(2024), **1**:1 1–35

RESEARCH

Trustful IoT Localization: Architecture and Application with Permissioned Blockchain

Zocca Guglielmo[1]

[1]Computer Science, University of Trento, Trento, 38123, Italy, Europe.
[1]Computer Science, Insa Lyon, Lyon, 69100, France, Europe.

Keywords: Privacy, Localisation, Blockchain, Iot

Abstract
The advent of blockchain technology has revolutionized data security and trust mechanisms. One area that particularly benefits from enhanced security measures is the Internet of Things (IoT). Numerous solutions have been proposed to integrate blockchain with IoT, yet many of these fail to address the privacy needs specific to certain applications. In this paper, we propose a blockchain-based solution that ensures both the security and privacy of IoT data. Our approach leverages the features of a permissioned blockchain, specifically Hyperledger Fabric, utilizing its collection feature for privacy. We test our solution using a network of devices known as CLOVES, which demonstrate strong performance characteristics. Additionally, we validate our approach with a specific use case: indoor localization.

Contents

1. Introduction

IoT technology is increasingly becoming an integral part of our daily lives and business operations. It is embedded in home appliances as network chips, found in security cameras, and widely utilized in the logistics field to track transported products. In localization use cases, IoT devices are used to determine the position of individuals indoors for various analytical purposes. For instance, in a museum, visitors

can be given a device that communicates with anchor devices to track their path throughout the exhibits, providing valuable insights into visitor behavior and preferences.

Despite the incredible range of applications, IoT devices often rely on a central server, which may not provide robust security. A central server can be tampered with, often without detection. Blockchain technology can enhance the architecture by providing integrity, auditability, and authenticity. However, public blockchains allow all data to be accessible to everyone, which is not suitable for applications requiring privacy and access control. For instance, in certain localization applications, it is necessary to ensure that sensitive data, such as the position of a target, is only accessible to specific clients and not to all users and peers connected to the blockchain.

In this project, blockchain technology is employed to store data from IoT devices. Specifically, we utilize Hyperledger Fabric, a private and permissioned blockchain. Hyperledger Fabric provides the necessary performance and security for our system, which handles data transmissions from numerous devices within nanoseconds. It also ensures privacy through its collection feature, allowing for data segmentation and controlled access. This means that data from a specific experiment can be restricted to participants only, ensuring confidentiality.

Moreover, Hyperledger Fabric allows for the development of smart contracts (chaincode) using general-purpose programming languages, enhancing the flexibility and ease of developing business logic. These smart contracts run within the blockchain, ensuring trust and integrity in their execution. This combination of features makes Hyperledger Fabric an ideal choice for securely managing and processing IoT data while maintaining privacy and performance.

For this research, due to limited funding and resources, we conducted performance tests using the CLOVES (Communication and Localization Testbed for Validation of Embedded Systems) testbed at the University of Trento. This testbed is accessible to anyone who subscribes to the service, providing a practical and cost-effective solution for evaluating business logic in a network of devices.

All of this technology is tested and evaluated using general or ad-hoc testing frameworks. For instance, Hyperledger Caliper is employed to assess the performance of chaincode execution.

Ultimately, every component of this project contributes to the creation of a prototype—a foundational step towards developing a secure architecture for processing position data from devices in a secure and private manner. The use case involves a client who can register with the system and conduct localization experiments alongside other clients. The client must build his/her network of devices and application business logic upon the security, functionality, and privacy provided by the blockchain. The code for the prototype system, along with all necessary tools for testing, can be found here.

At the end, the report is divided in: 1. Presentation of related works; 2. presentation of the base paper; 3. Explanation of our goal; 4. Explanation of the building blocks of our system; 5. Explanations of the different aspects we have introduce of our system; 6. Explanation in more detail of the security and localization algorithms utilized; 7. Example of deployment and use of our system. 8. Explanation of the tests executed; 9. Explanation and discussion of the tests results; 10. Conclusion of the report.

2. Related work

In recent years, numerous papers have been published discussing the integration of IoT and blockchain technologies. In many instances, blockchain has been implemented alongside machine learning techniques. Additionally, various studies have focused on privacy within blockchain systems, with many of these addressing solutions involving Hyperledger Fabric.

"Blockchain-based Edge Computing Resource Allocation in IoT: A Deep Reinforcement Learning Approach" [2] introduces the Asynchronous Advantage Actor-Critic (A3C) approach to allocate edge computing resources, demonstrating the integration of artificial intelligence (AI) with blockchains. The paper employs a deep reinforcement learning algorithm to handle Edge Computing Node (ECN) selection.

"BlockDeepNet: A Blockchain-Based Secure Deep Learning for IoT Networks" [3] presents a novel approach to deep learning within IoT networks. It introduces Blockchain-based secure Deep Learning

(DL) for IoT networks, facilitating secure and collaborative DL at the device level while ensuring data integrity and confidentiality. Furthermore, it introduces a collaborative DL mechanism within a blockchain environment, enabling local learning models on individual devices and aggregating these models at the edge server through blockchain transactions. The proposed framework, named PIRATE, provides Byzantine resilience for distributed learning in the 5G era by leveraging a secure computing framework based on blockchain sharding techniques.

"Blockchained On-Device Federated Learning" [4] introduces a Blockchained Federated Learning (BlockFL) architecture, enabling on-device machine learning without centralized training data or coordination. This architecture leverages blockchain's consensus mechanism to exchange and verify local learning model updates. BlockFL comprises devices and miners, where miners can be randomly selected devices or separate nodes like network edges (e.g., base stations) with fewer energy constraints. The BlockFL process involves devices computing and uploading local model updates to associated miners, miners exchanging and verifying updates, performing Proof-of-Work (PoW), generating blocks with verified updates, which are then added to the blockchain. Devices download these blocks, compute global model updates, and continue the learning process.

"A Hierarchical Blockchain-Enabled Federated Learning Algorithm for Knowledge Sharing in Internet of Vehicles" [5] introduces a hierarchical blockchain framework tailored for large-scale vehicular networks. The algorithm addresses the distributed nature and privacy requirements of IoVs by employing a hierarchical federated learning approach. Knowledge sharing is depicted as a trading market process, stimulating sharing behaviors, and formulated as a multileader and multiplayer game. Simulation results demonstrate enhanced sharing efficiency and learning quality. Additionally, the blockchain-enabled framework effectively handles certain malicious attacks. The paper proposes a lightweight blockchain featuring a novel consensus mechanism called Proof-of-Learning (PoL). This mechanism merges federated learning with consensus, utilizing learning quality as proof of work to prevent computing power wastage. Training model updates are modeled as a trading market, with training nodes selling their models to servers based on bidding prices, and servers adjusting bids according to the quality of received models.

"A framework for privacy-preservation of IoT healthcare data using Federated Learning and blockchain technology" [6] presents an architecture leveraging Blockchain and Federated Learning to enhance privacy preservation. The framework enhances scalability through the integration of blockchain and Federated Learning, demonstrated in a case study and performance analysis focusing on smart healthcare. It ensures privacy and security using techniques such as differential privacy and homomorphic encryption.

"Integration of Internet of Things and blockchain toward portability and low-energy consumption" [7] introduces an IoT-friendly Blockchain scheme designed to assess the viability of medical supply and drug transportation, aiming to enhance security and address privacy concerns. The system ensures fault tolerance, transparency, and traceability. It employs proof of authentication (PoAh) as a lightweight consensus algorithm suitable for IoT environments. In PoAh, miners, termed trusted devices, authenticate block sources using unique identifiers stored locally. The process involves compiling participant-generated data into a block, with each device appending its Unique Block Token (UBT). Blocks are disseminated across the network, and trusted nodes verify the source identifier against their local list. Validated blocks are added to the local chain and propagated further. Eventually, the entire network adopts the verified blocks.

"Towards a Scalable and Trustworthy Blockchain: IoT Use Case" [8] introduces a scalable and trustworthy blockchain (STB) architecture tailored for the IoT. The proposed framework leverages blockchain sharding and oracles to foster trust among unreliable IoT devices in a fully distributed and trustworthy manner. Notably, it introduces a Peer-To-Peer oracle network designed to guarantee data reliability, scalability, flexibility, and trustworthiness. Additionally, the paper presents a novel lightweight consensus algorithm engineered to significantly scale the blockchain while ensuring interoperability among its participants.

"A Blockchain-based Architecture for Secure and Trustworthy Operations in the Industrial Internet of Things" [9] recognizes the immense potential of merging blockchains with the IoT, particularly in smart industrial settings. This paper introduces a lightweight, scalable, and decentralized private blockchain-based IIoT network. The proposed architecture facilitates various industrial operations such as user and device registration, sensor and actuator data storage, and client service tasks, all while ensuring trustworthiness. Notably, the blockchain mechanism is constructed on a private network accessible exclusively to authorized users.

"PUF-derived IoT identities in a zero-knowledge protocol for blockchain" [10] presents an innovative authentication method where a microcontroller unit (MCU) internally generates a secret key, leveraging manufacturing variability as a physical unclonable function (PUF). By allowing the device to generate its own key, manufacturers avoid the need for a secure environment for external key generation, reducing costs. During production, after loading firmware onto chips, only an internal characterization is required, with the resulting public key, mask, and helper data stored for authentication and recovery purposes.

"Trustworthy IoT: An Evidence Collection Approach based on Smart Contracts" [11] introduces a service-oriented methodology utilizing blockchain and smart contracts for reliable evidence collection, forming the foundation for trustworthy IoT assurance evaluation. This approach balances trustworthiness and performance, leveraging Hyperledger Fabric blockchain for experimental validation. Unlike existing solutions, this methodology correlates evidence with its collection and aggregation method or the decisions derived from it. It also introduces privacy techniques, such as trustworthy data aggregation, which enhance privacy by concealing individual data points while retaining traceability to their source. The function data aggregation DA: Dp × op $\rightarrow$ A processes a set of data points Dp=dp1 , dp2 , . . . , dpn using an aggregation operator op$\in$sum, average, min, max, count, producing aggregated evidence A=op(Dp) stored in the blockchain.

"PrivChain: Provenance and Privacy Preservation in Blockchain-enabled Supply Chains" [12] introduces PrivChain, a blockchain solution employing Zero-Knowledge Succinct Non-Interactive Argument of Knowledge (ZK-SNARKs), an efficient non-interactive variant of zero-knowledge proofs (ZKPs). In PrivChain, participants do not share raw data; instead, they share proofs verifying the validity of their product-related data. Blockchain smart contracts automate the verification process of these proofs. Additionally, the paper proposes a monetary incentive mechanism to reward participants for producing successfully verified proofs.

"Using Blockchain to Achieve Decentralized Privacy in IoT Healthcare" [13] proposes a novel approach to address privacy concerns in IoT healthcare by storing pointers to encrypted data (represented as hashes) on the blockchain, thereby reducing the storage burden on the blockchain itself. The encrypted data is stored off-chain, leveraging The InterPlanetary File System (IPFS) for off-chain storage implementation. To ensure data security, the paper employs a symmetric key encryption scheme. Moreover, it introduces Fine-grained Access Control, empowering patients to grant specific permissions to medical staff for accessing their healthcare data. Patients retain the ability to modify or revoke access permissions, which are securely stored on the blockchain ledger as access-control policies, exclusively modifiable by the patient. This setup guarantees data transparency and auditability, allowing patients to monitor how medical staff access their healthcare data. Digital signatures, implemented using ECDSA, are appended to the data for authentication purposes. Additionally, patients generate secret keys for encrypting data using the AES symmetric encryption algorithm. Access permissions are expressed as policies, specifying the permissions granted by the patient to selected medical staff members, thereby governing access to specific or all patient data.

"Research on Distributed Blockchain-Based Privacy-Preserving and Data Security Framework in IoT" [14] introduces a framework leveraging gateway devices to store and establish a peer-to-peer (P2P) network for facilitating information exchange. Concurrently, a middleware server is deployed to handle data processing and enforce access control management. Through blockchain integration, the framework ensures robust security and privacy safeguards for IoT applications. All data undergo encryption and are stored within gateway nodes, accessible only to authorized entities following verification. To address the heterogeneous nature of IoT data from multiple sources, the framework employs hash algorithms

for data processing, including integration, cleaning, and conversion, thereby enabling seamless device interconnectivity. Hyperledger Fabric is utilized to bolster data privacy, establishing dedicated channels and encrypting data stored on the blockchain, thus enhancing overall security measures.

"A Privacy-Preserving Internet of Things Device Management Scheme Based on Blockchain" [15] integrates attribute-based encryption (ABE) and time-bound key management techniques to establish a privacy-preserving IoT device management system leveraging blockchain. ABE, a public-key encryption algorithm, associates user keys or ciphertexts with specific attributes. In this cryptosystem, decryption of ciphertexts is only possible if the attributes linked to the user key meet those associated with the ciphertext.

"Differentially Private Enhanced Permissioned Blockchain for Private Data Sharing in Industrial IoT" [16] leverages differential privacy to ensure data confidentiality. This approach ensures that the addition or removal of a single data record from a dataset does not significantly alter the output of an algorithm applied to the dataset, subject to the constraint of differential privacy. Differential privacy is achieved by introducing calibrated noise to the actual data, with the amount of noise controlled by the parameter ϵ, known as the differential privacy budget. The adversary model employed is robust, where the adversary possesses all information except the target person or data record. However, multiple queries on the same dataset can degrade privacy protection due to the accumulation of the privacy budget, resulting in a high value of ϵ. Additionally, adversaries can submit repeated queries to the same dataset, averaging the results to predict the actual data. Furthermore, the data holder's ability to respond to queries is limited by ϵ. The integration of differential privacy into the chaincode (smart contract) of Hyperledger Fabric involves accessing the ledger's data and then adding calibrated noise to the actual data before sharing it with the requester, ensuring privacy preservation through ϵ-differential privacy.

"Trustworthy Digital Twins in the Industrial Internet of Things with Blockchain" [17] presents a promising solution for creating digital replicas, or virtual representations, of physical entities such as products, processes, or services. These digital twins undergo comprehensive analysis, prediction, and optimization before real-world implementation. Operating within a closed-loop system, simulation data from digital twins inform and refine operations in the physical system, thereby enhancing overall performance. This bi-directional mapping between the physical and virtual realms is referred to as a Digital Twin (DT).

"The Use of Blockchain to Support Distributed AI Implementation in IoT Systems" [18] offers several contributions: a novel, secure blockchain architecture designed to facilitate Distributed Artificial Intelligence (DAI) on low-power, cost-effective IoT devices; Practical implementation of DAI using a scalable, distributed IoT hardware platform; Prediction and measurement of DAI performance utilizing blockchain in IoT devices, including accuracy, energy consumption, and overall system latency, leveraging data from trusted and robust platforms; A blockchain protocol featuring a new consensus mechanism, transaction formats, and block structures to assist nodes in managing DAI-based transactions and prediction requests.

"AI-enabled Blockchain: An Outlier-aware Consensus Protocol for Blockchain-based IoT Networks" [19] introduces the AI-enabled blockchain (AIBC) network featuring a two-step consensus protocol, utilizing an outlier detection algorithm in the first step and PBFT in the second. This paper proposes an architecture that harnesses the capabilities of edge computing, AI, and blockchain technologies to deliver robust, secure, and intelligent solutions for efficient data processing and sharing. To enhance the security of this architecture, a novel consensus mechanism based on Honesty-Based Distributed Proof of Work (DPOW) is introduced. This mechanism is well-suited for implementation in public blockchain-IoT applications and leverages the collective hash power of IoT devices to distribute the mining workload. Additionally, the presence of edge nodes facilitates the accommodation of the computational power and storage requirements of the blockchain.

"A blockchain-based trust management method for Internet of Things" [20] introduces a blockchain-based trust management mechanism (BBTM) designed to evaluate the trustworthiness of sensor nodes through mobile edge nodes. BBTM facilitates the creation of smart contracts for trust computation and ensures the verification of the computation process.

Many of these papers introduce the concept of computing the trustworthiness of devices on edge nodes. For example, in my project, I employ the same strategy as outlined in [20] using Hyperledger Fabric. It is notable that [11], [12], [14], and [19] also utilize Hyperledger Fabric. In most cases, Hyperledger Fabric is used for its permissioned blockchain capabilities but does not leverage its private data features. In [14], different channels within the Fabric blockchain are used to privatize data. In my report, I present and utilize the concept of collections in Hyperledger Fabric. This approach enhances flexibility and reduces memory and cost requirements. Unlike deploying a new channel, which creates a new ledger, collections allow for logical separation of data within the same ledger. This makes the architecture more efficient and cost-effective.

3. A Trust Architecture for Blockchain in Iot

This report introduces an IoT and blockchain architecture designed to determine the position of a target within a secure environment while ensuring that device and position data remain confidential to unauthorized users. The foundation of this research is the paper titled "A Trust Architecture for Blockchain in IoT" [1], which outlines a layered architecture aimed at enhancing the end-to-end trust of sensor data.

3.1. Architecture

The core components of this structure include devices, gateways, and blockchain technology. Devices serve as the means to collect data from the environment and transmit it to designated gateways. Gateways are responsible for assessing the trustworthiness of devices and subsequently integrating the data into the blockchain. Furthermore, the reputation of gateways is evaluated to ensure reliability within the system. The architecture operates by assessing the reliability of sensor observations at the data layer and adjusting block verification at the blockchain layer. This is facilitated through the implementation of the trust layer. The efficacy of the system is then tested within a localization scenario. The proposed trust layer comprises two pivotal modules: the data trust module and the gateway reputation module. These modules play a crucial role in ensuring the integrity and security of the data within the architecture.

3.2. Data trust module

The data trust module plays a critical role in assessing the trustworthiness of observations from devices. This trust is determined by three key elements: the confidence of the data source (conf), evidence from other observations (sens), and the reputation of the data source (rep). The confidence of a device's observation reflects the level of trust assigned by the device itself. In the context of the paper, this could be exemplified by the signal power received from a target by a device. The evidence from other observations refers to the level of support provided by neighboring devices for a given observation. For instance, in the scenario outlined in the paper, this value might represent the similarity between signals power calculated by other devices and those calculated by a specific device. The reputation of the data source is a dynamic value that evolves over the system's lifespan, influenced by the behavior of the device. Malicious behavior leads to a decrease in reputation, while positive behavior results in an increase. The trustworthiness of a device is computed using the formula: Trust = conf * sens * rep. This calculation integrates the confidence level, evidence from other observations, and reputation to determine the overall trustworthiness of the device's observation.

3.3. Gateway reputation module

The gateway reputation module is responsible for monitoring the long-term behavior of gateway nodes and adjusting block validation procedures accordingly based on the reputation of each gateway node. This reputation system entails regular updates from the blockchain layer, where the honesty of each

node in block mining $B(Gi)$ is recorded using both direct and indirect evidence. These reports are then used to update the reputation score of each node. The reputation module is intricately linked with the data trust mechanism, enhancing the block validation process by verifying both the trust values assigned to observations by the gateway and the device transactions reported in the block. Additionally, external sources of a node's reputation $Ext(Gi)$ can be incorporated into the reputation score. In summary, the reputation score, $Rep(Gi) \in [Repmin, Repmax]$, of node Gi is based on a function $g : Rep(Gi) = g[T(Gi), B(Gi), Ext(Gi)]$ where $T(Gi)$ captures how much other validator nodes trust Gi based on Gi 's trust value assignment to the observations. The reputation of a gateway node increases if the generated block is successfully validated and decreases otherwise, providing a dynamic mechanism for maintaining the integrity of the blockchain network.

3.4. Generation and Validation of block

Given that gateway nodes are recognized within the network and possess the necessary permissions to generate blocks, they don't engage in computationally intensive block mining competitions. Instead, this paper advocates for a lightweight block generation approach, where gateways produce blocks at predetermined intervals. Additionally, an adaptive block validation mechanism is proposed. Depending on the reputation of the block-generating node, each validator randomly verifies a portion of the transactions within the block.

3.5. Advantages

In summary, this system offers several advantages: a streamlined and rapid block generation process, the utilization of blockchain for ensuring data integrity, an adaptive block validation technique, and a robust trust system for both devices and gateways. These latter components enable the system to effectively address potential threats from malicious devices or gateways in a straightforward and efficient manner.

3.6. Lack of privacy

In summary, the weaknesses of this system include the use of a private blockchain that does not adequately anonymize the identities of gateways, a lack of robust security measures to protect data transmission between devices, targets, and gateways, and a scalability challenge due to the public and visible nature of all data within the blockchain. From the last assertion derives that there is no access control system to partition and privatize data among multiple users.

3.7. What we take

In our research we utilized the same scheme of the architecture, so divided in gateways and devices, with same interaction. Moreover,We adopted the concept of utilizing a private blockchain for enhanced performance and implemented a trust system for devices to address potential threats posed by malicious entities.

4. Our goal

The aim of our research is to enhance a blockchain structure by incorporating data privatization features, drawing inspiration from the paper "A Trust Architecture for Blockchain in IoT" [1]. We aim to demonstrate these novel characteristics through an example application—a prototype illustrating an end-to-end process from devices to users for calculating the position of a target.

Our goal is to achieve data privatization within the architecture, ensuring that experiment application data stored in the blockchain are accessible only to those responsible for the experiment or authorized individuals. We strive to achieve this property without compromising the inherent advantages of blockchain technology, including auditability, integrity, and authenticity.

5. Building blocks

5.1. Hyperleadger Fabric

Hyperledger Fabric is an open source enterprise-grade permissioned distributed ledger technology (DLT) platform, designed for use in enterprise contexts, that delivers some key differentiating capabilities over other popular distributed ledger or blockchain platforms. Fabric has a highly modular and configurable architecture, enabling innovation, versatility and optimization for a broad range of industry use cases including banking, finance, insurance, healthcare, human resources, supply chain and even digital music delivery.

5.1.1. Main Components

At a high level, Fabric is comprised of the following modular components:

- Peers node that maintain the ledger and propose transaction.
- An ordering service nodes that establishes consensus on the order of transactions and then broadcasts blocks to peers. Ordering service is logically decoupled from the peers that execute transactions and maintain the ledger.
- A membership service provider (MSP) that is responsible for associating entities in the network with cryptographic identities. For example can associate an entity to a user client or a admin client.
- A peer-to-peer gossip service that disseminates the blocks output by ordering service to other peers.
- Smart contracts ("chaincode") that runs within a container environment (e.g. Docker) for isolation.
- Endorsement and validation policy enforcement that can be independently configured per application.
- A service (as CA) for the creation of cryptographic material for the various entity in the network, utilized for the creation of the digital certificate and private key. They are used by entities in the network to identify and communicate with other entities.

5.1.2. Smart Contract

Fabric is the first distributed ledger platform to support smart contracts, or what Fabric calls "chaincode", authored in general-purpose programming languages such as Java, Go and Node.js, rather than constrained domain-specific languages (DSL). This means that most enterprises already have the skill set needed to develop smart contracts, and no additional training to learn a new language or DSL is needed. A smart contract functions as a trusted distributed application that gains its security/trust from the blockchain and the underlying consensus among the peers.

5.1.3. Permissioned characteristic

The Fabric platform is also permissioned, meaning that, unlike with a public permissionless network, the participants are known to each other, rather than anonymous and therefore fully untrusted. This means that while the participants may not fully trust one another (they may, for example, be competitors in the same industry), a network can be operated under a governance model that is built off of what trust does exist between participants. In such a permissioned context, the risk of a participant intentionally introducing malicious code through a smart contract is diminished. The participants are known to one another and all actions, whether submitting application transactions, modifying the configuration of the network or deploying a smart contract are recorded on the blockchain following an endorsement policy that was established for the network and relevant transaction type.

5.1.4. Consensus Protocol

One of the most important of the platform's differentiators is its support for pluggable consensus protocols that enable the platform to be more effectively customized to fit particular use cases and trust models. Fabric can leverage consensus protocols that do not require a native cryptocurrency to incent costly mining or to fuel smart contract execution. Avoidance of a cryptocurrency reduces some significant risk/attack vectors, and absence of cryptographic mining operations means that the platform can be deployed with roughly the same operational cost as any other distributed system.

5.1.5. Privacy

Hyperledger Fabric ensures confidentiality through its channel architecture and private data feature. The Membership Service Provider (MSP) allows for the definition of organizations, that are logical sets of peers and users. When a group of organizations within a channel needs to maintain privacy for data transactions and smart contracts (chaincode) from other organizations on that channel, they typically create a new channel consisting only of the organizations requiring access to the data and smart contract. However, this approach of creating separate channels introduces additional administrative overhead, such as managing chaincode versions, policies, and Membership Service Providers (MSPs). Moreover, it doesn't cater to scenarios where all channel participants should view all transactions while keeping a portion of their data private. To address these challenges, Fabric offers the option to create private data collections. These collections enable a defined subset of organizations within a channel to endorse, commit, or query private data without necessitating the creation of a separate channel. This feature streamlines administrative tasks and facilitates scenarios where data privacy is essential within a shared channel environment. A collection is the combination of two elements:

- The actual private data being transacted, which is transmitted peer-to-peer exclusively to the organization(s) authorized to access it. This data is conveyed via the gossip protocol and is stored in a private state database on the peers of authorized organizations. Authorized users and peers can access this data through chaincode. Importantly, the ordering service is not involved in this process and does not have visibility into the private data.
- The hash of the private data transacted, which undergoes endorsement, ordering, and is subsequently written to the ledger's world state on every peer within the channel. This hashed representation serves as proof of the transaction's occurrence and is utilized for state validation and potential audit purposes.

These two components enable data to remain private while still being endorsed and validated by all endorsement and order nodes.

5.1.6. Final thought

The combination of these differentiating design features makes Fabric one of the better performing platforms available today both in terms of transaction processing and transaction confirmation latency, and it enables privacy and confidentiality of transactions and the smart contracts that implement them.

5.2. Golang

Go is a statically typed, compiled high-level programming language developed at Google by Robert Griesemer, Rob Pike, and Ken Thompson. It bears syntactic resemblance to C, while offering memory safety, garbage collection, structural typing, and CSP-style concurrency. In this project, the Golang libraries provided by the Hyperledger foundation package page are utilized. Specifically, Golang is employed in both chaincode and application implementation. It was selected for its simplicity, efficiency, flexibility, usability, and cross-platform compatibility. Additionally, its ability to parallelize different operations and utilize channel variables for communication between parallelized operations makes it an ideal choice for this project.

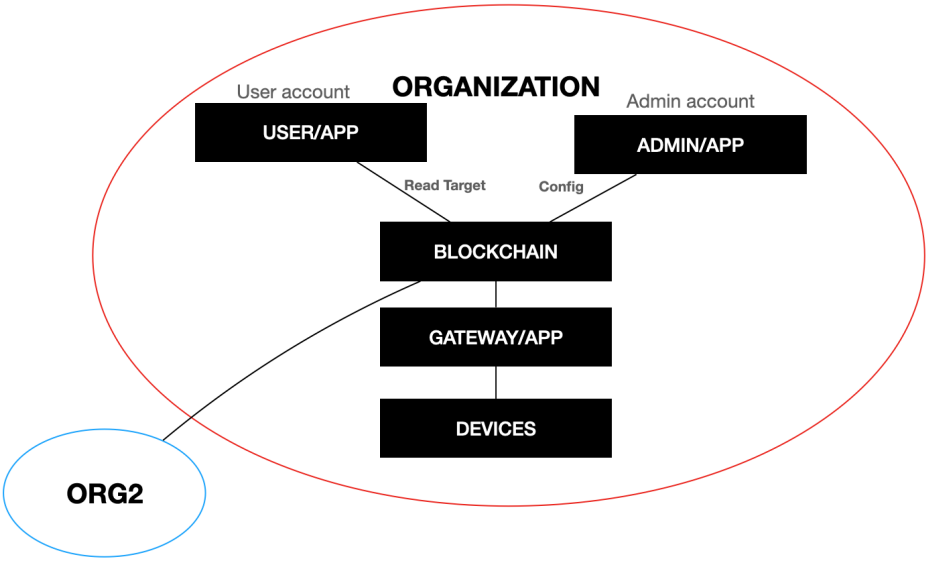

Figure 1. *Design of the project architecture.*

6. Our proposal

6.1. Architecture

The structure of the system (fig. 1) is composed of five elements:

- **Iot network**: This component represents the network of IoT devices responsible for capturing external information. In our project, these devices perform a ranging process with a specific target. During the prototype test, we represent this component using the CLOVES testbed. Additionally, there is a data encryption process applied to the information collected from the devices. This encryption process is simulated separately from the ranging process.
- **Blockchain**: This component is implemented using Hyperledger Fabric, where it stores all data related to devices and the target. The data is categorized into collections associated with specific organizations. Put simply, each administrator can create their own organization within Fabric and associate it with collections containing the target and devices. Consequently, only members of a given organization can access the associated data. These organizations and collections establish a form of access control and data privatization. Additionally, the blockchain hosts the chaincode necessary for various functionalities within the system. This includes calculations for target position by the gateway application, access to the target by users, and administrative tasks related to maintaining the data and system structure.
- **Gateway/App**: This component represents the nodes to which the devices connect to transmit data. These nodes host an application with administrative permissions. This application handles all communication operations with the blockchain and its chaincode. Its responsibilities include updating and creating device entities, as well as calculating and storing the position of the target in the blockchain.
- **User/App**: It is an entity with a user account within a specific organization. It has limited access, primarily to the target's position via a user application. Access to this account is granted by an organization administrator.
- **Admin/App**: It an entity that possesses an administrative account within a designated organization. Through an Admin application, it has the authority to modify device properties stored in the blockchain, access device and target data, and create new device entities. The individual holding this

account is often the same person who establishes the network of devices and initiates the gateway application to conduct experiments within the secure environment.

In the prototype, the blockchain component is simulated using Docker containers. For the project, a test network provided by the Hyperledger Fabric foundation is utilized. The Admin, User, and Gateway applications are implemented as Golang programs in the prototype. The network of devices is simulated using a physical testbed. While the applications and blockchain can be simulated and tested together in the prototype, the device network, including the ranging process and encryption, is simulated separately. In the prototype, the gateway application simulates to receive data from the devices, but only time considerations are taken into account.

6.2. Iot network

The testbed utilized for testing the IoT network is CLOVESwhich stands for Communication and Localization Testbed for Validation of Embedded Systems. CLOVES is a large-scale public testbed and the first of its kind for ultra-wideband (UWB) technology. UWB is a radio technology capable of utilizing very low energy levels for short-range, high-bandwidth communications across a wide spectrum of radio frequencies. This technology is increasingly used for applications such as sensor data collection, precise locating, and tracking.

Ultra-wideband technology enables the transmission of a significant amount of signal energy without causing interference with conventional narrowband and carrier wave transmissions in the same frequency band. Regulatory limits in many countries permit the efficient utilization of radio bandwidth, facilitating high-data-rate personal area network (PAN) wireless connectivity, longer-range low-data-rate applications, and the seamless coexistence of radar and imaging systems with existing communications systems.

In addition to UWB devices, Narrowband IEEE 802.15.4 devices are also deployed alongside them in the CLOVES testbed. All devices within the testbed are remotely accessible for firmware upload and collection of experiment logs. The indoor testbed areas offer various characteristics, such as narrow corridors and wide halls, catering to different types of studies related to communication and localization.

CLOVES is housed within the IoT testbed at the Department of Information Engineering and Computer Science at the University of Trento. For the project, the evb1000 device type is utilized. Some devices within the CLOVES network serve as anchors, while others act as the target for localization purposes.

To prepare the files required by CLOVES, the source codes for anchors and the target are compiled using the COOJA simulator. The resulting binary files (firmware for anchors and target) are then transferred to CLOVES for deployment and experimentation (fig. 2). You can access more general information about the process by following this link.

The concept revolves around anchors conducting two-way ranging (fig. 3) with the target, with each distance calculation being logged by the device. A parsing program then extracts crucial data from the log, including the device ID, target ID, calculated distance in millimeters, and the confidence level assigned to the distance. The confidence level is determined based on the Received Signal Strength Indicator (RSSI) from the target. If the signal power from the target exceeds a certain threshold, the device assigns maximum confidence. Conversely, if the signal power falls below another threshold, the device assigns minimum confidence. For cases falling outside of these thresholds, confidence is calculated using a formula based on the received RSSI. This approach is designed to account for variations in signal strength, where weaker signals may indicate greater distance or obstacles between the target and device, potentially leading to less precise measurements. This methodology draws inspiration from the foundational paper "A Trust Architecture for Blockchain in IoT" [1].

After parsing, the data is encrypted using XOR encryption. The encryption key is a symmetric key, stored both in the device and within the corresponding device entity in the blockchain. Before encryption, the data is hashed, and the resulting digest is sent along with the encrypted data. This approach ensures

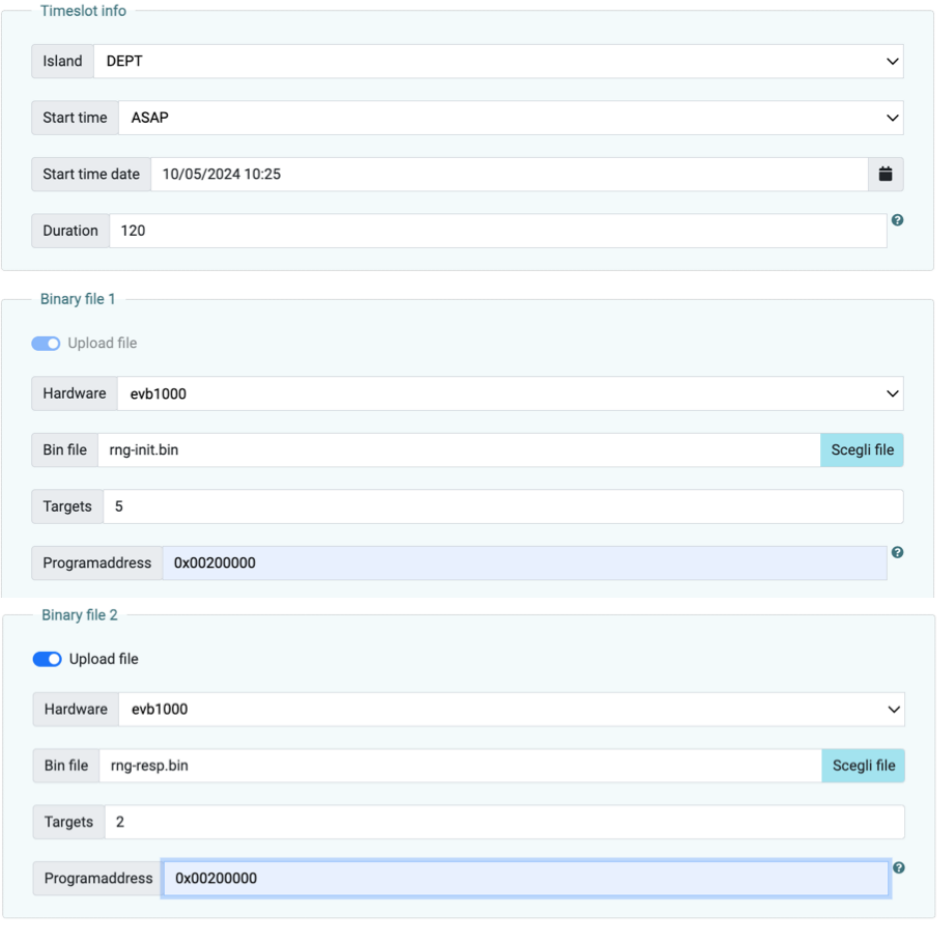

Figure 2. *How initialize an experiment in testbed.*

confidentiality through encryption and integrity through hashing. XOR encryption was chosen for its simplicity and performance. To ensure authenticity, the gateway application checks that the ID of the received data matches the ID of the device entity in the blockchain. This method was also chosen for its simplicity and performance.

In the specific case of the prototype, the three phases—ranging, parsing, and encryption—are separate, sequential processes connected through files. Each phase uses the file generated from the previous phase. Future projects building on this research may aim to integrate these phases more closely to create a more realistic and efficient prototype.

6.3. Blockchain

6.3.1. Test network (prototype network)

Hyperledger provides developers with a test network that includes all the essential elements for Hyperledger Fabric, along with some example applications. In this research, this network is customized to meet the specific requirements of the project. From a physical perspective (fig. 4), the network comprises the following components:

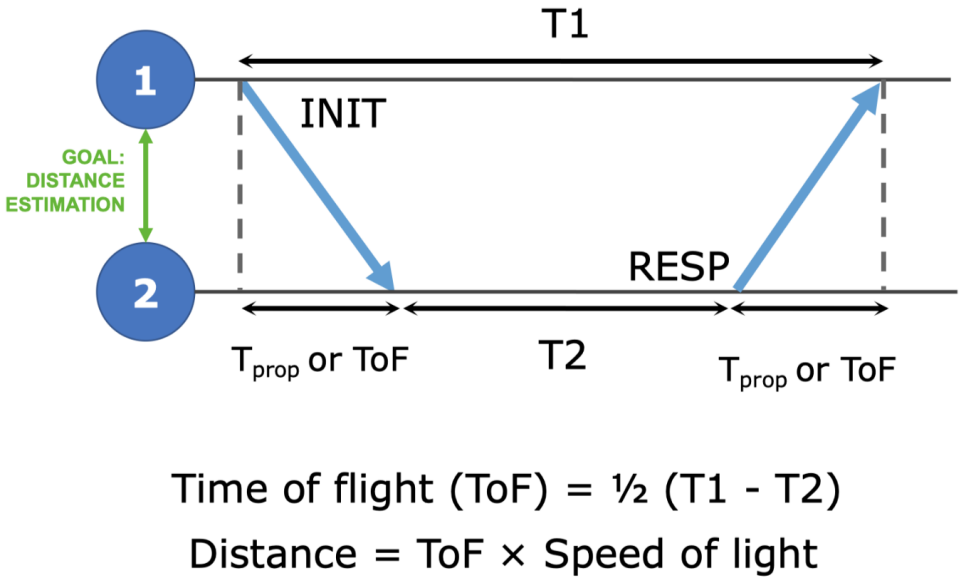

Figure 3. *Explanation of the two-way ranging.*

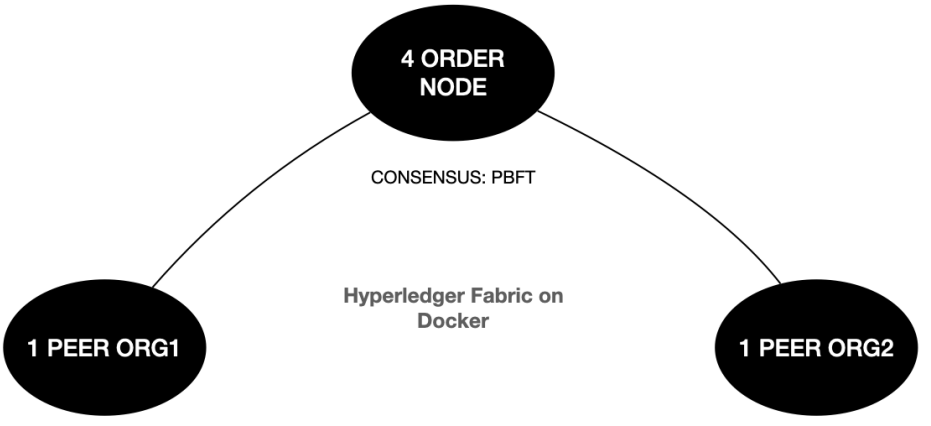

Figure 4. *Physical point of view of the prototype blockchain.*

- 4 Order Nodes that create blocks using the PBFT (Practical Byzantine Fault Tolerance) consensus algorithm, which provides countermeasures against potential malicious entities within the network.
- 1 peer node that handles the endorsement, proposal of transactions, and maintenance of collections for Organization 1.
- 1 peer node that handles the endorsement, proposal of transactions, and maintenance of collections for Organization 2.

In the prototype, instead of using a service for the creation of cryptographic material, a command named "cryptogen" is utilized. These components are simulated using Docker containers. The logical structure of the test network consists of the following components:

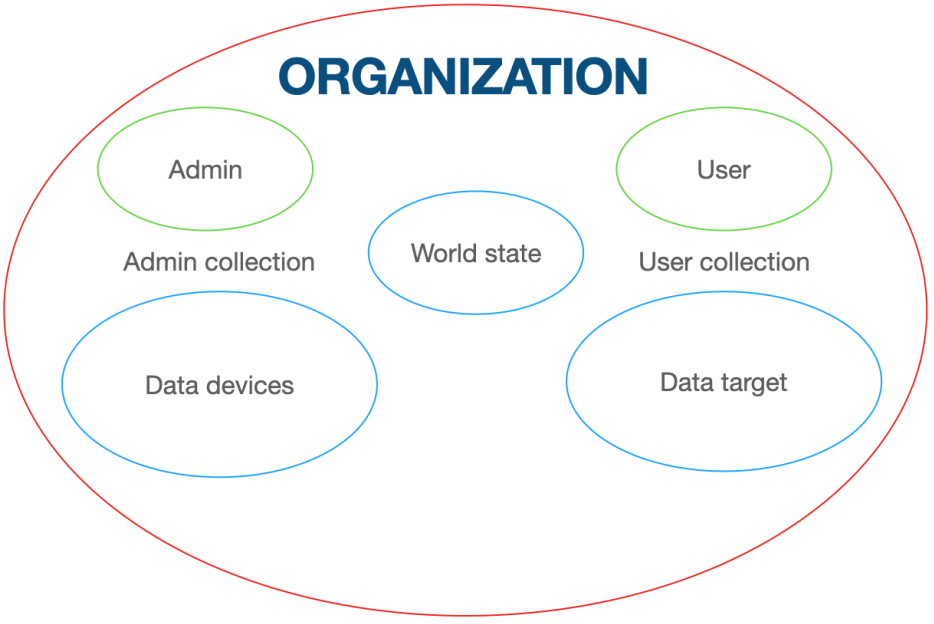

Figure 5. *Logical point of view of the prototype blockchain.*

- 2 organization (fig. 5)
- 1 admin client credential for every organization.
- 1 user client credential for every organization.
- 1 world state of the blockchain for all organizations.
- 2 collections for every organization. The collections of an organization can only be accessed by clients or peers that belong to that specific organization, ensuring data privacy and access control within the blockchain network.

  The devices collection contains device data and can only be accessed by Admin entities (client, application, or peer) within the same organization. A device entity in the collection includes the following fields: ID of the device; key for decrypting data from the device; X coordinate of the device (relative to the network map); Y coordinate of the device (relative to the network map); distance observed by the device; Confidence level of the device in its observation; evidence value, representing the support for the device's observation from neighboring devices; reputation of the device; trust level in the device's observation; IDs of neighboring devices.

  The target collection contains target data and can be accessed by User and Admin entities (client, application, or peer) within the same organization. However, only Admin entities can modify the data. A target entity in the collection includes the following fields: ID of the target; X coordinate of the target (relative to the network map); Y coordinate of the target (relative to the network map); timestamp of the target's position update; status indicator showing whether the target's position is updated or not.

6.3.2. Chaincode

The business logic of the blockchain is implemented in Golang due to its efficiency, usability, and simplicity. Within this chaincode, a single smart contract named "PositionContract" is defined and used by all organizations. Some data passed to the contract methods are kept in a transient state to ensure these data are not visible in the blockchain transaction corresponding to that contract call. This transient option ensures confidentiality of the transaction data. Overall, this contract provides the following functionalities:

- **CreateDevice(ctx contractapi.TransactionContextInterface, collection string)**: This method allows an admin to create a device within the same organization. The admin passes transient data, including the device ID, X and Y coordinates, decryption key, device neighborhood, and initial reputation. Additionally, the name of the collection to insert the device into is provided as a normal argument. During the method execution, several checks are performed: verification that the specified collection is a device collection; validation of the caller's admin status; confirmation of the correctness of the initialization data, such as ensuring the number of neighbors is greater than 1; checking that a device with the specified ID does not already exist.

  Any device data not provided by the admin is initialized to a default value of 0. Finally, the device is created with the provided initialization data.
- **UpdateDevice(ctx contractapi.TransactionContextInterface, collection string)**: This method allows an admin to update certain parameters of a device. The admin provides transient data, including the device ID, the new decryption key, and the new neighborhood. Additionally, the name of the collection containing the device to update is passed as a normal argument. During the execution of the method, several checks are performed: verification that the specified collection is indeed a device collection; validation of the caller's admin status; confirmation of the correctness of the update data, such as ensuring the number of neighbors is greater than 1; checking that a device with the specified ID exists in the blockchain.

  Any device data not provided by the admin is initialized to the existing value retrieved from the existent device with that ID. Finally, the device is updated with the provided information.
- **CreateTarget(ctx contractapi.TransactionContextInterface, collection string)**: This method allows an admin to create a target within the same organization. The admin provides transient data, including the target ID. Additionally, the name of the collection to insert the target into is passed as a normal argument. During the execution of the method, several checks are performed: verification that the specified collection is indeed a target collection; validation of the caller's admin status; confirmation of the correctness of the initialization data; checking that a target with the specified ID does not already exist.

  Any target data not provided by the admin is initialized to a default value of 0, except for the variable representing the updating status, which is initialized to false. Finally, the target is created with the initialization data.
- **UpdateDeviceObsConf(ctx contractapi.TransactionContextInterface, collection string)**: This method allows an admin to update a device's confidence and observation. The admin provides transient data, including the device ID, the calculated distance, the confidence in the device's observation, the minimum confidence, and the maximum confidence. Additionally, the name of the collection containing the device to update is passed as a normal argument. During the execution of the method, several checks are performed: verification that the specified collection is indeed a device collection; validation of the caller's admin status; confirmation of the correctness of the update data, such as ensuring the confidence is within the specified range; checking that a device with the specified ID exists.

  Any device data not provided by the admin is initialized to the existing value retrieved from the device with that ID. Finally, the existent device is updated with the provided information.
- **UpdateDeviceEvRep(ctx contractapi.TransactionContextInterface, collection string)**: This method enables an admin to update a device's observation evidence and trust, and device reputation. The admin provides as transient data: ID of the device; reward or penalty respectively in case of high evidence and high confidence, and in case of low evidence and high confidence; reward or penalty respectively, in case of high evidence and low confidence, and in case of low evidence and low confidence; the threshold indicating high or low evidence values; the maximum reputation. Additionally, the name of the collection containing the device to update is passed as a normal argument. During the execution of the method, several checks are performed: verification that the specified collection is indeed a device collection; validation of the caller's admin status; confirmation of the correctness of the update data, such as ensuring that the maximum confidence is positive; checking that a device with the specified ID exists.

After these checks, the evidence and trust of observation, as well as the reputation of the device, are calculated (see the Algorithm section for more detail). Any device data not provided by the admin or calculated are initialized to the existing value retrieved from the existent device with that ID. Finally, the device is updated with the provided information.

- **DeleteDevice(ctx contractapi.TransactionContextInterface, collection string)**: This method allows an admin to delete a specific device. The admin provides transient data, including the device ID. Additionally, the name of the collection containing the device to delete is passed as a normal argument. During the execution of the method, several checks are performed: verification that the specified collection is indeed a device collection; validation of the caller's admin status; confirmation of the correctness of the data for the deletion; checking that a device with the specified ID exists. Once these checks are completed, the device is deleted.
- **DeleteTarget(ctx contractapi.TransactionContextInterface, collection string)**: This method allows an admin to delete a specific target. The admin provides transient data, including the target ID. Additionally, the name of the collection containing the target to delete is passed as a normal argument. During the execution of the method, several checks are performed: verification that the specified collection is indeed a target collection; validation of the caller's admin status; confirmation of the correctness of the data for the deletion; checking that a target with the specified ID exists. Once these checks are completed, the target is deleted.
- **PositionTarget(ctx contractapi.TransactionContextInterface, collectionD string, collectionT string, dateU string)**: This method enables an admin to calculate the position of the target. The admin provides transient data, including the target ID, maximum error tolerable, and a list of device IDs to consider for the calculation. Additionally, as normal arguments, the name of the collection containing devices, the name of the collection containing the target, and the timestamp are passed. During the execution of the method, several checks are performed: verification that the first collection is indeed a device collection and the second collection is a target collection; validation of the caller's admin status; confirmation of the correctness of the data for the calculation; checking that a target with the specified ID exists.

  After these checks, the three devices with the greatest trust are retrieved. The distances calculated by these devices are used to calculate the position (see the Algorithm section for more detail). Any device data not provided by the admin or calculated are initialized to the existing value retrieved from the existent device with that ID. Finally, the target is updated with the calculated position, timestamp, and marked as updated.
- **ReadTarget(ctx contractapi.TransactionContextInterface, collection string)**: This method enables both an admin and a user to read target data. The ID of the target to read is provided as transient data, while the name of the collection where the target is stored is passed as a normal argument. During the execution of the method, the following steps are performed: verification that the specified collection is indeed a target collection; confirmation of the correctness of the transient data; retrieval of the target data with the provided ID; if the target exists, it is returned by the method.
- **ReadAllIDDevices(ctx contractapi.TransactionContextInterface, collection string)**: This method allows an admin to read all device IDs. The name of the collection where the devices are stored is passed as a normal argument. During the execution of the method, the following steps are performed: verification that the specified collection is indeed a device collection; confirmation that the caller is an admin; retrieval of a list containing the IDs of all devices in the collection, which is then returned.
- **ReadDevice(ctx contractapi.TransactionContextInterface, collection string)**: This method allows an admin to read specific device data. The ID of the device to read is provided as transient data, while the name of the collection where the device is stored is passed as a normal argument. During the execution of the method, the following steps are performed: verification that the specified collection is indeed a device collection; confirmation that the caller is an admin; validation of the correctness of the transient data; retrieval of the device data with the provided ID; if the device exists, its data is returned by the method.

6.4. Gateway application

This prototype application operates on a gateway and computes the target's position using the distances obtained from associated devices via the "PositionContract" smart contract. The application is developed in Golang due to its simplicity and efficiency. All functionalities within the application require an admin account to execute. Therefore, a gRPC connection is established between the application and the blockchain as an admin. Additionally, this prototype version of the application includes several methods designed exclusively for testing purposes.

6.4.1. Global

In the global section of the application, key definitions and parameters are outlined to manage device and target entities, customize the application, and set global variables for different account types. Device entity is defined with: id of the device; decryption Key; X and Y coordinates of the device on a predefined map; target distance observed from the device; confidence in device observation; evidence level supporting the device; reputation of the device; trust assigned to the device's observation; IDs of nearby devices.

Target entity is defined with: Id of the target; X and Y of the target coordinates on a predefined map; timestamp of the last update; indicator for target update status.

The type for device initialization is defined with: id of the device; decryption Key; X and Y coordinates of the device on a predefined map; IDs of nearby devices; Initial reputation of the device.

As global parameters there are: minimum and maximum confidence for a device observation; maximum confidence for a device; reward or penalty respectively in case of high evidence and high confidence, and in case of low evidence and high confidence (PRH); reward or penalty respectively, in case of high evidence and low confidence, and in case of low evidence and low confidence (PRL); thresholds indicating high or low confidence and evidence values; collections where devices and targets are stored; maximum error for position calculation; maximum reputation value; dimension of a batch of observations to compute the average.

At the end, there are two global variables tracking the number of current observations retrieved and successful updates in evidence, reputation, and trust. These two numbers are utilized to track the count of updates in their respective cases, enabling the main process to wait until these counts reach the expected value.

6.4.2. Main

The main process of the application follows a sequence of steps. Firstly, it establishes a connection with the blockchain, through specific methods. Then, it initializes the network by adding the initial devices and target. Subsequently, it enters a loop where it iterates through the devices in the system. For each device, it initiates the process to read the data stream. Afterward, it updates the observation in the device entity. Following this, it proceeds to update the evidence, trust of observations, and device reputation for each device. Once all devices have been updated, the position calculation begins. The loop then restarts. If any errors occur during the updating process, the main process blocks until the issue is resolved. However, if the position calculation encounters problems, the target is not updated, indicating potentially insufficient precision in the measurements. Additionally, for testing purposes, the prototype main calculates the time interval of each loop.

6.4.3. Functionalities

The application provides the following functionality:

- **initLedger(contract *client.Contract, devices []deviceTransientInt, idTarget string)**: This method initializes the ledger with initial device and target data.
- **readStreamDevice(idDevice string, c chan string)**: In the prototype scenario, this function simulates the reading of a stream of observations received from the device, considering possible transmission delay. It retrieves this data from a specific device file and operates concurrently with the rest of the

program. The function inserts the data into a channel variable that will be read by the observation updating process. If any errors occur during the reading process, it restarts the read operation until the entire file is successfully read.

- **EncryptDecrypt(input, key string)**: This method decrypts a message from a device using XOR encryption with a specified key.
- **getHash(s string)**: This method generates and returns the digest of a message from a device.
- **insertObs(contract *client.Contract, idDevice string, idTarget string, c chan string)**: This method handles the updating of confidence and distance observations for a specific device, specifying the maximum and minimum confidence levels. Executed as a goroutine to allow parallel processing with other device updates, it first retrieves the decryption key from the blockchain. Then, it enters a loop to read the channel variable where the encrypted data and digest is inserted. After decryption, it computes the hash of this data and compares the digest with the one from the channel variable to ensure data integrity. If the correspondence is confirmed, it verifies the correctness of the target and device IDs to maintain data authenticity. If all conditions are met, distances and confidences are accumulated until a certain number of observations is reached. In case of errors during execution, the function restarts by re-reading the decryption key, as unexpected data may have been received.

  Since the interval between data arrival and processing may differ, devices aren't updated for every observation to stabilize the situation. Instead, data is buffered and devices are updated with an average distance and confidence. The tradeoff between accuracy and performance can be controlled via a global variable: the dimension of the observation batch to compute the average. This dimension is required to be a pair integer since data from devices arrives in pairs: encrypted data and digest.
- **upDateDevices(contract *client.Contract, devices []string)**: This method updates the evidence and trust of observations and the reputation of devices. It utilizes global parameters for reputation rewards and penalties, a threshold to determine high or low evidence values, and the maximum reputation limit. To optimize performance, each device is updated within its own goroutine, allowing parallel processing. The upDateDevices method only completes once all goroutines have finished executing. If a goroutine encounters an error, it will restart, addressing potential conflicts with other processes accessing the blockchain.
- **upDateTarget(contract *client.Contract, idTarget string, devices []string)**: This method updates the position of the target by utilizing observation data from the specified devices. It also incorporates a global parameter representing the maximum permissible measurement error. If an error occurs during the process, the target position is not updated, as the measurement error may exceed acceptable limits.
- **seeAllIDDevice(contract *client.Contract)**: This method retrieves and returns all device IDs from the collection, serving as a valuable utility for the main process.

6.5. Admin application

This prototype application allows an admin client to read device data, read target data, update devices, and create devices within their own organization through the "PositionContract". The application is implemented in Golang, chosen for its simplicity and efficiency. A gRPC connection is established between the application and the blockchain, with the admin account facilitating these operations.

6.5.1. Global

In the global part of the application, we define various types and entities essential for initializing and updating devices, managing targets, and setting global parameters. The device initialization type initializes devices with unique identifiers, decryption keys, X and Y coordinates on a predefined map, a list of nearby device IDs, and initial reputation of device. The target entity represents a target with attributes such as id, X and Y coordinates on a predefined map, timestamp of the last update, and a flag indicating whether the target has been updated. The device update type is used for updating devices with attributes like device ID, new decryption key, and updated list of nearby device IDs. The device entity represents a device with attributes including device ID, decryption key, coordinates on a predefined

map, observed target distance, confidence level of the observation, evidence level of the observation, reputation of the device, trust level of the observation, and IDs of nearby devices. As global parameters there are essential parameters such as the name of the target collection, the name of the device collection, and information required to connect to the blockchain as an admin.

These definitions and parameters are fundamental for managing devices and targets effectively, facilitating interaction with the blockchain, and ensuring accurate representation of the network's state.

6.5.2. Main

The application initiates by establishing a connection with the blockchain, through specific methods. Following this, it prompts the admin to specify the desired operation: reading device information, reading target data, updating a device, or creating a new device within their organization. Once the admin selects an operation, they are prompted to provide the necessary information for the chosen task. Upon completion of the operation, the main process solicits a new choice from the admin, continuing the interactive workflow.

6.5.3. Functionalities

The application provides the following functionality:

- **insertDevice(contract *client.Contract, device deviceTransientInt)**: This method enables an admin to add a new device to the devices collection.
- **DeleteDevice(contract *client.Contract, idDevice string)**: This method allows an admin to delete the indicated device from the devices collection.
- **seeDataDevice(contract *client.Contract, idDevice string)**: This method allows an admin to read the indicated device from the devices collection.
- **updateDevice(contract *client.Contract, device deviceTransientUp)**: This method enables an admin to update the indicated device and its data.
- **seeDataTarget(contract *client.Contract, idTarget string)**: This method allows an admin to read the indicated target from the target collection.

6.6. User application

This prototype application enables a user client to access target data via the "PositionContract" contract. The application is written in Golang for its simplicity and efficiency. To execute the application, a grpc connection is established between the application and the blockchain as a user.

6.6.1. Global

In the global section of the application, there is a definition of a type for reading target data, consisting of the target's ID, its x and y coordinates on a predefined map, the timestamp of the update, and a variable indicating whether the target is updated. Additionally, the global parameters include the name of the target collection and all the necessary information to connect to the blockchain as a user.

6.6.2. Main

The application first establishes a connection with the blockchain, through specific methods. Then, it prompts the user to choose between terminating the process or reading a target. If the user selects the latter option, they are prompted to enter the ID of the target they wish to read. Once the reading is complete, the main process prompts the user for a new choice.

6.6.3. Functionalities

The application provides the following functionality:

- **seeDataTarget(contract *client.Contract, idTarget string)**: This method allows a user to read the indicated target from the target collection. If the target hasn't been updated, the function returns an error.

7. Algorithm

In this section, specific operations executed within the system are elaborated in detail:

7.1. Encryption of an observation from a device

This algorithm involves applying the XOR operation between a single character key and every character of the string passed to the algorithm to encrypt the string. The XOR operator's property allows for the same set of instructions to be used for decryption as well, making the algorithm simple and fast.

```
void encryptDecrypt(char inpString[],char key)
{
    // Define XOR key
    // Any character value will work
    char xorKey = key;
    // calculate length of input string
    int len = strlen(inpString);
    // perform XOR operation of key
    // with every caracter in string
    for (int i = 0; i < len; i++)
    {
        if(i == len-1){
            continue;
        }

        inpString[i] = inpString[i] ^ xorKey;

    }
}
```

Listing 1. *Encrypt and Decrypt device observation.*

7.2. Hashing of the observation from device

The hash function is applied to the observation from the device before encryption, and the resulting digest is sent along with the encrypted data to the appropriate gateway.

```
size_t getHash(const char* cp)
{
    size_t hash = 0;
    while (*cp)
        hash = (hash * 10) + *cp++ - '0';
    return hash;
}
```

Listing 2. *Get digest of the device observation.*

7.3. Two-way ranging

See figure 3. The two-way ranging process involves calculating the period T1, which is the time between when the device sends the first message and when it receives the response from the target. Additionally, the period T2 is calculated, which is the time between when the target receives the message and when it sends the response to the device. Then, the time of flight (ToF) between the device and target is determined using the formula $ToF = (T1 - T2)/2$. Finally, the distance is calculated as $distance = ToF * speedoflight$.

7.4. Calculation of the confidence

The confidence level is determined based on the Received Signal Strength Indicator (RSSI) received from the target device. This approach is inspired by the paper "A Trust Architecture for Blockchain in IoT" [1]. If the signal power from the target exceeds a certain threshold value RSSIUP, the device assigns a maximum confidence level of 1. Conversely, if the signal power falls below another threshold value RSSILOW, the device assigns a minimum confidence level of 0.4. For cases falling between these thresholds, the confidence level is calculated using the formula $9/4 + rssval/40$, where rssval represents the RSSI value calculated. These choices are based on the understanding that weaker signals may indicate greater distance or obstacles between the target and device, leading to potentially less precise measurements.

```
double conf = 0;

if(rssval>(RSSIUP)){
conf = 1;
}else{
if(rssval<(RSSIINF){
  conf = 0.4;
}else{
  conf = 9/4 + rssval/40;
}

}
```

Listing 3. *Calculate the confidence of device observation.*

7.5. Calculation of the evidence of a device observation

The concept originates from the paper "A Trust Architecture for Blockchain in IoT" [1]. It leverages the correlation among device observations to compute the evidence component for observation trust. The evidence (Evi) for an observation is determined based on data received from neighboring devices, whose information is stored in the device entity on the blockchain. If a neighbor's observation supports a device's observation, it increases the device's evidence by an amount proportional to the neighbor's observation confidence. Conversely, if a neighbor's observation contradicts the device's observation, it decreases the device's evidence by a value proportional to the neighbor's observation confidence. The final device confidence is adjusted according to the number of neighbors (num).

In this project, a neighbor's observation regarding the distance to the target supports the device's observation if the triangle formed by the edge between the two devices and the two edges between the devices and the target satisfies the triangle inequality. If this property is not upheld, it indicates that the distance calculated by these devices cannot pertain to the same target.

```
    var Evi float32 //Evidence of device
```

```
    var num int //number of neighbour
    var tmp float32
    num = 0
    tmp = 0

    for _, idN := range deviceData.Neigh {
        num = num + 1
        // Check if Device already exists
        DeviceAsBytes, err = ctx.GetStub().GetPrivateData(collection,
            idN)
        if err != nil {
            return fmt.Errorf("failed to get device: %v", err)
        } else if DeviceAsBytes == nil {
            fmt.Println("Device not exist: " + idN)
            return fmt.Errorf("this device not exist: " + idN)
        }

        //Unmarshal the device data
        var deviceNeigh Device
        err = json.Unmarshal(DeviceAsBytes, &deviceNeigh)
        if err != nil {
            return fmt.Errorf("failed to unmarshal JSON: %v", err)
        }

        //Check triangle inequality
        if (float64(deviceData.Obs+deviceNeigh.Obs) >= Distance(New(
            deviceData.X, deviceData.Y), (New(deviceNeigh.X,
            deviceNeigh.Y)))*1000) && ((Distance(New(deviceData.X,
            deviceData.Y), (New(deviceNeigh.X, deviceNeigh.Y)))*1000
            + float64(deviceNeigh.Obs)) >= float64(deviceData.Obs)) {
            tmp = tmp + deviceNeigh.Conf
        } else {
            tmp = tmp - deviceNeigh.Conf
        }
    }
    //Calculate the evidence of the device
    Evi = tmp * (float32(1) / float32(num))
```

Listing 4. *Calculate the evidence of a device reputation.*

7.6. Calculation device reputation

The concept is derived from the foundational paper "A Trust Architecture for Blockchain in IoT" [1]. It elucidates the intricate relationship between the trust level in an observation and the long-term reputation of its data source. A higher reputation of a node engenders greater trust in the node's observations. The reputation of a device evolves over time, with the guiding principle of reputation updates (based on the observation confidence and the evidence of neighbours observations) being as follows: the magnitude of reputation reward or penalty should be proportional to the reported confidence.

When a device exhibits high confidence in its observation (i.e., Conf $\geq$ ThresholdConf) and the observation is corroborated by other nodes (i.e., Evi $\geq$ ThresholdEv), the device merits a substantial increase (PRH) in its reputation. Conversely, if a device provides observations with high confidence that

are contradicted by other nodes, its reputation should undergo a significant decline. Similarly, rewards and penalties (PRL) for observations with low confidence should be less pronounced, i.e., PRL < PRH.

In the context of this project, measures are implemented to prevent reputations from becoming excessively high or low. Specifically, reputation cannot be reduced below 0, nor can it exceed a maximum value (MaxRep).

```
var Repu int //Reputation of device
Repu = deviceData.Rep

//Update the reputation of the device
if deviceData.Conf >= deviceInput.ThreashConf && Evi >= deviceInput.
    ThreashEv {
    Repu = Repu + deviceInput.PRH
    if Repu > deviceInput.MaxRep {
        Repu = deviceInput.MaxRep
    }
}
if deviceData.Conf < deviceInput.ThreashConf && Evi >= deviceInput.
    ThreashEv {
    Repu = Repu + deviceInput.PRL
    if Repu > deviceInput.MaxRep {
        Repu = deviceInput.MaxRep
    }
}
if deviceData.Conf >= deviceInput.ThreashConf && Evi < deviceInput.
    ThreashEv {
    Repu = Repu - (deviceInput.PRH + 1)
    if Repu < deviceInput.MaxRep {
        Repu = deviceInput.MaxRep
    }
}
if deviceData.Conf < deviceInput.ThreashConf && Evi < deviceInput.
    ThreashEv {
    Repu = Repu - (deviceInput.PRL + 1)
    if Repu < deviceInput.MaxRep {
        Repu = deviceInput.MaxRep
    }
}
```

Listing 5. *Calculate the reputation of a device.*

7.7. Calculation of the device observation trust

The concept is derived from the foundational paper "A Trust Architecture for Blockchain in IoT" [1]. Due to the interrelation between a device's observation confidence and evidence, and device reputation, the trustworthiness of a device's observation can be calculated as follows: Trust = Confidence * Reputation * Evidence.

7.8. Calculation of target position

To determine the target's position, a form of multilateration is employed, involving the intersection of circumferences (see Fig. 6). In this project, the positions of the three most trusted devices are considered as the centers of these circumferences, with the calculated distances serving as their radii.

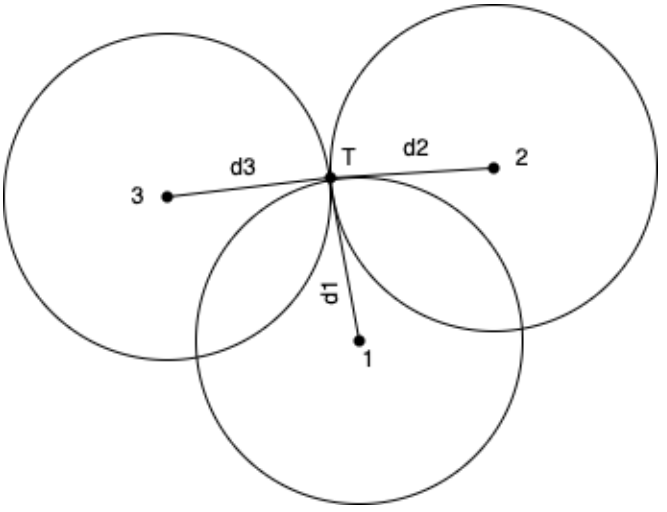

Figure 6. *Multilateration.*

Initially, the intersection between the circumferences generated by the first two devices is computed. If no intersection occurs, the target's position cannot be calculated. If the intersection yields a single point, it becomes the target's position. In cases where two points are obtained, both are plugged into the analytical equation of the circumference defined by the third device ($x^2 + y^2 - r^2 = 0$). Subsequently, the point that yields a solution closest to 0 and is within the maximum error threshold (Thresh) is selected as the target's position. If the value derived from the expression $x^2 + y^2 - r^2$ exceeds the maximum error, the position is not calculated due to the anticipated high margin of error.

```
// Stable sort by trust and descending order
slices.SortStableFunc(devicesDistcol, func(a devicesDist, b
    devicesDist) int {
    return cmp.Compare(b.Trust, a.Trust)
})

//take only 3 devices wiht higher trust

var d1 devicesDist
var d2 devicesDist
var d3 devicesDist

for i, d := range devicesDistcol {
    if i == 0 {
        d1 = d
    }
    if i == 1 {
        d2 = d
    }
    if i == 2 {
        d3 = d
    }
}

x1 := float64(d1.X)                    //Coordinate x
y1 := float64(d1.Y)                    //Coordinate y
```

```
x2 := float64(d2.X)                          //Coordinate x
y2 := float64(d2.Y)                          //Coordinate y
x3 := float64(d3.X)                          //Coordinate x
y3 := float64(d3.Y)                          //Coordinate y
r1 := float64(d1.Distance) / float64(1000) //Distance observed
r2 := float64(d2.Distance) / float64(1000) //Distance observed
r3 := float64(d3.Distance) / float64(1000) //Distance observed

var xtarg float64 //Coordinate x of the target
var ytarg float64 //Coordinate y of the target

// Calculate the distance between the centers of the circles
d := math.Sqrt(math.Pow(x1-x2, float64(2)) + math.Pow(y1-y2, float64
    (2)))

// Check if circles do not intersect
if d > r1+r2 || d < math.Abs(r1-r2) {
    return fmt.Errorf("no solution")
}

// Check if circles are tangent to each other
if d == r1+r2 || d == math.Abs(r1-r2) {
    fmt.Println("The circles are tangent to each other.")
    // Calculate the coordinates of the tangent point using the
        midpoint formula
    xtarg = (x1 + x2) / float64(2)
    ytarg = (y1 + y2) / float64(2)

    // Make submitting client the owner
    target := TargetInfo{
        ID:   idTarget,
        X:    float32(xtarg),
        Y:    float32(ytarg),
        Date: time.Now().Format(time.DateTime),
        Upd:  true,
    }
    targetJSONasBytes, err := json.Marshal(target)
    if err != nil {
        return fmt.Errorf("failed to marshal target into JSON: %v",
            err)
    }

    //Put the target in the collection
    err = ctx.GetStub().PutPrivateData(collectionT, idTarget,
        targetJSONasBytes)
    if err != nil {
        return fmt.Errorf("failed to put target into private data
            collecton: %v", err)
    }
} else {
    // Calculate the distance from center_p to the intersection line
    a := (r1*r1 - r2*r2 + d*d) / (float64(2) * d)
    // Calculate the distance from intersection point to the
        intersection line
    h := math.Sqrt(r1*r1 - a*a)
```

```
    // Calculate the intersection points
    xposs1 := x1 + (a/d)*(x2-x1) + (h/d)*(y2-y1)
    yposs1 := y1 + (a/d)*(y2-y1) - (h/d)*(x2-x1)
    xposs2 := x1 + (a/d)*(x2-x1) - (h/d)*(y2-y1)
    yposs2 := y1 + (a/d)*(y2-y1) + (h/d)*(x2-x1)
    //Select the best solution
    minv1 := math.Abs((math.Pow(x3-xposs1, float64(2)) + math.Pow(y3-
        yposs1, float64(2))) - r3*r3)
    minv2 := math.Abs((math.Pow(x3-xposs2, float64(2)) + math.Pow(y3-
        yposs2, float64(2))) - r3*r3)
    if minv1 <= minv2 {
        if minv1 >= tresh {
            return fmt.Errorf("no solution")
        }
        xtarg = xposs1
        ytarg = yposs1
    } else {
        if minv2 >= tresh {
            return fmt.Errorf("no solution")
        }
        xtarg = xposs2
        ytarg = yposs2
    }
```

Listing 6. *Calculate the target position.*

8. Example of use case

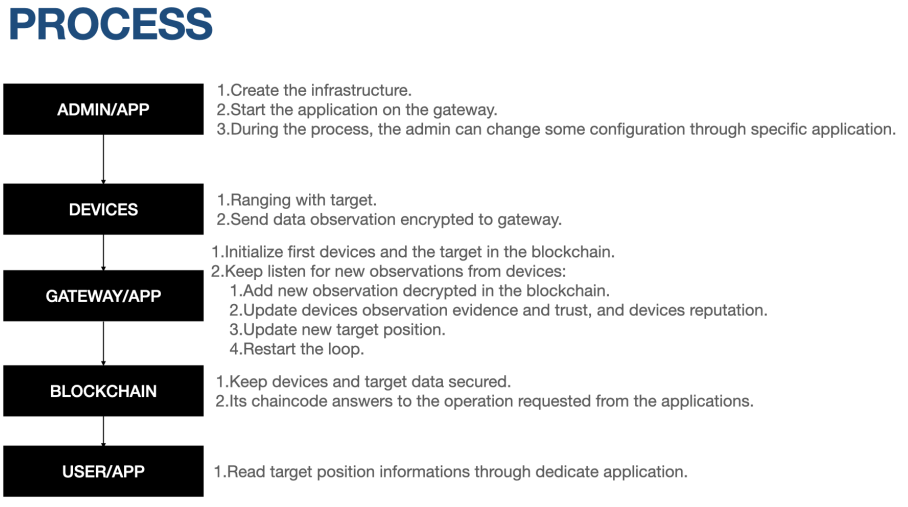

Figure 7. *Example of deployment and use of the prototype architecture.*

In this section, an example of deployment, implementation, and execution of the system is presented (see Fig. 7):

8.1. Initialization

Initially, the Hyperledger Fabric infrastructure is deployed with several nodes and a channel, along with the contract "PositionContract." Next, a client registers as the admin of their organization on the blockchain, obtaining an admin account. Within the organization, the client establishes the necessary collections and specifies the peers that will be connected to the blockchain, either by creating new ones or utilizing existing ones.

The client proceeds to deploy the devices and target (The devices network), creating an application based on the prototype gateway application. This application includes the initial devices and the target. The admin then determines which peer or peers will serve as the gateway(s) and connects them to the devices.

Finally, the admin initiates the application on the gateway using own account and starts the firmware for ranging deployed in the devices. All operations performed on the gateway reference the contract already present in the blockchain.

8.2. Processing

The devices perform two-way ranging with the target and send their results encrypted to the gateway. The gateway begins by checking the devices in the system. For each device, it receives and decrypts the data, buffering enough observations to compute distance and confidence averages, which are then updated on the blockchain. Subsequently, the gateway updates the devices' reputation, as well as the evidence and trust for device observations. Finally, it computes the target's position and updates it on the blockchain. Once completed, the gateway returns to checking the devices in the system and capturing observations in the buffer, thus restarting the cycle.

8.3. Admin client

An admin can create and operate their own application based on the prototype admin application. This application establishes a connection with the "PositionContract" on the blockchain with the admin account. Through this interface, the admin gains control over certain aspects of the blockchain infrastructure. Specifically, the admin can read device data, access target data, update device information, and create new devices as needed.

8.4. User client

An admin has the authority to register another client with a user account. Subsequently, a user can independently create and operate their own application based on the prototype user application. This application establishes a connection with the "PositionContract" on the blockchain with the user account. Through this interface, the user gains the ability to access and retrieve the position of the target whenever it is updated.

9. Evaluation

9.1. Objectives

The prototype undergoes analysis across three primary dimensions: correctness, privacy, and performance.

- **Correctness**: This topic focuses on rigorous testing to ensure the accuracy of calculations for device observation trust, target positioning, as well as the operations for reading and updating device data and target data.

Table 1. *The input devices data for the tests.*

Device	Data						
	X	Y	Neighbour	Decryption key	Confidence	Distance	Reputation
1	3	2	(2,3)	P	1	4242 mm	5
2	10	4	(1,3)	P	1	4123 mm	5
3	5	8	(2,1)	P	1	3162 mm	5
4	1	1	(1,1)	P	8	4242 mm	5
5	3	2	(3,2)	P	1	2 mm	5

- **Privacy**: In this segment, the prototype is subjected to various scenarios to evaluate its response when privacy is compromised.
- **Performance**: In this section, the performance of the prototype is evaluated across different components, including the chaincode, applications, and the device network.

9.2. Design

For each topic discussed below, we'll first outline the input parameters for the tests and then describe the tools and structure of the tests.

9.2.1. Test inputs

The inputs are:

- **Correctness and Privacy tests inputs**: For these tests, we use several test devices with IDs 1, 2, 3, 4, and 5, along with a test target with ID 7. The first three devices are designed to provide accurate observations for localizing a specific target with ID 7. See Table 1 for the input data of the devices. All these devices have an initial reputation 5 and key of decryption "P". The device 1 has coordinates (3,2) and neighbors 2,3, device 2 has coordinates (10,4) and neighbors 1,3, device 3 has coordinates (5,8) and neighbors 1,2, device 4 has coordinates (1,1) and neighbors 2,1, device 5 has coordinates (3,2) and neighbors 3,2. The device 5 has the same position as device 1 to study the case of device 1 with incorrect observation. The device 4, instead, has a wrong confidence value. Respectively the observation and confidence of devices are in order: 4242 mm and 1; 4123 mm and 1; 3162 mm and 1; 4242 mm and 8; 2 mm and 1. All these tests are in the context of organization 1. The values (table 2) for the application parameters used in the tests are: 0.4 as minimum confidence; 1 as maximum confidence; 2 as reward or penalty respectively in case of high evidence and high confidence, and in case of low evidence and high confidence; 1 as reward or penalty respectively, in case of high evidence and low confidence, and in case of low evidence and low confidence; 0.7 as the threshold that indicates when a confidence value is high or low; 0 as the threshold that indicates when an evidence value is high or low; "DeviceAdmin1PrivateCollection" as collection where are memorized the devices; "TargetOrg1PrivateCollection" as collection where are memorized the target; 0.01 as maximum position error; 20 as maximum reputation; 6 as dimension of a batch of observations to retrieve before compute the average; 100 ms as time to retrieve a device observation

  Those parameters that are also found in "A Trust Architecture for Blockchain in IoT" [1] have been assigned the same values as those presented in the paper for the sake of simplicity and to maintain consistency with the experimental setup described in the paper.
- **Performance tests inputs**: For these tests are considerated all the configuration files (check config/-core.yaml and Network/bft-confing/confitgx.yaml in GitHub page) of the test network. The specific devices and target data used as input are not crucial for evaluating these tests.

Table 2. *Applications parameters for the tests.*

Parameter	Value
Max_{Conf}	1
Min_{Conf}	0.4
PRH	2
PRL	1
$Thresh_{Conf}$	0.7
$Thresh_{Ev}$	0
$Collection_{Devices}$	DeviceAdmin1PrivateCollection
$Collection_{Target}$	TargetOrg1PrivateCollection
$MaxError_{Pos}$	0.01
Max_{Rep}	20
$DimensionBatch_{Obs}$	6
TimeReadObs	100 ms

9.2.2. Test tools and structure

The tools and structure of the tests are:

- **Correctness tests structure and tools**: In this topic, the correctness of various application functionalities is tested using the Golang testing package. Some functionalities, such as device observation, evidence and trust updating, and device reputation updating, are tested in their test versions without considering performance strategies such as multithreading, as they are not pertinent to this test topic. The test functions can be found in the files "Position_test.go" and "Conf_test.go". Below is an outline of the design of the test functions:
  - **TestInitAsAdminOrg1**: This is the test of blockchain initialization as the admin of organization 1 with devices 1, 2, and 3, and target 7.
  - **TestAddObsAsAdminOrg1**: This is the test of adding observations and observation confidences of devices 1, 2, and 3 as the admin of organization 1.
  - **TestAddObsWithErrConfAsAdminOrg1**: This is the test of adding observations and observation confidences of devices 1, 2, 3, and 4 as the admin of organization 1. In this test, device 4 has a wrong confidence value. The expected outcome is a message indicating that it cannot be possible to insert the observation of device 4 because the confidence isn't between the minimum and maximum confidence.
  - **TestAddEvAsAdminOrg1**: This is the test of updating evidence and trust of observations, and updating the reputations of devices 1, 2, and 3 as the Admin of organization 1. Considering the test application parameters, the initialization devices, and the methodology to do this operation (see Algorithm section), any device entity must have a reputation of 7 and evidence of 1.
  - **TestPosCalculateAsAdminOrg1**: This is the test of calculating the position of the target as the Admin of organization 1, using devices 1, 2, and 3. Considering the test application parameters, the initialization of devices, and the methodology to perform this operation (see Algorithm section), the target position is updated.
  - **TestSeeTargetAsUserOrg1**: This is the test of retrieving the data of an updated target as a user of organization 1.
  - **TestSeeTargetNOTUpdatedAsUserOrg1**: This is the test of retrieving the data of a target that has not been updated as a user of organization 1. The test must produce a message indicating that a user cannot read a target if it hasn't been updated.

  - **TestAddEvWithIncorrectObsAsAdminOrg1**: This is the test of updating the evidence and trust of observations, and the updating of reputations of devices 5, 2, and 3 as the Admin of organization 1. Device 5 provides a distance that cannot correspond to the same target as devices 2 and 3 because it doesn't respect the triangular inequality. Considering the test application parameters, the initialization of devices, and the methodology for this operation (see Algorithm section), the evidence and reputation of device 5 become -1 and 2, respectively.
  - **TestPosCalculateWithIncorrectObsAsAdminOrg1**: This is the test of updating the position of the target with devices 5, 2, and 3 as the Admin of organization 1. Device 5 provides a distance that cannot correspond to the same target as devices 2 and 3 because it doesn't respect the triangular inequality. Considering the test application parameters, the initialization of devices, and the methodology for this operation (see Algorithm section), the test must indicate that the position cannot be calculated because the calculations are incorrect. The multilateration does not produce an acceptable result due to the faulty observation from device 5.
  - **TestPosCalculateWith4devicesAsAdminOrg1**: This is the test of updating the position of the target with devices 1, 5, 2, and 3 as the Admin of organization 1. Device 5 provides a distance that cannot correspond to the same target as devices 2 and 3 because it doesn't respect the triangular inequality. However, the target position will be updated because the devices with the highest trust are considered, namely devices 1, 2, and 3.
  - **TestUpdateDeviceAsAdminOrg1**: This is the test of updating the device decryption key or device neighborhood by an Admin of organization 1.
  - **TestSeeDeviceAsAdminOrg1**: This is the test of retrieving the data of a device as an Admin of organization 1.

- **Privacy tests structure and tools**: In this topic, we test how the system handles potential privacy breaches across different application functionalities using the Go testing package. The tests are located in the files "Position_test.go" and "Conf_test.go". The design of the test functions is as follows:

  - **TestInitAsUserOrg1**: In this test, we assess whether a user from organization 1 can access the devices collection of the same organization during the initialization process as a user of organization 1. The test should confirm that a user cannot access the devices collection.
  - **TestInitAsAdminOrg2**: In this test, we verify whether a client from organization 2 can access the devices collection of organization 1 through blockchain initialization as an admin of organization 2. The test should indicate that a client cannot access the devices collection of an organization to which it does not belong.
  - **TestSeeTargetAsAdminOrg2**: In this test, we examine whether a client from organization 2 can access the target collection of organization 1 by retrieving target data of organization 1 as an admin of organization 2. The test should indicate that a client cannot access the target collection of an organization to which it does not belong.

- **Performance tests structure and tools**: The performance tests are executed in various environments and with diverse tools:

  - **Application position computation**: In this test, the time taken to complete a cycle of target position computation by the gateway application is measured. This is done during a simulation of the gateway application, which also includes the addition of observations and confidence, and the updating of evidence, reputation, and trust.
  - **Ranging computation**: In this test, the average time taken to perform a ranging operation in the testbed environment between a device and the target is calculated. This calculation is derived from logs of the testbed experiment using a Python parsing code.
  - **Chaincode computations**: In the analysis of the chaincode operations performance using the Hyperledger Caliper tool, the following metrics are computed:

Table 3. *Performance tests results.*

Type	Result	
	AvgTime (ms)	Throughput (tps)
Application position computation	9000 ± 1000	–
Ranging computation	0.5	–
Chaincode comput. reading target	20 ± 20	73
Chaincode comput. reading device	50 ± 20	28
Chaincode comput. updating device	2100 ± 20	0.5
Chaincode comput. adding observation	2070 ± 20	0.5
Chaincode comput. evidence, reputation and trust	2080 ± 20	0.5
Chaincode comput. position computation	2110 ± 20	0.5

* Throughput and average time to execute the chaincode operation for adding a device observation and device observation confidence are computed.
* Throughput and average time to execute the chaincode operation for updating the evidence and trust of an observation, as well as the reputation of a device, are computed.
* Throughput and average time to execute the chaincode operation for position computation are measured.
* Throughput and average time to execute the chaincode operation for reading the target are measured.
* Throughput and average time to execute the chaincode operation for reading a device are measured.
* Throughput and average time to execute the chaincode operation of updating a device by an Admin are measured.

9.2.3. Execution

The procedure for executing the tests or simulation can be found on the GitHub page. The GitHub repository provides a detailed explanation of the directory structure and contents of the prototype too.

10. Results

In this section, the results of the tests are presented and discussed.

10.1. Exposition

10.1.1. Correctness, Security and Privacy

All these test types yielded positive results (PASS), verifying the system's intended correctness, security, and privacy.

10.1.2. Performance

See the results in Table 3. Below is a brief explanation of the results:

- The average time to update the target position, including the addition of observations and confidence, and the updating of evidence, reputation, and trust, using the prototype application is approximately 9 seconds.
- The average time to perform a ranging operation in the testbed environment between a device and the target is approximately 0.5 milliseconds.

- In the case of considering only the chaincode operations, the performance metrics are as follows: For reading the target position, the average time is 20 ms with a throughput of 73 tps (transactions per second). Reading a device takes an average time of 50 ms with a throughput of 28 tps. Updating a device requires an average time of 2100 ms with a throughput of 0.5 tps. Adding an observation and confidence takes around 2070 ms with a throughput of 0.5 tps. Updating the trust, evidence, and reputation consumes about 2080 ms with a throughput of 0.5 tps. Finally, computing the target position involves an average time of 2110 ms with a throughput of 0.5 tps.

10.2. Discussion

10.2.1. Correctness

In terms of correctness and functionality from the tests and simulation derive the following positive aspects:

- The successful execution of all application and chaincode operations demonstrates the robustness and reliability of the system.
- The synchronization achieved by the Gateway application, which waits for a certain number of observations from every device before proceeding with the computation, ensures consistency and accuracy in the processing of device observations.
- The use of Ultra-wideband (UWB) radio technology in the CLOVES testbed ensures that the prototype devices network operates with precision and speed, facilitating accurate and rapid measurements.

On the downside, the tests and simulation revealed a few drawbacks:

- The simulation doesn't fully replicate the connection between the testbed network and the application, limiting the testing of certain system aspects.
- The network exhibits some rigidity post-initialization; while new devices can be created, there's no provision for device deletion, and only one target can be accommodated without the ability to add more.
- Given the prototype nature of the tests, certain conditions of a real-world scenario are not accounted for.
- The multilateration algorithm used assumes that the devices and target are all located in the same plane, which can limit its applicability in certain scenarios.

10.2.2. Security and Privacy

In terms of security and privacy from the tests and simulation derive the following positive aspects:

- The successful execution of all application and chaincode operations demonstrates the security and privacy of the system.
- This system mitigates the risk of malicious devices through trust calculation. However, its effectiveness relies heavily on the presence of numerous benign devices within the network.
- This system mitigates the risk of malicious architecture internal nodes through the PBFT consensus protocol. However, its effectiveness relies heavily on the presence of numerous benign nodes within the network.
- This system ensures privacy by segregating data between different experiments using the collection concept. As a result, a client or peer cannot access the data of devices and targets from an experiment to which they do not belong.
- This system provides varying levels of privacy, distinguishing between users and admins. Users have access only to target data, while admins have access to both target and device data, thanks to the collection concept.
- The blockchain ensures the availability and integrity of the data, providing a secure and tamper-proof environment.

- The data transmitted between devices and the gateway are encrypted and hashed, ensuring both integrity and confidentiality.

On the downside, the tests and simulation revealed a few drawbacks:

- The system currently lacks security measures during the ranging process between devices and the target.
- The current implementation lacks an efficient and highly secure method to ensure the reliability of data transmitted between devices and the gateway. Currently, only the device and target ID of the arrived message are checked, which may not be sufficient for ensuring complete reliability.
- The service's availability is impacted by its dependency on receiving data from all necessary devices to complete the target position calculation. A distributed denial-of-service (DDoS) attack targeting a single device can disrupt data transmission, effectively blocking the service. While this approach ensures the service's security and precision by considering all devices, it also introduces a vulnerability wherein the service can be stalled if any device fails to transmit data.

10.2.3. Performance

In terms of performance from the tests and simulation derive the following:

- Good performance is achieved by utilizing ultra-wideband technology, which enables rapid ranging calculations.
- The application's position computation interval is quite large, which can lead to significant desynchronization between the real and calculated positions of the target. To mitigate this, the application buffers the data received from devices, reducing the impact of differences in data arrival and target position calculation speeds. Additionally, the application attempts to parallelize operations such as adding observations and confidence, and updating evidence, trust, and reputation, which seems to improve the time for position computation. In fact, considering the sum of average times for all chaincode operations needed to calculate the target, it amounts to approximately 14 seconds, which is less than the average time calculated from the application, 9 seconds.

11. Conclusion

This paper introduces a prototype IoT and blockchain infrastructure for conducting experiments, specifically in localization, in a trustworthy and privacy-conscious manner. It builds upon the concepts outlined in the paper "A Trust Architecture for Blockchain in IoT" [1], which provides the foundation and principal method for managing the issue of malicious devices. The resulting prototype offers a solid foundation that can be readily downloaded, tested, and utilized to develop more realistic use cases.

Despite its strengths, the prototype does have several limitations. These include the time required for target position calculation, the restriction to a single target, unrealistic simulation of communication between the gateway and devices, lack of security between devices and targets, inflexibility in the structure, inability to remove devices from the blockchain, and the potential for the gateway main process to become blocked if a device stops transmitting.

However, the project successfully achieves its primary goal of enhancing the privacy aspects through the concept of collections in Hyperledger Fabric. It introduces two levels of privacy: between localization experiments and between administrators and users. Additionally, the prototype implements device trust calculation as outlined in "A Trust Architecture for Blockchain in IoT" [1] to mitigate the risk of malicious devices, along with other techniques to enhance the system's trustworthiness, such as encrypting data transmitted between devices and the gateway. Furthermore, the prototype implements verifiable processes for calculating the target's position, allowing administrators to manage the system to some extent, and enabling users to access target data. While not a deployable version, this system test version serves as a valuable foundation for future projects aiming to develop more efficient, realistic, and secure systems in a similar domain.

Acknowledgments. We express our gratitude to Professor Hasan Omar for his invaluable guidance and supervision throughout this project at INSA Lyon University. His expertise and support have been instrumental in shaping our research and achieving our goals.

Data Availability Statement. Prototype data and code can be found in Github page: https://github.com/GuglielmoZocca/TrustLocalizationProject.

Ethical Standards. The research meets all ethical guidelines, including adherence to the legal requirements of the study country.

References

[1] Dedeoglu, V., Jurdak, R., Putra, G. D., Dorri, A., & Kanhere, S. S. (2019, November). A trust architecture for blockchain in IoT. In Proceedings of the 16th EAI international conference on mobile and ubiquitous systems: computing, networking and services (pp. 190-199).

[2] He, Y., Wang, Y., Qiu, C., Lin, Q., Li, J., & Ming, Z. (2020). Blockchain-based edge computing resource allocation in IoT: A deep reinforcement learning approach. IEEE Internet of Things Journal, 8(4), 2226-2237.

[3] Rathore, S., Pan, Y., & Park, J. H. (2019). BlockDeepNet: A blockchain-based secure deep learning for IoT network. Sustainability, 11(14), 3974.

[4] Kim, H., Park, J., Bennis, M., & Kim, S. L. (2019). Blockchained on-device federated learning. IEEE Communications Letters, 24(6), 1279-1283.

[5] Chai, H., Leng, S., Chen, Y., & Zhang, K. (2020). A hierarchical blockchain-enabled federated learning algorithm for knowledge sharing in internet of vehicles. IEEE Transactions on Intelligent Transportation Systems, 22(7), 3975-3986.

[6] Singh, S., Rathore, S., Alfarraj, O., Tolba, A., & Yoon, B. (2022). A framework for privacy-preservation of IoT healthcare data using Federated Learning and blockchain technology. Future Generation Computer Systems, 129, 380-388.

[7] Maitra, S., Yanambaka, V. P., Puthal, D., Abdelgawad, A., & Yelamarthi, K. (2021). Integration of Internet of Things and blockchain toward portability and low-energy consumption. Transactions on Emerging Telecommunications Technologies, 32(6), e4103.

[8] Moudoud, H., Cherkaoui, S., & Khoukhi, L. (2021, June). Towards a scalable and trustworthy blockchain: Iot use case. In ICC 2021-IEEE International Conference on Communications (pp. 1-6). IEEE.

[9] Latif, S., Idrees, Z., Ahmad, J., Zheng, L., & Zou, Z. (2021). A blockchain-based architecture for secure and trustworthy operations in the industrial Internet of Things. Journal of Industrial Information Integration, 21, 100190.

[10] Prada-Delgado, M. Á., Baturone, I., Dittmann, G., Jelitto, J., & Kind, A. (2020). PUF-derived IoT identities in a zero-knowledge protocol for blockchain. Internet of Things, 9, 100057.

[11] Ardagna, C. A., Asal, R., Damiani, E., El Ioini, N., & Pahl, C. (2019, July). Trustworthy iot: An evidence collection approach based on smart contracts. In 2019 IEEE International Conference on Services Computing (SCC) (pp. 46-50). IEEE.

[12] Malik, S., Dedeoglu, V., Kanhere, S., & Jurdak, R. PrivChain: Provenance and Privacy Preservation in Blockchain enabled Supply Chains. arXiv 2021. arXiv preprint arXiv:2104.13964.

[13] Meisami, S., Beheshti-Atashgah, M., & Aref, M. R. Using Blockchain to Achieve Decentralized Privacy In IoT Healthcare. arXiv 2021. arXiv preprint arXiv:2109.14812.

[14] Tian, H., Ge, X., Wang, J., Li, C., & Pan, H. (2020). Research on distributed blockchain-based privacy-preserving and data security framework in IoT. IET Communications, 14(13), 2038-2047.

[15] He, Q., Xu, Y., Liu, Z., He, J., Sun, Y., & Zhang, R. (2018). A privacy-preserving Internet of Things device management scheme based on blockchain. International Journal of Distributed Sensor Networks, 14(11), 1550147718808750.

[16] **Islam, M., Rehmani, M. H., & Chen, J. (2024).** Differentially private enhanced permissioned blockchain for private data sharing in industrial IoT. Information Sciences, 658, 119997.

[17] Suhail, S., Hussain, R., Jurdak, R., & Hong, C. S. (2021). Trustworthy digital twins in the industrial internet of things with blockchain. IEEE Internet Computing, 26(3), 58-67.

[18] Alrubei, S. M., Ball, E., & Rigelsford, J. M. (2021). The use of blockchain to support distributed AI implementation in IoT systems. IEEE Internet of Things Journal, 9(16), 14790-14802.

[19] Salimitari, M., Joneidi, M., & Chatterjee, M. (2019, December). Ai-enabled blockchain: An outlier-aware consensus protocol for blockchain-based iot networks. In 2019 IEEE Global Communications Conference (GLOBECOM) (pp. 1-6). IEEE.

[20] Wu, X., & Liang, J. (2021). A blockchain-based trust management method for Internet of Things. Pervasive and Mobile Computing, 72, 101330.