context because IoT networks may not have a concept of cryptocurrency readily integrated, and we want to minimize complexity. However, one could imagine a variant where each node has a stake (like reputation or deposit) that it could lose if it misbehaves; that's more of a penalty mechanism than a consensus mechanism. Recent research has even suggested IoT-specific consensus like Proof-of-Resource (PoR) where nodes prove they have certain resources (like free memory or sensors) to earn the right to publish blocks. Such approaches are intriguing as future directions – they could allow even sensor nodes to partake in consensus by leveraging the very resource constraints (for example, demonstrating you have sufficient battery and radio bandwidth before being allowed to add many transactions, to prevent spam).

In our design's default mode, consensus is maintained by a small subset of moderately powerful nodes. This keeps the blockchain lean (not every sensor does everything) and fits with many real deployments where perhaps each field site or area has a gateway. If fully decentralized (every node a validator) is desired, one could scale down the consensus (e.g., use a simplified consensus where each node takes turns to add blocks in a round-robin – effectively Proof-of-Authority by rotation). The main security consideration is that the consensus algorithm should tolerate some fraction of nodes being malicious, since we are explicitly trying to secure against insider threats (compromised IoT nodes). Therefore, a BFT-style consensus (able to handle f malicious out of n) is preferable to a simple leader election. Many permissioned blockchains (Fabric, Quorum, etc.) already implement such BFT consensus or can plug one in.

3.4 Lightweight Smart Contracts for Routing and Trust

Smart contracts in the BCR-IoT protocol are used to automate verification and enforcement tasks that would otherwise be difficult to manage in a distributed setting. We design these contracts to be lightweight – meaning the logic is straightforward and executes with low computational overhead – so that even constrained environments or limited-execution platforms (like IOTA's second-layer or Fabric's chaincode on small VMs) can run them.

The primary smart contract (or set of chaincode functions) in our system has the following responsibilities:

a) Route Validation Contract

This contract is invoked when a proposed route is submitted (as in the RREP stage). It automatically checks the ledger for the requisite witness transactions from each hop. Pseudocode logic: on Route Proposal(route R, source S, dest D): for each consecutive pair (node i, node_{i+1}) in R, look up a recent "Link Proof" event for those two nodes under the same RREQ ID; if any link proof is missing or if any node in R is not registered/authorized, then reject the transaction; else accept and mark route R as valid until a certain expiration time. By doing this in a contract, we eliminate manual offline checking – the consensus nodes all run this same code and will only commit the route if it passes. This ensures all validators enforce the same security policy for routes. The result is that only legitimate routes get recorded.

b) Reputation/Trust Management Contract
We include a contract that keeps track of a trust score or reputation for each node. This score can be simple (e.g., an integer count of misbehaviors or a rating between 0 and 100). Initially, all nodes might start with a neutral score. The ledger updates this score based on certain events:

- If a node is part of a route that successfully delivered data, perhaps it gains a small trust increment (or conversely, you mainly decrement on bad behavior).
- If a node was supposed to forward a packet but was suspected of dropping it (for example, the destination reports a packet loss or an intermediary watchdog node reports a drop), the contract will decrement that node's trust.
- If a node issues an invalid transaction or is caught in an attempted attack (e.g., it claimed a neighbor that was not actually a neighbor, or it tried to flood the network with RREQs above a threshold), its trust score drops significantly.
- Optionally, periodic "rewards" for good participation (staying online, responding to RREQs, etc.) could boost trust to incentivize reliability.

The trust scores stored on-chain are accessible to the routing protocol: when selecting routes, the protocol (via the CH or via the contract itself) can prefer routes that avoid low-trust nodes. For instance, if two route replies come back, one including a node with very poor trust, the source (or a smart contract) can disregard that route. In extreme cases, the contract can blacklist a node (trust score too low) which essentially means excluding it from routing entirely until possibly re-authenticated or manually reset. This creates a self-policing network: repeated malicious actions get logged and that node is gradually isolated. Traditional routing lacked this memory – once an attack happened, nodes had to independently decide to avoid someone, whereas here the knowledge is global and persistent.

c) Incentive Contract (optional)
If we integrate a token mechanism (which could be a custom token on the ledger), a contract could implement micropayments for forwarding. For example, the source S attaches a tiny reward for delivering a packet to D. On successful delivery (D or intermediate nodes could log an acknowledgment transaction when they forward or receive data), the contract releases the reward to the forwarding nodes. Conversely, if delivery fails and a particular node is identified as dropping, that node might lose a collateral deposit. This idea, similar to the "contractual routing" model and prior works like Sprite (a credit-based system for ad-hoc networks), can directly motivate nodes to participate correctly. However, implementing this requires a currency and handling the complexities of who pays and how to prevent abuse (e.g., an attacker might try to farm rewards by spamming its own routes). For our theoretical design, we consider this an optional extension for scenarios where an incentive layer is appropriate (such as a multi-owner IoT network where devices belong to different stakeholders and need compensation to relay others' traffic).

d) Topology and Performance Contract (monitoring)
Another auxiliary contract can monitor general network health – for example, tracking how many RREQs are going on, how many get replies, average hop count, etc. This is more for analysis and optimization: the contract could flag if certain areas have consistently long routes (maybe suggesting a new CH needed), or if a particular node seems to attract a lot of traffic (perhaps a hint of a central data aggregator that might become a bottleneck or target). This goes beyond classic routing into network management, but since we have the data on-chain, it's feasible to use it.

The lightweight nature of these contracts is seen in that each operation is simple lookups or additions, and often the logic triggers only on certain events (route discovery, packet delivery). They do not require heavy computation like large-scale matrix math or big data processing on-chain. By keeping contracts simple, we minimize the execution cost on the blockchain nodes and reduce latency.

It's worth noting that IOTA's base protocol does not natively support expressive smart contracts on the ledger (it's more a transaction ledger), but the IOTA Smart Contract (ISC) layer or integrating with an off-chain compute can achieve similar outcomes. In a Fabric scenario, chaincode easily handles all the above as it's essentially running on a host machine.

e) Security of contracts
We make sure these contracts themselves are secure: e.g., the route validation contract should only accept properly signed witness logs; the trust update contract should ensure one event triggers one update (to avoid a malicious spammer giving someone bad reputation by faking multiple reports – but since all reports are signed by real nodes, that is controlled). Also, we ensure that a compromised node cannot directly manipulate its trust score – it would have to misbehave and get caught to be lowered, or behave to be raised, as the logic is predetermined. The immutability of the blockchain means the history of trust changes is available for auditing if needed (for example, an owner of a device might want to dispute if their device was unfairly blacklisted – they can see all recorded incidents).

3.5 Route Maintenance and Updating the Ledger

Routes in IoT networks can break due to node mobility, node failure (battery depletion), or environmental changes affecting link quality. Our BCR-IoT protocol handles route maintenance similarly to traditional protocols but adds the blockchain logging for consistency and security.

If a link on a route breaks (e.g., node X cannot reach node Y anymore due to Y moving out of range or shutting down), X will detect a link failure (no link-layer ACK or an error from the MAC). In a normal AODV, X would send a RERR to the source to notify that route is invalid. We do the same but with signed RERR messages. When a RERR is generated, it is also posted to the blockchain: a RouteError transaction stating "Link X->Y on route S-D broken at time t". This promptly invalidates the route in the ledger's view. The source S on seeing the RERR (or on observing on the ledger) knows it must initiate a new route discovery if it still has data for D. The trust management contract may treat frequent link breakage differently from malicious drop – link break is not malicious per se, so it might not penalize X or Y (except if

there is suspicion that one of them deliberately went silent; but that's hard to distinguish from normal failure).

Because the blockchain keeps track of active routes, we can also optimize route discovery frequency: if the same source S and dest D communicate often, S can check the ledger if a valid recent route exists before flooding a new RREQ. Even if S had not cached it, the blockchain might have it. This is effectively leveraging the blockchain as a distributed cache or routing table. All nodes can query it (through their CH) to find if a route to a given destination is known. This "global memory" aspect can reduce overhead by reusing routes network-wide, something not possible in purely local algorithms. However, we must ensure using a cached route is safe – the ledger would have an expiration or a last-seen timestamp, and if that is older than some threshold (or if a RERR was logged later), it should be considered stale. If up-to-date, using it avoids a new flood, saving energy and time.
Scalability considerations: The architecture and protocol described are intended to scale to networks of potentially hundreds or thousands of IoT devices, but not unlimited size. One must consider how the blockchain itself scales: if using cluster heads, as clusters grow, CHs may become a bottleneck. This can be alleviated by increasing the number of CHs (more clusters with fewer devices each), or by using a multi-tier blockchain (e.g., each region has its own ledger and occasionally syncs with others – beyond our current scope but a possible extension). The amount of blockchain traffic is proportional to routing activity; in relatively static networks with occasional data flows, this is small. In worst-case scenarios (very dynamic network with continuous flows), the overhead could be high – optimizations like aggregate transactions, limiting frequency of route updates, or even using probabilistic routing with partial info might come into play.

We assume typical IoT conditions: mostly stable topology with infrequent changes, so the blockchain overhead is amortized. In extremely dynamic cases, one might integrate our design with opportunistic routing and only use blockchain for crucial transactions.

Finally, we highlight an implicit assumption: the communication between IoT devices and CHs, and among CHs, needs to be secure (to not introduce a new vulnerability). We rely on the fact that all messages are signed and that CHs run the consensus – so even if an attacker eavesdrops or injects at the radio level, they cannot forge a signed message or alter the blockchain without keys. Encryption of data packets can be done at the application layer if needed; the routing protocol itself doesn't need to encrypt RREQs/RREPs (they're not secret), but could if privacy is a concern (though then intermediate forwarding becomes tricky if payload is encrypted; likely better to keep routing messages authenticated but plaintext).

With the design laid out, we next compare its expected performance and security properties with the traditional protocols and evaluate how effectively it meets the IoT network requirements.

4. Security and Performance Analysis

In this section, we analyze how the proposed blockchain-based routing protocol (BCR-IoT) fares in terms of security (resilience to attacks), scalability, communication overhead, latency, and energy consumption. We compare these aspects with the baseline protocols (RPL and AODV) to highlight improvements and trade-offs. Since our work is theoretical, the analysis is qualitative and based on logical reasoning supported by findings from related studies. We also use insights from literature where similar approaches were simulated or measured.

4.1 Resilience to Routing Attacks

One of the primary motivations for BCR-IoT is to dramatically improve security relative to RPL and AODV, which are vulnerable to numerous attacks. Here we examine common attack scenarios and how our protocol addresses them:

- **Blackhole/Sinkhole Attacks:** In both RPL and AODV, a blackhole can easily occur because nodes trust advertised routes or rank values without a global verification. In RPL, a node might advertise a very good rank to lure traffic (sinkhole) and then drop it (blackhole); in AODV, a node replies to RREQs with a false short path. In BCR-IoT, any route reply or RPL-like rank advertisement must be validated against the blockchain records. A malicious node cannot unilaterally declare it has the best path – it would need to forge a whole sequence of link proofs on the ledger, which is not possible without collusion from neighbors. If a node tries to drop packets silently after attracting them, the trust contract will catch this due to missing delivery acknowledgments. Over time, its reputation plummets, and it will be excluded from routes. This is a significant improvement: where traditional protocols might continually be fooled by a blackhole until an external mechanism intervenes, BCR-IoT builds detection and exclusion into the routing process (with an immutable audit trail to support it). Even without a trust score, the immediate effect is that the source will see on the ledger that packets didn't reach the destination and which hop was the last confirmed – pinpointing the suspicious node. By contrast, in AODV, the source might only know the packet was lost but not where.
- **Wormhole Attacks:** A wormhole is harder to tackle but our route validation mechanism provides some defense. If two colluding attackers create a wormhole, to successfully use it in our protocol they must fake neighbor relations on the ledger. This requires them to at least produce signed witness transactions for a link that doesn't actually exist in normal radio range. While they could do that (since they are colluding, one can sign for "I heard from the other"), these fake transactions might be detectable by context (e.g., if the network usually doesn't have that link). In addition, because every other hop on the route is validated, a wormhole doesn't let the attackers inject false nodes, it only shortcut the path. If their goal was to snoop data, they'd still have to forward it (since dropping would get them caught as blackhole). If their goal was to disrupt, they could drop via wormhole – but then two nodes lose reputation at once. There is a rich area of research on wormhole detection (like packet leash protocols, etc.),

which could be integrated: for example, the blockchain could store location claims of nodes and detect if an alleged one-hop link is geographically impossible. For our scope, we assume wormholes are at least partially mitigated because the system would notice unusual route patterns and possibly because other nodes will report inconsistencies. At worst, the wormhole provides a way to forward quicker, but it cannot tamper with data without being the dropping point (which is then just a blackhole case). RPL has almost no built-in wormhole defense, and AODV neither; so any capability here is an improvement.

- **Sybil Attack:** This is when one attacker pretends to be multiple identities. In RPL/AODV, a Sybil attacker could inject multiple fake RREQs or appear as multiple nodes to confuse routing. In BCR-IoT, Sybil is mitigated at the identity layer – since each identity must be registered on the blockchain, creating many fake identities would either require compromising many real keys or somehow subverting the registration process. If an attacker pre-loaded a bunch of fake identities, the network could be spammed, but we can implement limits (like the contract could throttle the rate of new identities or require proof-of-work to register). Additionally, if a Sybil node tries to act as many nodes, the trust system will eventually correlate that those identities always appear together or misbehave similarly, raising suspicion. Hyperledger Fabric's permissioned nature essentially nullifies Sybil by not letting arbitrary nodes join. So BCR-IoT is robust against Sybil except possibly during initial bootstrapping if not carefully controlled.
- **Rank and Routing Metric Manipulation:** This is specific to RPL – an attacker changes its rank to something illegitimate. In our design (which is more AODV-like on demand), we don't have ranks; but in any distance-vector context, a node lying about distance is akin to claiming a false route. Because route proposals are verified via actual neighbor sequences, it can't lie about distance without physically being at that position in the route. For instance, if an attacker claims to be only one hop away from the destination (to shorten distance), it would have to produce a witness from the destination as its neighbor – essentially saying "I'm neighbor to D". If that's false, the real neighbors of D (or D itself) won't corroborate that on the ledger. So rank manipulation is stopped by requiring collective validation of any topological claim. This addresses a host of RPL attacks (rank, version, even DIS flooding to some extent because the blockchain can detect excessive solicitations).
- **Denial of Service & Resource Exhaustion:** One concern is that introducing blockchain could open new DoS angles – e.g., an attacker flooding RREQs to overload the ledger with transactions. We mitigate this by requiring PoW in IOTA or by having rate controls in the consensus (a Fabric network can simply ignore excessive requests from one identity or put them in a lower priority). The nature of blockchain also means a single node's misbehavior (like spamming) is globally visible, and countermeasures can be taken (like temporarily not routing for that node, or slashing its deposit if using one). Traditional protocols often fail more silently under resource exhaustion (neighbors drop packets, but the network may not immediately know who's spamming). Here, if an attacker sends 1000 RREQs per second, everyone sees those on the ledger and can react (the contract could implement a rate limit where if >N RREQ from same source in timeframe, ignore or penalize). So BCR-IoT inherently provides a trace and tools to manage DoS.
- **Data Integrity and Confidentiality:** Our focus is routing security, but ensuring that the data packets aren't tampered in transit is also important. BCR-IoT's blockchain can guarantee the integrity of routing decisions, but does not automatically encrypt data (that can be handled by application-layer encryption). However, if an attacker tries to tamper with data, the receiver could notice (via checksum or crypto verification) and then log that the data was corrupted in transit. This might be beyond what routing normally concerns, but in a holistic sense, a smart contract could note that "packets from S to D consistently corrupted when passing through node X" – implying node X might be tampering or faulty. This again would lower trust of X.In sum, BCR-IoT provides strong defenses against attacks that plague RPL and AODV. By having a shared ledger of routing state and events, it transforms many silent or local attacks into detectable global events. A key point is that even if an attack succeeds briefly (e.g., one packet drop), it leaves a forensic evidence on the ledger (like where it likely dropped) and the network can adapt. Traditional protocols often required external intrusion detection or couldn't trace issues beyond local logs. Our approach builds detection and response into the routing fabric.

No security solution is perfect, of course. The main assumption is that the majority of validator nodes remain honest. If an attacker can compromise a majority of CHs (or whatever nodes run consensus), then they could falsify the ledger – this is analogous to an attacker owning most network infrastructure in any system, in which case little can be done. We assume that scenario is highly unlikely due to the distributed trust (e.g., CHs might be run by different organizations or have tamper-resistant hardware). Additionally, by using permissioning, we drastically reduce the attack surface for consensus.

4.2 Routing Overhead and Scalability

Any added security usually comes with overhead. We now consider the communication and computation overhead introduced by BCR-IoT and whether it is acceptable for IoT networks, as well as how the system scales with network size compared to RPL/AODV.

4.2.1 Communication Overhead: In BCR-IoT, extra communications come in two forms: on-chain transactions and possibly slightly larger routing packets.

a) On-chain transactions: These include neighbor witness logs, route proposals, route error reports, etc., which are not present in traditional routing. However, not every single packet triggers a blockchain transaction. We try to aggregate and minimize frequency:
   - During route discovery, a witness transaction might be created per hop. In a worst-case n-hop route, n-1 witness logs + 1 route proposal = n transactions. But we can have CHs batch them into one block or compress multiple hops into one multi-signed transaction. For rough comparison: AODV in

flooding a RREQ sends a lot of RREQ packets (O(n) per node in worst-case flood), whereas we send those plus log them. If n is large (say 100 nodes), the flood overhead dominates anyway. The ledger writing is mostly handled by CHs at a higher layer, possibly over a more reliable connection.

- For each data session, we might incur these route setup transactions once. Data packets themselves are not individually put on chain (unless using micropayment per packet, which we consider optional). So data forwarding overhead is identical to AODV (just normal unicast per hop).
- Periodic trust updates or heartbeat transactions can be tuned (maybe only upon notable events or at low frequency).

The additional bandwidth used by blockchain traffic is a concern on low-power radios. But if CHs are, say, border routers with Ethernet or high-power radios, that's fine. If every node were participating via the same 802.15.4 link, then yes the control overhead might be higher than RPL's trickle of DIOs. One study indicated RPL was most power-efficient among AODV and LOADng in 6LoWPAN. BCR-IoT will not beat RPL in raw efficiency because we intentionally add more information exchange. The question is: is it within a reasonable factor? If RPL sends periodic updates every minute, and our BCR-IoT might send a bit more due to security events, perhaps the overhead is 10-20% more in quiet networks, and maybe 2x in very dynamic networks. Given IoT radios often sit idle or at low duty cycles, many networks can tolerate some overhead increase for the sake of security (especially for critical applications).

b) Routing packet size: We include signatures on RREQ/RREP. A typical ECC signature might be 64 bytes. RPL messages are few bytes normally; adding 64 bytes could be significant. However, on a per-hop basis, 64 bytes extra is usually acceptable in 802.15.4 frames (127-byte max) as long as not too many stacked signatures. We avoided accumulating all signatures in the RREQ because that would blow up size; instead we log to chain. So RREQ carries essentially one signature (of initiator), maybe an HMAC from intermediate if needed for immediate neighbor check, but not a chain of sigs. RREP carries one signature (dest). So the packets are only slightly larger than AODV's. RPL's DIO could be secured by its internal mechanism (which also adds similar overhead per message with MIC or signature if used). So BCR-IoT's on-air packet overhead is comparable to using RPL's built-in security option, but we do much more with it.

4.2.2 Computation Overhead: IoT devices will perform cryptographic operations: verifying signatures of neighbors, maybe doing a PoW (in IOTA's case) or some hashing for transactions. Modern IoT nodes can handle elliptic curve signatures – e.g., using ECDSA or Ed25519, a typical microcontroller might verify in a few milliseconds to tens of ms. This is more cost than doing nothing, but it may be acceptable given that not every packet requires a new verification (neighbors can cache public keys, etc.). The validator nodes handle the brunt of consensus computations. If using Fabric, that might even run on cloud or edge servers, so sensors do almost zero heavy lifting beyond signing their messages. If using IOTA and assuming even sensors do PoW: IOTA's PoW difficulty can be adjusted such that a sensor can do it in maybe 100 ms, which might be acceptable for occasional transactions.

The effect on energy consumption ties both comm and comp. Transmitting extra bytes and computing crypto will consume more energy per operation. There is a risk that on battery-operated nodes, this reduces lifetime. However, consider that many IoT devices (like sensors) wake up, send some data, sleep. The routing overhead only occurs when needed. If the network is mostly static with infrequent communications, the overhead might barely dent the battery. If the network is very active, those devices likely have a power supply or the network is designed to accommodate traffic anyway. Additionally, our hierarchical approach can allow very sleepy nodes – a leaf node could wake up, send data to CH, and CH handles routing with others. The leaf might not even hear all the blockchain chatter, saving its energy.

Interestingly, a study on a trust-based RPL security mechanism showed it achieved significant security gains with only ~2.3% increase in average power consumption. Our approach is more comprehensive than a simple trust model, but if efficiently implemented, the overhead might be on the order of a few percent to, say, 50% more in worst cases. Over provisioning battery by that factor could be worthwhile for the security provided.

4.2.3 Scalability: Scalability can refer to number of nodes and also to network density/traffic. RPL is known to handle large networks (hundreds of nodes) by its trickle algorithm which reduces message frequency as the network stabilizes. AODV can handle reasonably large networks but its flooding can become costly beyond a certain scale or high mobility. BCR-IoT's scalability depends on the blockchain's scalability. If using IOTA, the system can potentially scale to very large networks because there isn't a hard limit – more transactions means more verification but also faster consensus. The partitioning of network into clusters also helps horizontal scalability: you can add more clusters with new CHs joining the blockchain network. A permissioned blockchain like Fabric can scale to perhaps dozens of organizations and hundreds of nodes, but if we imagine thousands of tiny nodes through gateways, that's still fine (only gateways run Fabric, thousands of leaves connect to, say, 10 gateways).

One concern is ledger size growth: if the network runs for years and keeps logging, the blockchain can become huge (MBs to GBs). Storage is cheap on gateways, but not on sensors. However, sensors don't have to store the ledger; only validators do. Validators can also use pruning or summarizing techniques: e.g., check-points after certain intervals (like a snapshot of trust scores and active routes, then prune older transaction history beyond some window). As long as one archival node keeps everything for auditing offline, others could drop old data to save space. This is similar to Ethereum's concept of state vs history or IOTA's periodic snapshotting of the tangle to keep size manageable.

Compared to RPL, which is extremely lightweight in overhead but has poor security, our protocol is heavier but scales in a controlled way. Compared to AODV, our initial

route discovery is similar or slightly more overhead (since AODV floods, we flood plus log events – the flood dominates complexity). AODV's problem at scale is lots of broadcasts if many sources simultaneously need routes. Our approach could alleviate some of that by route caching via ledger: nodes can look up routes rather than always flood, which actually could reduce overhead in a busy network. In an extreme scenario with many flows, AODV floods might congest the network, whereas BCR-IoT might reuse known routes or at least spread out the information via blockchain (which can propagate over possibly out-of-band channels among CHs, not using the same radio as devices). So it's plausible that beyond a certain scale, our approach might show better throughput for large IoT networks, thanks to better coordination of routing info.

4.2.4 Latency: The route establishment latency in BCR-IoT includes the time to perform consensus. For example, if a RREQ is flooded and reaches dest in 50 ms (depending on hops and propagation), and then an RREP is returned in another 50 ms, AODV would consider the route ready at that point (~100 ms). In our case, we would ideally wait until the route is validated on-chain. If using Fabric with sub-second finality, maybe add 500 ms, so route ready in ~600 ms. If using IOTA and waiting for confirmation (which might be a few seconds for safety), route ready in ~2-3 seconds. This is additional delay before data can be confidently sent. We could allow data to start flowing tentatively while validation is underway to optimize for time, especially if we expect it to pass. But to be safe, one might wait. This latency is not an issue for non-real-time applications (a few seconds setup for a multi-hop route in IoT is often fine, e.g., environmental sensing). For real-time or very delay-sensitive flows (like some industrial control), this could be a downside – though such scenarios often have fixed topologies or engineered routes, not discovering on the fly.

Once routes are established, the per-hop latency for data forwarding is not affected (we don't add processing per packet except maybe an ID check which is trivial). In RPL, you might have slightly longer routes sometimes (because RPL might not choose the absolute shortest path if using certain metrics), whereas in on-demand protocols typically you get near-shortest paths. Our protocol chooses routes primarily based on who responded, but the ledger could help choose an optimal route if multiple are proposed (it can store metrics). We might incorporate a slight bias to routes with fewer hops or higher quality, which could reduce end-to-end latency of data relative to a random or trust-only choice.

In terms of consistency, one must consider latency of information propagation: in RPL, if a node becomes malicious, it might take a while for others to notice or for trickle to fix topology, whereas in BCR-IoT, as soon as an incident is recorded on ledger, everyone knows at next block. That is a faster reaction (at most the block time). So while initial setup is slower, adaptation to failures or attacks can be faster globally. This is a different aspect of network performance – resilience speed.

4.3 Comparative Summary (BCR-IoT vs Traditional Protocols)

We consolidate the comparison as follows:

4.3.1 Security: BCR-IoT strongly outperforms RPL and AODV. RPL and AODV in default forms provide little security (aside from optional measures rarely used), being vulnerable to multiple attacks. BCR-IoT offers built-in authentication, audit trails, and trust management. It effectively mitigates blackhole, rank, Sybil attacks, and provides tools to handle wormholes and DoS, which neither RPL nor AODV could handle without external solutions.

4.3.2 Scalability: RPL is known for good scalability (hundreds of nodes) in static scenarios due to its efficient upkeep. AODV scales less gracefully if there are frequent route discoveries, but still used in moderate network sizes (tens of nodes in field tests, possibly 100s with optimizations). BCR-IoT introduces new scaling dimensions: the blockchain network among CHs must scale with number of clusters. Permissioned blockchains can scale to enterprise-level (tens of orgs, 100s of nodes). DAG-based approach can scale much further. For extremely large IoT (thousands of nodes), one might hierarchicalize further or use sharding (e.g., separate blockchains per region that occasionally sync). We foresee BCR-IoT can scale to networks of at least the same order of magnitude as RPL networks (hundreds of devices) and likely more, due to route info sharing. The overhead per device might increase slower than linear because of shared knowledge (10 route discoveries might serve 100 nodes if info reused). In contrast, RPL overhead increases with nodes but slowly (since DIO trickle slows down in bigger network), and AODV overhead increases roughly linearly with number of pairwise communications demands.

4.3.3 Overhead & Efficiency: RPL is the most efficient in terms of control traffic under normal operation, as it sends periodic beacons and occasional adjustments. AODV is reactive so overhead is proportional to traffic demands – idle network means no overhead, active means possibly many RREQ floods. BCR-IoT's overhead is proportional to routing changes and attacks: in a stable network with no malicious activity, overhead could be moderate (just route setups and occasional trust updates). In a network facing many attacks, ironically RPL/AODV might fail (so low overhead but network not working), whereas BCR-IoT will have overhead (as it logs and fights attacks) but keep network service. If we quantify, RPL control overhead is often <5% of network bandwidth in many deployments; AODV might spike to higher during route finds. BCR-IoT might make that maybe 10% in steady state if transactions are small and infrequent relative to data. This is speculative – an actual measurement would require prototyping.

- **Latency:** RPL can have higher *data* latency if topology repairs are slow (one reason some prefer reactive protocols for dynamic networks). AODV can have route discovery latency but after that quick. BCR-IoT adds a setup latency due to consensus but ensures reliable performance after. For many IoT applications that are not ultra-low-latency, this is acceptable. If needed, parameters like block time can be tuned down to sub-second to minimize delay, at the cost of more frequent blocks.
- **Energy:** As indicated in Section 4.2, RPL tends to be extremely frugal on energy (devices mostly sleeping except brief DIO exchanges) – it was designed for that. AODV requires nodes to wake for network-wide floods

occasionally, consuming more energy especially in dense networks (more receptions of RREQ). BCR-IoT requires nodes to do crypto and possibly more listening (e.g., to hear CH announcements). The cluster head approach can actually save energy for leaves because a leaf might just talk to CH and not participate in entire network flood (CH might do RREQ on behalf of cluster, via other CHs). So if we partition well, leaves save energy (like a TDMA schedule to talk to CH). CHs spend more (they might be mains-powered or have stronger battery). Thus, energy usage is not uniform: it is shifted towards more capable nodes. This is a classic trade-off in heterogeneous networks. Overall network lifetime can be prolonged if critical battery-powered nodes are not overburdened. So BCR-IoT can be energy-aware by design (i.e., only a subset pays the price). This is different from RPL where every node including tiny ones must rebroadcast DIO and maintain state, or AODV where any node can be part of route discovery propagation.

Table 1 below qualitatively summarizes the comparison:

Table 1: Qualitative comparison of RPL, AODV, and the proposed BCR-IoT protocol

Aspect	RPL (LLN)	AODV (MANET)	BCR-IoT (Proposed)
Security (attacks)	Low – vulnerable to rank, blackhole, etc. (unless external security added)	Low – vulnerable to blackhole, wormhole, etc.	**High** – built-in authentication, immutable logs; mitigates major attacks (blackhole, Sybil, etc.)
Scalability (nodes)	High for static networks (100s of nodes); struggles if many mobile nodes	Moderate (100s possible but flooding overhead increases)	**Moderate-High** – Scales via cluster heads; ledger handles many nodes, but depends on blockchain network capacity (which can be scaled with infrastructure)
Routing Overhead	Very low in steady state (trickle suppresses messages)	Varies (low when idle, high during frequent route discoveries)	**Medium** – additional overhead for ledger transactions and signatures; overhead rises with attack incidence (for security audit logs)
Latency (route setup)	Low for existing routes, but DODAG repairs can be slow; parent switch may take seconds	Route discovery latency can be medium (flood propagation time)	**Medium** – adds consensus delay (potentially sub-second to a few seconds) for route establishment; after setup, forwarding is fast
Energy Usage	Optimized for minimal energy (nodes mostly sleep)	Depends on traffic; flooding drains more energy in dense net	**Higher for validators**, low for regular nodes – cryptographic operations and more listening incur some energy cost, but mostly offloaded to cluster heads; overall consumption manageable (possible 5–20% increase over RPL in exchange for security)
Trust / Accountability	None – no memory of misbehaviour, relies on external IDS	None – each route discovery is anew, no global memory	**Strong** – persistent trust scores for nodes, misbehaviour recorded and punished; network has long-term memory and node accountability
Privacy (routing info exposure)	Medium – an observer can guess topology from RPL DIOs	Medium – route requests/replies are plaintext	**Medium-Low** – All route info is on blockchain (which could be permissioned to limit access). Fabric channels could restrict who sees what. If public, all interactions are recorded (could be sensitive). Extra encryption layers can be added for payload privacy.

The comparison shows that the blockchain-based approach significantly enhances security and trust at the cost of additional overhead and complexity. In contexts like industrial IoT, smart city infrastructure, or any application where security is paramount (e.g., medical IoT data, military sensor networks), this trade-off is often justified. Attacks that could easily partition or deceive an RPL/AODV network would be thwarted or detected by BCR-IoT, ensuring network availability and data integrity in scenarios where failures could be very costly. On the other hand, extremely constraint-driven applications (like a tiny sensor network meant to last years on battery with minimal traffic) might opt to stick with RPL if the threat model is mild, due to BCR-IoT's higher demands.

4.4 Discussion of Limitations and Mitigations

While BCR-IoT offers many improvements, it's important to recognize its limitations. First, the requirement of a blockchain infrastructure means additional complexity in network setup and maintenance. IoT deployments must manage the blockchain nodes (CHs or equivalent), keep them in sync, and secure their software – essentially adding an overlay network that network administrators need to monitor. If a blockchain node fails or is hacked, it could disrupt the routing for that cluster. This is partly mitigated by having multiple validators and perhaps backup cluster heads that can take over if one fails (consensus can continue as long as a quorum remains).

Second, blockchain consistency vs network partition: if the network splits (e.g., some cluster heads lose connection to others), the blockchain could temporarily partition, and routing information might diverge. This could lead to suboptimal or even insecure routes until the partition heals (like one partition might not know a node was flagged malicious in the other). Mechanisms to merge ledgers or ensure at least local safety in partitions would be needed – possibly outside the scope of basic design (it could be handled by defaulting to more cautious routing if not enough validators are reachable).

Another limitation is dependency on time synchronization to some extent. Not as much as some systems, but for logging

events and detecting anomalies like wormholes, having loosely synchronized clocks helps (to compare timestamps). We might need the CHs to sync clocks via NTP or GPS. IoT devices themselves can be unsynchronized and it won't break the protocol, but anything cross-network often benefits from time sync.

Future research directions we identify include: more efficient consensus tailored to IoT (like Proof-of-Resource as mentioned, or scheduled consensus where each validator leads for a short epoch to reduce chatter), integrating machine learning to dynamically adjust trust thresholds (e.g., an ML algorithm could analyze the blockchain data to predict which nodes might fail soon or become malicious, pre-emptively alerting the network), and real-world testbed evaluations. Particularly, implementing BCR-IoT on a small testbed of IoT nodes (with Raspberry Pi or Arduino-class devices as regular nodes and some single-board computers as CHs) would help measure actual overhead, latency, and energy usage, validating the assumptions made in this analysis. Another promising direction is to consider interoperability with existing protocols: for example, could we design BCR-IoT as an extension to RPL (so that if you have RPL nodes and BCR-IoT nodes, they can coexist)? Perhaps by treating the blockchain as an external audit system while RPL continues handling basic routing – giving an upgrade path for legacy systems.

5. Strengths, Limitations, and Future Work

5.1 Strengths of the proposed approach

The BCR-IoT protocol offers a paradigm shift in securing IoT networks. By using blockchain as a backbone for routing, we achieve a level of decentralized trust and transparency unprecedented in traditional network protocols. Every routing decision and action is verifiable and accountable. This dramatically reduces the impact of insider attacks – a compromised node can no longer lie about its status or actions without the rest of the network knowing. In essence, we have introduced a form of collective security enforcement: the network watches itself. This is a powerful concept for IoT, where physical security of nodes is weak; even if some nodes are taken over by an adversary, they cannot easily bring down the network or spoof others because the blockchain (maintained by the healthy nodes) acts as a gatekeeper. Another strength is longevity of trust information: decisions are not just made on the fly from scratch (as in AODV each time); instead, the network "learns" about nodes' behavior over time and can optimize routing accordingly (avoiding troublemakers, preferring reliable nodes). This could lead to more stable network performance in long-lived IoT deployments. Additionally, the approach is flexible: by changing smart contract logic, one can tweak the routing policy or trust model without altering the core firmware of all devices. For example, if a new type of attack emerges, a software update to the contracts or validator logic could implement detection for that attack's pattern on the ledger. This updatability is valuable given IoT devices often have infrequent firmware updates once deployed.

Another strength is that the architecture naturally accommodates multi-owner environments. Consider a smart city where sensors from different departments (traffic, weather, energy) share a network. Traditional routing might struggle with trust if devices don't trust each other cross-department. But with a blockchain, each department can run a validator node and collectively maintain the routing ledger, ensuring no single party can bias the routing for everyone yet everyone can trust the outcome. This is aligned with blockchain's general strength of enabling trust among mutually distrusting parties.

5.2 Limitations and challenges

The benefits do come at the cost of complexity and overhead, as thoroughly discussed. One limitation is reliance on somewhat more powerful nodes (CHs). If an IoT scenario is extremely homogeneous and ultra-constrained (say a network of simple contact sensors with nothing that can act as a CH), deploying a blockchain may be impractical. We've essentially assumed a heterogeneous network or an external infrastructure. This assumption holds in many IoT contexts (smartphones, gateways, or edge servers are around), but not in all (a pure ad-hoc network in a remote area with only identical tiny sensors might not have a "big" node). A potential mitigation is the development of ultra-light blockchain clients that even small microcontrollers can run – for example, use of minimalistic consensus like witness-coordinated consensus where nodes take turns (like a token-passing ledger). Research prototypes of lightweight blockchain for IoT (with minimal code footprint) are underway in the community and could help address this.

Another challenge is data privacy: Blockchain's strength is transparency, but that could conflict with privacy requirements. Routing information might reveal node locations or communication patterns. If an attacker gains read access to the ledger (which in a permissionless chain is open), they could glean sensitive info (like which sensors talk frequently – possibly indicating their roles). We mentioned that Fabric's channels can restrict who sees what (so maybe only the organization validators see full details, while outsiders cannot). In IOTA (public), data is public, but one could encrypt certain fields in transactions. Our design could incorporate encryption for route details, such that only authorized nodes can decrypt the actual path in a transaction (maybe using a group key). However, adding encryption complicates validation (the validators need to check route consistency; they could do so on encrypted data only if using more advanced cryptographic proofs). This is a trade-off: our current design favors security and trust over confidentiality of the meta-data. In future work, exploring privacy-preserving blockchain routing (using techniques like zero-knowledge proofs to prove a route is valid without revealing it fully) would be very interesting.

5.3 Future Research Directions

Building on this work, several avenues emerge:
a) Prototype Implementation and Experimentation: Implement BCR-IoT in a simulator (e.g., Cooja/Contiki for IoT combined with a blockchain simulator, or using Hyperledger Fabric SDK with a network emulator) to measure performance under various scenarios. This would validate assumptions about overhead and latency, and allow fine-

tuning. Metrics like packet delivery ratio under attack, control overhead, and energy usage can be quantified.

b) Optimizing Consensus for IoT: Investigate custom consensus algorithms tailored to the typical scale and trust assumptions of IoT networks. For instance, a hybrid consensus where on small scales a fast BFT is used, and if the network grows or becomes more open, it gracefully switches to a more decentralized scheme. Or a consensus that exploits the sensor network's physical properties – e.g., using the fact that network diameter is small, one could design a fast flood-based agreement (not unlike RPL's root announcements) combined with cryptographic commitments. The Proof-of-Resource (PoR) concept is one such idea: nodes prove they have done certain sensor tasks (like providing data or uptime) to earn the right to validate blocks, thus tying consensus to useful work.

c) Integration with Software-Defined Networking (SDN): There is an emerging idea of using blockchain with SDN controllers to secure networks. One could imagine an SDN-like approach to IoT where controllers use blockchain to share network state and collectively control the network. In fact, if an SDN controller is compromised, a blockchain of controllers could detect conflicting instructions. For routing, our approach is distributed, but some IoT deployments have central control – merging those paradigms could yield a robust hybrid.

d) Cross-domain Routing and Blockchain Interoperability: Future IoT networks might interconnect different blockchain systems. Perhaps one network uses IOTA and a neighboring network uses Fabric; how would routing work across them? Research into blockchain interoperability (via relays or atomic swaps of info) could enable routes that span multiple administrative domains, each securing their part but handing off information through a gateway that is on both ledgers.

e) Energy Harvesting and Economic Models: If we introduce incentive mechanisms (like micropayments for forwarding), the economics of the network become important. Future work could simulate what reward level is needed to encourage participation, or how to prevent abuse of rewards. Also, one could tie it with energy harvesting – e.g., a node that has surplus solar energy could advertise willingness to take more routing load for some reward, while battery-powered nodes might avoid it unless necessary. A blockchain could dynamically broker such deals.

f) AI for Attack Prediction: With the wealth of data on the ledger, using machine learning to analyze patterns (perhaps off-line or by the validators) could predict attacks or node failures before they happen. For example, if a node's behavior gradually changes (longer delays, slight increase in dropping), an ML model might flag it as likely to fail or be compromised soon, prompting preventive measures (like rerouting traffic preemptively). This predictive security could greatly enhance network resilience.

g) Real-world applicability: Ultimately, any solution must justify its complexity by the security need. We see critical applications like smart grids, healthcare monitoring, or defense sensor networks as prime candidates where the data and availability are so important that adding blockchain is warranted. The progression of IoT in industry suggests that as deployments scale and become part of national infrastructure, security frameworks like BCR-IoT will draw interest. In fact, standards bodies (IETF, IEEE) have started discussing blockchain in network management and trust (e.g., IETF's blockchain for trust management drafts). Our work can inform these discussions by showing a concrete design and its trade-offs.

6. Conclusion

In this paper, we presented a comprehensive design for a Blockchain-Based Secure Routing Protocol tailored to IoT networks, addressing the pressing need for enhanced security in resource-constrained environments. By integrating blockchain technology – whether through IOTA's DAG or Hyperledger Fabric's permissioned ledger – with routing processes, we introduced a decentralized trust layer that mitigates many vulnerabilities of traditional protocols like RPL and AODV. Our proposed protocol leverages the immutable and consensus-driven nature of blockchain to act as a shared source of truth for routing decisions, thereby preventing malicious nodes from misrepresenting network information and enabling the detection and isolation of attacks such as blackholes, wormholes, and Sybil incursions.

We detailed the architecture, highlighting how resource-constrained IoT devices can participate with minimal overhead by offloading heavy tasks to more capable validators (cluster heads or gateways), and how smart contracts can automate route validation and trust management. The comparative analysis underscored significant improvements in security and reliability – for example, routes are established with collective verification and data integrity is ensured end-to-end – at the cost of some additional latency and overhead. The evaluation discussion acknowledged that while our approach is more complex than conventional routing, the evolution of IoT is reaching a point where such complexity is justifiable. IoT networks are forming the backbone of critical services (smart cities, industrial automation, etc.), and attacks on these can have real physical consequences. Thus, the strongest point of our work is demonstrating that it is feasible to embed strong security and trust mechanisms at the routing layer without rendering the system unusable for low-power devices. We showed pathways to implement consensus in a lightweight manner and to contain overhead mostly to infrastructure nodes, thus fitting within IoT constraints.

Our work opens several avenues for future exploration – from optimizing the performance of the blockchain layer to deploying pilot networks to validate our assumptions. We also encourage work on standardizing such approaches, possibly as an extension to existing routing protocols (imagine an RFC for "Secure Routing for LLNs using Distributed Ledger").
In conclusion, the blockchain-based secure routing protocol offers a promising solution for secure, scalable, and robust IoT networking. It combines the best of both worlds: the proven efficiency of protocols like RPL/AODV for routing, and the trust guarantees of blockchain for security. As IoT networks continue to expand in scale and importance, such interdisciplinary approaches will be key to ensuring they

remain resilient against the evolving threat landscape while still operating within the practical limits of small devices. The work presented here lays a strong theoretical foundation for that vision, bringing us a step closer to IoT networks that are not only smart and connected but also secure and self-healing by design.

References

[1] Muzammal, S. M., et al. “A Trust-Based Model for Secure Routing against RPL Attacks in Internet of Things.” *Sensors*, vol. 22, no. 18, 2022, 7052.
[2] Ran, C., et al. "An improved AODV routing security algorithm based on blockchain technology in ad hoc network." *EURASIP J. Wireless Comm. and Networking*, 2021.
[3] Wijesekara, P. A. D. S. N., et al. “A Survey on Blockchain-Based Routing in Communication Networks.” *IJEEI*, vol. 11, no. 3, 2023.
[4] Ramezan, G., and C. Leung. “A blockchain-based contractual routing protocol for the Internet of Things using smart contracts.” *Wireless Comm. and Mobile Computing*, 2018, Article ID 4029591.
[5] Chen, X., et al. "Blockchain-based RPL optimization for secure and reliable IoT routing." *IEEE Internet of Things Journal*, 2023 (referenced in).
[6] Chia, P. S., et al. "Routing Protocols Performance on 6LoWPAN IoT Networks." *IoT*, vol. 6, no. 1, 2022, p.12.
[7] Abuagoub, A. M. A. "Security Concerns with IoT Routing: A Review of Attacks, Countermeasures, and Future Prospects." *Advances in Internet of Things*, vol. 14, 2024.
[8] Hyperledger Fabric – IBM Blockchain. “What is Hyperledger Fabric?” IBM, [Online].

Author Profile

Thejiya V is an Assistant Professor in the Department of Computer Application, Krupanidhi Group of Institutions, Bangalore. She completed her Bachelors in Computer Science & Engineering and her Masters in Computer Science & Engineering from Amrita School of Engineering, India under Amrita University. Her current research interests lie in the areas of Computer Networks and Artificial Intelligence.