Application and architecture for localization in a trustful iot environment with permissioned blockchain support

BASE

In this report is proposed an iot and blockchain architecture to establish a position of a target in trustful environment, without disclosing the information of devices and position data to no authorized user.
This research is based on the paper “A Trust Architecture for Blockchain in IoT” (1).
In that paper is presented a layered architecture for improving the end-to-end trust of sensors data.
The main component of the structure are the devices, the gateways and blockchain. The devices are the means to capture some data from the environment and send them to apposite gateway. The gateways evaluate the trust of devices and insert the data in the blockchain. Between gateways it is evaluated the reputation of gateways. The blockchain maintains secure and untempered the data.
The architecture evaluates the trustworthiness of sensor observations at the data layer and adapts block verification at the blockchain layer through the proposed data trust and gateway reputation modules. Then, proposed system is evaluated in a localization scenario.
The trust layer proposed consists in two key modules: (1) the data trust module; and (2) the gateway reputation module. The data trust module calculates the trust given to an observation from a device. This trust consists in three elements: Confidence of the data source (conf); Evidence from other observations (sens); Reputation of the data source (rep).
The confidence of a device observation is the level of trust a device gives to its observation. In the paper case is the signal power received from a target by a device.
The evidence from other observations is a value that represents the level of support neighbor devices give to an observation of a device. For example, in the paper case, the value can be the similarity of the distances value calculated from other devices with the distance calculated by a specific device.
The reputation of the data source is a value that evolve during the life of system and it depends on behavior of the device. If the device behaves maliciously the reputation decreases, contrary the reputation increases.
The trust of a device is calculated through this formula: Trust = conf * sens * rep.
Instead, the gateway reputation consists in module that tracks the long-term behavior of gateway nodes and adapts the block validation depending on the reputation of the current gateway node. The proposed reputation consist in frequent updates from the blockchain layer, where each node’s honesty in block

mining, B(Gi), is reported based on direct and indirect evidence, and used to update the node's reputation score. The reputation module further integrates the data trust mechanism to the block validation process by validating: (1) the observation trust values assigned by the gateway, and (2) the device transactions reported in the block to update the reputation score of the gateway node. External sources of a node's reputation E x t(Gi), which can be imported from other systems, can also be fed into the node's reputation score. In summary, the reputation score, Rep(Gi) ∊ [Repmin,Repmax], of node Gi is based on a function g : Rep(Gi) = g [T(Gi),B(Gi),E x t(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 the gateway node increases if the generated block is validated, and decreases otherwise.
Then, since the gateway nodes are known by the network and have permissions to generate blocks, they do not need to compete for block generation using computationally expensive block mining mechanisms. This paper proposes a lightweight block generation mechanism, where gateways generate blocks in periodic intervals.
Moreover, there is an adaptive block validation mechanism. Depending on the reputation of the block generating node, each validator randomly validates a percentage of the transactions in the block.
In summary the advantages of this system are a simple and fast block generation procedure, the use of blockchain to have untampered data, an adaptive block validation technique and a system of trust for devices and gateways. The last two let the system to counter possible malicious devices or gateways in a simple and fast way.
Instead, in summary the weakness are a use of a private blockchain that no privatize the identity of the gateways, a no really secure technique to protect the data between devices, targets and gateways and no way to scale the structure to more user, because of every data is public and visible in the blockchain, so there isn't a access control system to separate and privatize the data between more users than one.
In my research I utilized the same scheme of the architecture, so divided in gateways and devices, with same interaction. Then I take the idea to utilize a private blockchain for performance reason and the idea of the trust system for the device case to counter possible malicious devices.

GOAL

The target of my research is to give to the structure presented in the paper "A Trust Architecture for Blockchain in IoT" (1) some feature of privatization of data in the blockchain, with also the aim to show these new characteristics through an example of possible application building a prototype of the end to end process from device to the user in the case localization case.

INTRODUCTION

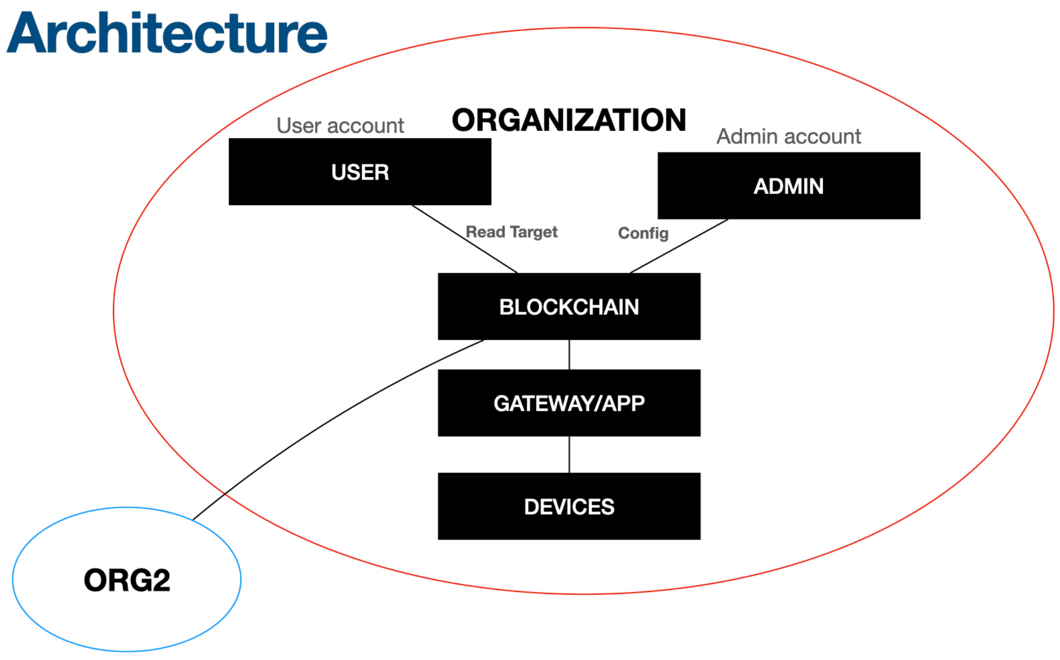
Iot technology is more and more in our life and in the business. You can find them as network chip in home appliances or security camera. You can find it in logistic field to check the products transported or in localization use case, where them can be used to calculate the position of some person indoor for analysis case. For example you can give to some person in a museum a device that communicate with some anchor devices to determinate the path used by the person to visit the museum.
Although this incredible possibility of application, the devices are still connected to a central server that doesn't give an assured security. Instead, the technology of blockchain can give this security simply existing for its property. For this reason, in this project this technology is used to memorize the data from devices. In the blockchain data can't be tempered. In particular in the research Hyperledger Fabric blockchain is utilized. It is a private and permissioned blockchain. It can give the performance and security needed for our system, where many devices send data in nano seconds. It also gives privacy, through the concept of the collection, and a possibility to utilize common programming language to write the business logic of blockchain making developing more flexible. An experiment can be assigned to a collection of data and who doesn't participate to this experiment cannot access to the data. Moreover Fabric can give the possibility to write smart contact (chaincode) in more general programming language, so making the business logic more flexible. The running of the contract happen in the blockchain making it trustful.
For this research, because of lack of found and material, it is used, for devices network, the testbed CLOVES (Communication and Localization Testbed for Validation of Embedded System) of university of Trento. That can be used by anyone after subscribe to the service.
Every this technology is tested and evaluated with general or ad-hoc testing framework. For example Hyperledger Caliper is used to test performance of chaincode (business logic of Hyperledger Fabric) execution.
Eventually, every piece of this project is used to create a prototype, a starting point, for developing a secure architecture to process position data received from devices in a secure and private way. The use case is a client that can register to this system and do his experiment of localization, with other clients, building his devices network and application business logic upon the security, functionality and privacy that are given by the blockchain.
The code of the prototype system, together with every tool to test it, can be find here.
At the end, the report is divided in: 1. Presentation of related works; 2. Presentation of the global system structure; 3. Brief explanation of CLOVES testbed; 4. Brief explanation of used language Golang and the reasons; 5. Explanation of Hyperledger Fabric characteristics, its functionality utilized and the business logic built on it; 6. Explanation of the prototype application running on gateways, admin and user nodes; 7. Explanation in more detail of the security and localization algorithms utilized; 8. Explanation of the tests done; 9. Result of the tests; 10. Conclusion of the report.

RELATED WORK

In these years many papers have been published that talk about the union between iot and blockchain world. In many cases blockchain was implemented also with machine learning technique. In other cases it is considered privacy in blockchain, many of those deal with Hyperledger Fabric. “Blockchain-based edge computing resource allocation in IoT: A deep reinforcement learning approach” (2) gives Asynchronous Advantage Actor-Critic (A3C) approach, to allocate the edge computing resources, which exemplifies how artificial intelligence (AI) can be combined with blockchains. In this paper, it was used the deep reinforcement learning algorithm to do with the ECN (Edge Computing Nodes) selection. “BlockDeepNet: A Blockchain-Based Secure Deep Learning for IoT Network” (3) gives a method to do deep learning in iot network. It introduce Blockchain-based secure DL for IoT network that can support a secure, collaborative DL at the device level and which provides data integrity and confidentiality in the IoT network. Moreover, it introduces a collaborative DL mechanism in the blockchain environment to support local learning models at the device level and aggregation of these local learning models at the edge server through blockchain transactions. It provides the byzantine-resilience for distributed learning in 5G era proposing a secure computing framework based on the sharding-technique of blockchain, namely PIRATE. A similar idea is proposed in “Blockchained On-Device Federated Learning” (4). It proposes a blockchained federated learning (BlockFL) architecture where local learning model updates are exchanged and verified. This enables on-device machine learning without any centralized training data or coordination by utilizing a consensus mechanism in blockchain. The logical structure of BlockFL consists of devices and miners. The miners can physically be either randomly selected devices or separate nodes such as network edges (i.e., base stations in cellular networks), which are relatively free from energy constraints in mining process. The operation of BlockFL is summarized as follows: Each device computes and uploads the local model update to its associated miner in the blockchain network; Miners exchange and verify all the local model updates, and then run the Proof-of-Work (PoW); Once a miner completes the PoW, it generates a block where the verified local model updates are recorded; and finally, the generated block storing the aggregate local model updates is added to a blockchain, also known as distributed ledger, and is downloaded by devices. Each device computes the global model update from the new block. “A Hierarchical Blockchain-Enabled Federated Learning Algorithm for Knowledge Sharing in Internet of Vehicles” (5) proposes hierarchical blockchain framework feasible for the large scale vehicular networks. The hierarchical federated learning algorithm is designed to meet the distributed pattern and privacy requirement of IoVs. Knowledge sharing is then modeled as a trading market process to stimulate sharing behaviours, and the trading process is formulated as a multileader and multi-player game. Simulation results show that the proposed hierarchical algorithm can improve the sharing efficiency and learning quality. Furthermore, the blockchain enabled framework is able to deal with certain malicious attacks effectively. The paper proposes a light-weight blockchain by exploiting a new consensus mechanism named as Proof-of-Learning (PoL). The proposed mechanism combines federated learning with consensus, utilizing the learning quality as the proof of works, which avoids the waste of computing power. It models the update of training models as a trading market. A multi-leaders and multi-followers non-cooperative game is formulated, where training nodes can choose to sell their learning models to servers according to the bidding prices, and servers adjust their bidding according to the quality of received training models. “A framework for privacy-preservation of IoT healthcare data using Federated Learning and blockchain technology” (6) proposes a Blockchain and Federated Learning-enabled architecture for privacy-preservation, improving scalability using blockchain and Federated Learning in the case study and performance analysis for smart healthcare. It provides privacy and security with the help of differential privacy and homomorphic encryption. “Integration of Internet of Things and blockchain toward portability and low-energy consumption” (7) presents an IoT-friendly Blockchain scheme implemented to evaluate the feasibility of medical supply and drug transportation to aid security and mitigate privacy issues which are fault-tolerant, transparent, and traceable. It uses proof of authentication as lightweight consensus algorithm that can be implemented in the IoT environment. In PoAh, miners are called trusted devices and authenticate the source of the block with a unique identifier. The identifier tokens are stored locally on the trusted nodes. The process begins by compiling data generated by network participants into a block. The device compiling the block includes its unique identifier: the Unique Block Token (UBT). The devices in the network disperse the block throughout the network. The trusted node receives the block and checks the identifier of the source. If the token is in the list, the trusted node adds it to its local chain and forwards it along the network. The validated block is then added by the rest of the devices in the network. “Towards a Scalable and Trustworthy Blockchain: IoT Use Case” (8) proposes a scalable and trustworthy blockchain (STB) architecture that is suitable for the IoT; which uses blockchain sharding and oracles to establish trust among unreliable IoT devices in a fully distributed and trustworthy manner. In particular, it gives a Peer-To-Peer oracle network that ensures data reliability, scalability, flexibility, and trustworthiness. Furthermore, it introduces a new lightweight consensus algorithm that scales the blockchain dramatically while ensuring the interoperability among participants of the blockchain. “A Blockchain-based Architecture for Secure and Trustworthy Operations in the Industrial Internet of Things” (9) realizes the great potential of integrating blockchains with the IoT in a manner suitable for smart industrial environments. This paper proposes a lightweight, easily expandable, and decentralized private blockchain-based IIoT network. The proposed architecture is used to perform several industrial operations, including user and device registration, sensor and actuator data storage, and client service tasks, in a trustworthy manner. The proposed blockchain mechanism is built on a private network that provides access to authorized users only. “PUF-derived IoT identities in a zero-knowledge protocol for blockchain” (10) proposes an alternative authentication approach in which an MCU generates a secret key internally is introduced, exploiting manufacturing variability as a physical unclonable function (PUF). As the key is generated by the device itself, manufacturers save the expense of a secure environment for external key generation. In production, once chips are loaded with a firmware, it is only necessary to run an internal characterization and pass on the resulting public key, mask and helper data to be stored for authentication and recovery. “Trustworthy IoT: An Evidence Collection Approach based on Smart Contracts”(11) proposes a service based methodology based on blockchain and smart contracts for trustworthy evidence collection at the basis of a trustworthy IoT assurance evaluation. The methodology balances the provided level of trustworthiness and its performance, and is experimentally evaluated using Hyperledger fabric blockchain. It is based on blockchain and smart contract to guarantee collection of reliable evidence whose integrity is proven over time. Differently from existing solutions, its methodology links the evidence to the way in which it is collected and aggregated, or a decision based on it is taken. It also introduces technique for privacy as trustworthy data aggregation that substantially increases privacy, since it permits to hide details about the single data points, while keeping the possibility to track back data to its origin. Function data aggregation DA: $Dp \times op \rightarrow A$ takes as input a set of data points Dp={dp1 , dp2 , . . . , dpn}, the aggregation operator op∈{sum, average, min, max, count}, and produces as output the aggregated evidence A=op(Dp) that is stored in blockchain. “PrivChain: Provenance and Privacy Preservation in Blockchain enabled Supply Chains”(12) presents a blockchain, PrivChain, that uses Zero-Knowledge Succinct Non-Interactive Argument of Knowledge (ZK-SNARKs), an efficient non-interactive variant of ZKPs. Instead of sharing this data, participants share proofs of their valid data pertaining to products. The verification of such proofs is then automated by a blockchain smart contract. A monetary incentive mechanism is proposed to reward participants for generating successfully verified proofs. “Using Blockchain to Achieve Decentralized Privacy In IoT Healthcare” (13) introduces the idea to store the pointers to the data (hash of encrypted data) on blockchain to lighten the storage space of blockchain. The encrypted data are stored on off-chain storage. For the implementation of off-chain storage, it is used The InterPlanetary File System (IPFS). To satisfy the security of data, it uses a symmetric key encryption scheme. It introduces Fine-grained Access Control where each patient can grant a set of permissions to any desired member of the medical staff for accessing a patient's healthcare data. Also, the patient can alter or revoke the set of granted access permissions. These permissions are securely stored on blockchain ledger as access-control policies, where only the patient can change or revoke them. Then, it assures data transparency and auditability. Patients have complete and accurate transparency over their collected healthcare data, and they can know how medical staff can access to healthcare data. Digital signature is added to the data for authentication purposes. For the implementation of the digital signature, it is used ECDSA. The patient also generates a secret key for encrypting data with an AES symmetric encryption algorithm. It is denoted the data access permissions by policies, which indicates the permissions that the patient gives to the selected member of the medical staff so that she can access a particular type or all of the patient's data. “Research on distributed blockchain-based privacy-preserving and data security framework in IoT” (14) adopts gateway devices to store data and build a P2P network for achieving information interaction. At the same time, the middleware server is used to process data and realize access control management. By adopting blockchain, it provides effective security and privacy protection for IoT. All data are encrypted and stored in gateway nodes, and can only be used by specific entities after verification. It is adopted hash algorithm to process data through integration, cleaning, conversion and other measures, to eliminate the multi-source heterogeneity of IoT data, and truly realize the interconnection of devices. It is used Hyperledger-Fabric to ensure data privacy by creating dedicated channels and encrypting the data on the chain. “A privacy-preserving Internet of Things device management scheme based on blockchain” (15) combines attribute-based encryption (ABE) and time-bound key management techniques to achieve privacy-preserving IoT device management–based blockchain. ABE is a public-key encryption algorithm in which user keys or ciphertexts are related to attributes. In this cryptosystem, the ciphertext can only be decrypted if the attributes corresponding to the user key satisfy the attributes corresponding to the ciphertext. “Differentially private enhanced permissioned blockchain for private data sharing in industrial IoT” (16) utilizes differential privacy. With that the removing or adding of a single data record from a dataset does not significantly change the output of an algorithm (applied to the dataset) under the constraint of differential privacy. To achieve differential privacy, a calibrated noise is added to the actual data. The amount of noise is controlled by the parameter ϵ which is known as differential privacy budget. Furthermore, the adversary model is stronger than other privacy preservation techniques i.e., the adversary has all other information except his target person/data record. However, in case of multiple queries on the same dataset, the privacy protection degrades due to accumulation of privacy budget which results in high value of ϵ. Similarly, the adversary can send repeated queries to the same dataset and take the average to predict the actual data. Moreover, the number of queries which the data holder can answer is bounded by ϵ. It amalgamates differential privacy into the chaincode (smart contract) of HF which first accesses the ledger’s data and then a calibrated noise is added to the actual data before sharing it with the requester which guarantees privacy preservation through ϵ-differential privacy. “Trustworthy Digital Twins in the Industrial Internet of Things with Blockchain” (17) introduces the promising solution to create a digital fingerprint (or virtual replica) of the underlying product, process, or service where all operations must be analyzed, predicted, and optimized before its real-world implementation. Following a closed-loop, the simulation data are fed back to the physical system to calibrate the operations and to enhance the system performance. Such a bi-directional mapping between physical space and virtual space is called Digital Twin (DT). “The Use of Blockchain to Support Distributed AI Implementation in IoT Systems” (16) has as contributions: a novel, and secure blockchain architecture for supporting DAI on low-power and low-cost IoT devices.; Practical implementation of DAI using scalable and distributed IoT hardware platform; Prediction and measurements of DAI using blockchain in IoT devices with a performance analysis that includes, accuracy, energy consumption, and overall system latency utilizing data from trusted and robust platforms; A blockchain protocol that includes a new consensus mechanism, and transaction and block formats that help nodes handle DAI-based transactions and prediction requests. “AI-enabled Blockchain: An Outlier-aware Consensus Protocol for Blockchain-based IoT Networks” (18) proposes the AI-enabled blockchain (AIBC) network with a 2-step consensus protocol using an outlier detection algorithm as the first step and PBFT as the second one. This paper proposes an architecture that combines the strengths provided by edge computing, AI, and blockchain technologies to provide robust, secure, and intelligent solutions for secure and faster data processing and sharing. To secure the proposed architecture a new

concept for consensus mechanism based on Honesty-Based Distributed Proof of Work (DPOW) were devised. It is suitable for the implementation in public blockchain-IoT applications and takes advantage of the IoT devices' collective hash powers to share the work of mining a block. The presence of edge nodes helps in accommodating the computation power and storage that blockchain requires. "A blockchain-based trust management method for Internet of Things"(19) proposes a blockchain-based trust management mechanism (BBTM), where the trustworthiness of sensor nodes will be evaluated by the mobile edge nodes. BBTM can create smart contracts for trust computation and verify the computation process.
In many of these papers there is the concept to compute the trust of devices in edge node. For example in my project I utilize the same strategy of (19) with Hyperledger Fabric.
We can notice that (11), (12), (14) and (18) utilize hyperledger fabric. In most of them the fabric is used for a permissioned blockchain but don't use the private characteristic. In (14) is written to use different channel of fabric blockchain to privatize the data. In my report I want to present and utilize the concept of collection in fabric blockchain. It lets the architecture to be more flexible and less expensive in term of memory and cost. Instead of deploy a new channel, that creates a new ledger, with collection the data are separated logically in the same ledger.

ARCHITECTURE



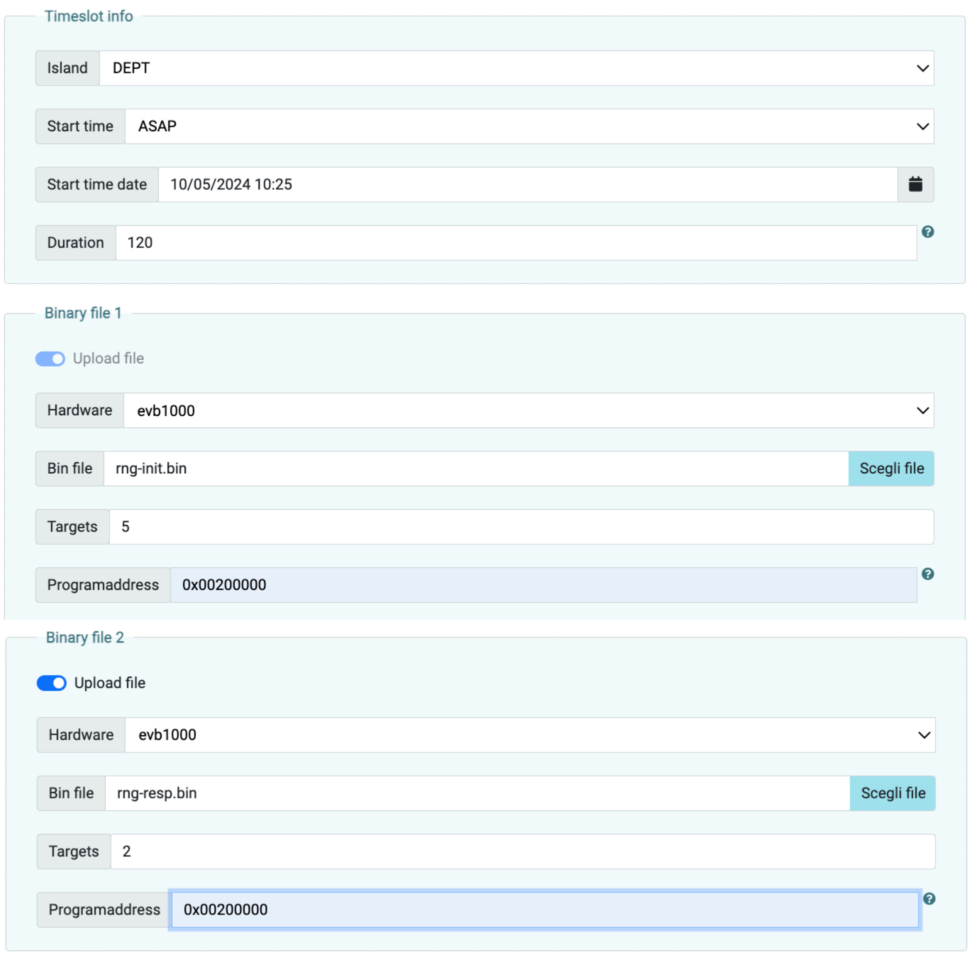
The structure of the system is composed of five elements:

1. Devices: this component represents the network of iots devices that capture exterior information. In project case they compute ranging process with a specific target. In the prototype this part is represented trough the testbed CLOVES. There is also a process of encryption on data from devices. It is simulated apart the ranging process.
2. Gateway/App: this component represents the nodes the devices connects to send the data. In these nodes run an application with admin permission. This application does all the communication operations with blockchain and its chaincode to update and create the devices entity, and calculate and memorize the position of the target in the blockchain.
3. Blockchain: It is implemented through Hyperledger Fabric. It memorized all data concern to devices and target. The data are divided in collections associated to an organization. In simple words every admin can create an own organization, a logical element in fabric, and associate to this organization some collections that contains a target and some devices. In this manner only who belongs to this organization can access that data. These organizations and collections can create some sort of access control and privatization. In the blockchain it is also deployed the chaincode that is needed by the gateway application for calculate target potion, by the user to access target and by admin to maintain the data and structure of the system.
4. User: It is an entity with an account of user in a specific organization. it can only access to position of target through a User application. This account can be given by an organization admin.
5. Admin: It is an entity with an admin account in the specific organization. It can change some property of devices in the blockchain, read devices data, read target data and create new device entity in the blockchain through an Admin application. The person that have this account can be seen also the same person that build the network of devices and start the application in the gateway to do his experiment in the trustful environment.

In the case of prototype, the component of blockchain is simulated in docker containers. For the project it is utilized a test network that is given by the hyperleadger fabric foundation. The Admin, User and Gateway application are presented in the prototype as Golang program. The devices network are simulated through a physical testbed. The applications and blockchain can be simulated and tested together in the prototype. Instead, the network of devices, so the rag is simulated apart. The process of ranging and encryption can’t be connected to the other part of the architecture in the prototype. In the prototype it is tried to simulate the gateway application receiving data from drives only considering the time.

TESTBED

The testbed utilized for test the iot network is CLOVES. CLOVES is a large-scale public testbed, the first available for ultra-wideband (UWB). Narrowband IEEE 802.15.4 devices are also deployed alongside UWB ones. All devices are remotely accessible for firmware upload and collection of experiment logs. The different characteristics of the indoor testbed areas (e.g., narrow corridors vs. wide halls) cater to different types of studies concerning communication and localization. CLOVES is part of the iot testbed at the Department of Information Engineering and Computer Science at the University of Trento. For the project it is used the device type evb1000. Some devices in Cloves network are used as anchor and other device as target for the localization. To create the files utilized by CLOVES, the source codes in C for anchor and target are compiled through COOJA simulator. The obtained bins files (the firmware for anchor and target) are passed to CLOVES (fig. 1).


Timeslot info

Island	DEPT
Start time	ASAP
Start time date	10/05/2024 10:25
Duration	120

Binary file 1

Upload file

Hardware	evb1000
Bin file	rng-init.bin
Targets	5
Programaddress	0x00200000

Binary file 2

Upload file

Hardware	evb1000
Bin file	rng-resp.bin
Targets	2
Programaddress	0x00200000

fig. 1

It can be find more general information on the process on this link.
The idea is that the anchors in turn do some two way ranging (fig. 2) with the target.
Every time the distance is calculated the device writes in a log. The most important data in the log are extracted through a parse program. These data are: the id of the device, the id of the target, the calculated distance in millimeter and the confidence that device gives to the distance. The confidence is based on RSSI (Received Signal Strength Indicator) from the target. The idea come from the base paper “A Trust Architecture for Blockchain in IoT” (1). If the signal power from target is upper than certain value then device gives maximum confidence. If the signal power from target is lesser than certain value then device gives minimum confidence. If it isn’t one of this case the confidence is calculated trough a formula based on RSSI received. The concept of these choices is that if the signal isn’t

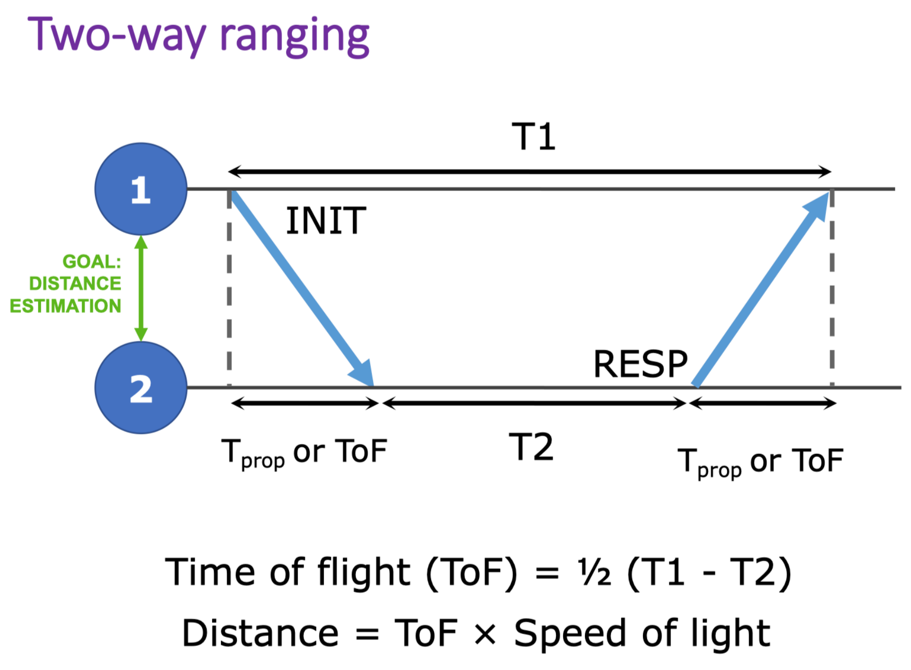

Two-way ranging
T1
1
INIT
GOAL: DISTANCE ESTIMATION
2
RESP
T_{prop} or ToF
T2
T_{prop} or ToF
Time of flight (ToF) = ½ (T1 - T2)
Distance = ToF × Speed of light

Fig.2

strong enough then the measurement by the device can be less precise because of the large distance or obstacle between target and device.
After parsing, the parsed data are encrypted through xor encryption. It can be thought that the key of encryption is a symmetric key and it is stored in device and also in the device entity in the blockchain. Before the encryption the data are hash ed and the digest is sent with the correspondent encrypted data. The idea is that the encryption gives confidentiality and hashing gives integrity. It was choose xor encryption for it simplicity and performance. For authenticity, on the application, when the gateway analyze the data received, it is checked that id of the data received correspond with the id of device entity in the blockchain. It is taken this choice for simplicity and performance too.
In the special case of the prototype the three phases ranging, parsing and encryption are separated, sequential and connected with files, that is a phase utilize the file generated from previous phase. In future projects, that might base on this research, may connect the phases together to have a more realistic prototype.

GOLANG

Go is a statically typed, compiled high-level programming language designed at Google by Robert Griesemer, Rob Pike, and Ken Thompson. It is syntactically similar to C, but also has memory safety, garbage collection, structural typing, and CSP-style concurrency. In this project it is used the Golang libraries implementation that are given by the Hyperledger foundation package page. In specific, Golang is used in chaincode and application implementation.
It is chosen for its semplicity, efficency, flexibility, usability and cross-platform propriety.

BLOCKCHAIN

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. Fabric is the first distributed ledger platform to support smart contracts 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. 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. First, 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. 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. 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. A smart contract, or what Fabric calls "chaincode", functions as a trusted distributed application that gains its security/trust from the blockchain and the underlying consensus among the peers. Hyperledger Fabric, being a permissioned platform, enables confidentiality through its channel architecture and private data feature. In channels, participants on a Fabric network establish a sub-network where every member has visibility to a particular set of transactions. Thus, only those nodes that participate in a channel have access to the smart contract (chaincode) and data transacted, preserving the privacy and confidentiality of both. Private data allows collections between members on a channel, allowing much of the same protection as channels without the maintenance overhead of creating and maintaining a separate channel. At a high level, Fabric is comprised of the

following modular components: peers node that maintain the leader and propose transaction; ordering service nodes 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; membership service provider is responsible for associating entities in the network with cryptographic identities. For example can associate an entity to a user client and a admin client; peer-to-peer gossip service disseminates the blocks output by ordering service to other peers; Smart contracts ("chaincode") run within a container environment (e.g. Docker) for isolation; Endorsement and validation policy enforcement that can be independently configured per application; A service for the creation of cryptographic material of various entity in the network (as CA), 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.

2. Privacy: Through the Membership Service Provider (MSP) can be defined an organization, that consists in a logical set of peers. In cases where a group of organizations on a channel need to keep data private from other organizations on that channel, they have the option to create a new channel comprising just the organizations who need access to the data. However, creating separate channels in each of these cases creates additional administrative overhead (maintaining chaincode versions, policies, MSPs, etc), and doesn't allow for use cases in which you want all channel participants to see a transaction while keeping a portion of the data private. That's why Fabric offers the ability to create private data collections, which allow a defined subset of organizations on a channel the ability to endorse, commit, or query private data without having to create a separate channel. A collection is the combination of two elements: The actual private data, sent peer-to-peer via gossip protocol to only the organization(s) authorized to see it. This data is stored in a private state database on the peers of authorized organizations, which can be accessed from chaincode on these authorized peers. The ordering service is not involved here and does not see the private data; A hash of that data, which is endorsed, ordered, and written to the ledgers of every peer on the channel. The hash serves as evidence of the transaction and is used for state validation and can be used for audit purposes.
   This two components let data to be private but at the same time the data are endorsed and validated from all endorsement and order nodes.
3. Test network (prototype network): Hyperledger gives to developer a test network that contains all element for hyperleader fabric with some example of application. In these research this network is customized to meet project necessities. In the point of view of physical characteristic (fig. 3) there are the following components: 4 Order Nodes that create a block through PBFT consensus algorithm. This algorithm can give some countermeasures against possible malicious entity in the network; 1 peer node for the endorsement, proposing of transactions and maintaining of collections in organization 1; 1 peer node for the endorsement, proposing of transactions and maintaining of collections in organization 2. In prototype isn't utilized a service for the creation of cryptographic material but a command named "cryptogen".
   These components are simulated through docker containers.

PHISYCAL: TEST NETWORK

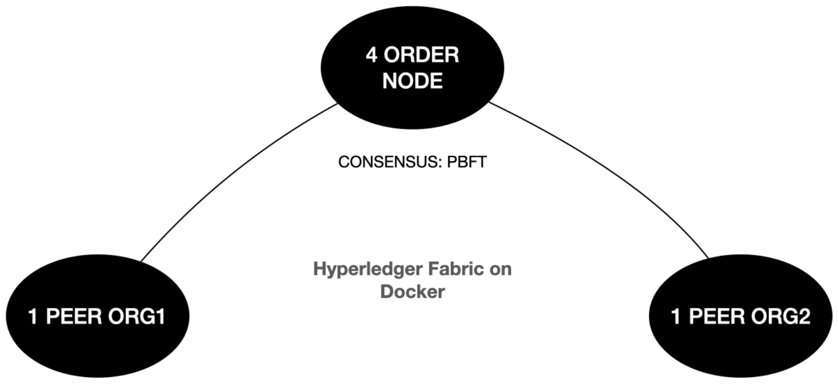


fig. 3

LOGICAL: TEST NETWORK

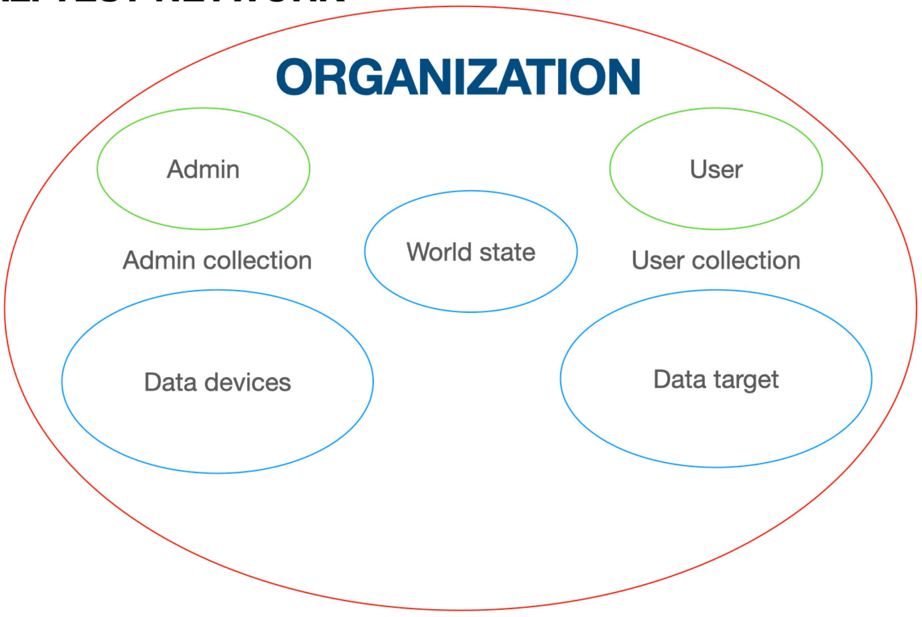


fig. 4

The logical aspect of test network is composed of the following components: 2 organization (fig. 4); 1 admin client credential for every organization; 1 user client credential for every organization; 1 world state of the blockchain for all organization; 2 collections for every organization. The collection of an organization can be accessed only by a client or peer of that organization.
The Admin collection contains the devices data and can be accessed only by Admin entities (client, application or peer) of the same organization. A device entity in the collection is composed of the following field: id of the device; key for the decryption of data from the device; X coordinate of the device (second of the network map); Y coordinate of the device; distance observed by the

device; confidence of the device on its observation; evidence, value that represents the support to the device observation from neighbor devices; reputation of the device; trust on the device observation; id of the devices near the device.
The User collection contains the target data and can be accessed by User and Admin entities (client, application or peer) of the same organization, but can be only modified by Admin entities. The target entity in the collection is composed of the following fields: id of the target; X coordinate of the target (second of the network map); Y coordinate of the target; timestamp of the position target update; a value to indicate if the target is updated or not.

4. Chaincode: The business logic of the blockchain is written in Golang for its efficiency, usability and simplicity properties. In this chaincode is defined only a smart contract "PositionContract" that is used from all organization. Some data passed to the contract methods are in a transient state in order to not show these data in the blockchain transaction correspondent to that contract call. Simply, the transient option let to have confidentiality on the transaction data. All in all, this contract gives the following functionalities:
   - CreateDevice(ctx contractapi.TransactionContextInterface, collection string): this method let an admin to create a device in the same organization. The data passed by admin used for the device creation as transient data are the id, the x coordinate, the y coordinate, the key for the decryption, the neighborhood of the device and the initial reputation of device. Then, as normal argument it is passed the name of the collection to insert the device. During the execution of the method, firstly it is checked that the collection in the argument is a device collection. Secondly, it is checked the caller is an admin. Then, it is checked the correctness of the data for the initialization. For example the neighbors must be more than 1. After, it is checked that a device with that specific id doesn't exist yet. Every device data that are not passed by admin are initialize to 0 value. At the end, the device is created with the initialization data.
   - UpdateDevice(ctx contractapi.TransactionContextInterface, collection string): this method let an admin to change some device parameters. The data passed by admin used for the device updating as transient data are id of device, the new decryption key and the new neighborhood. Then, as normal argument it is passed the name of the collection where is the device to update. During the execution of the method, firstly it is checked that the collection in the argument is a device collection. Secondly, it is checked that the caller is an admin. Then, it is checked the correctness of the data for the updating. For example the neighbors must be more than 1. After, it is checked that a device with that specific id must already exist in blockchain. Every device data that are not passed by admin are initialize to the value retrieve from existent device with that id. At the end, the device is updated.
   - CreateTarget(ctx contractapi.TransactionContextInterface, collection string): this method let an admin to create a target in the same organization. The data passed by admin used for the target creation as transient data is the id. Then, as normal argument it is passed pass the name of the collection to insert the target. During the execution of the method, firstly it is checked that the collection in the argument is a target collection. Secondly, it is

checked the caller is an admin. Then, it is checked the correctness of the data for the initialization. After, it is checked that a target with that specific id doesn't exist yet. Every device data that are not passed by admin are initialize to 0 value except the variable that represent the updating status that is initialize to false. At the end, the target is created with the initialization data.

- UpdateDeviceObsConf(ctx contractapi.TransactionContextInterface, collection string): this method let an admin to update a device confidence and observation. The data passed by admin used for the device updating as transient data are id of device, the distance calculated, the confidence on the device observation, the minimum confidence and the maximum confidence. Then, as normal argument it is passed the name of the collection to find the device to update. During the execution of the method, firstly it is checked that the collection in the argument is a device collection. Secondly, it is checked the caller is an admin. Then, it is checked the correctness of the data for the updating. For instance that the confidence is between the minimum and the maximum. After, it is checked that a device with that specific id must already exist. Every device data that are not passed by admin are initialized to the value retrieve from existent device with that id. At the end, the device is updated.
- UpdateDeviceEvRep(ctx contractapi.TransactionContextInterface, collection string): this method let an admin to update a device observation evidence and trust, and device reputation. The data passed by admin used for the device updating as transient data are: id of device; reward or penalty in case of high evidence, and high confidence and in case of low evidence and high confidence respectively; reward or penalty in case of high evidence and low confidence, and in case of low evidence and low confidence respectively; threshold that indicate when an evidence value is high or low; maximum reputation. Then, as normal argument it is passed the name of the collection to find the device to update. During the execution of the method, firstly it is checked that the collection in the argument is a device collection. Secondly, it is checked the caller is an admin. Then, it is checked the correctness of the data for the updating. For instance that the maximum confidence must be positive. After, it is checked that a device with that specific id exists. Then, it is calculated the evidence and trust of observation, and reputation of the device (see Algorithm section for more detail). Every device data that are not passed by admin or calculated are initialized to the value retrieve from existent device with that id. At the end, the device is updated.
- DeleteDevice(ctx contractapi.TransactionContextInterface, collection string): this method let an admin to delete a specific device. The data passed by admin used for the device cancelation as transient data is the id. Then, as normal argument it is passed the name of the collection to find the device to delete. During the execution of the method, firstly it is checked that the collection in the argument is a device collection. Secondly, it is checked the caller is an admin. Then, it is checked the correctness of the data for the cancelation. After, it is checked that a device with that specific id must already exist. At the end, the device is deleted.

- DeleteTarget(ctx contractapi.TransactionContextInterface, collection string): this method let an admin to delete a specific target. The data passed by admin used for the target cancelation as transient data is the id. Then, as normal argument it is passed the name of the collection to find the target to delete. During the execution of the method, firstly it is checked that the collection in the argument is a target collection. Secondly, it is checked the caller is an admin. Then, it is checked the correctness of the data for the cancelation. After, it is checked that a target with that specific id must already exist. At the end, the target is deleted
- PositionTarget(ctx contractapi.TransactionContextInterface, collectionD string, collectionT string, dateU string): this method let an admin to calculate the position of the target. The data passed by admin used for the position calculation as transient data are id of target, maximum error tolerable and list of all device id to consider for the calculation. Then, as normal argument are passed the name of the collection to find devices, the name of the collection to find the target and the timestamp. During the execution of the method, firstly it is checked that the first collection is a device collection and second collection is a target collection. Secondly, it is checked the caller is an admin. Then, it is checked the correctness of the data for the updating. After, it is checked that a target with that specific id must already exist. Next operation is retrieving the three devices with greater trust. The distance calculated by these devices is used to calculate the position (see Algorithm section for more detail). Every devices data that are not passed by admin or calculated are initialized to the value retrieve from existent device with that id. At the end, the target is updated with calculated position, timestamp and it is marked as updated.
- ReadTarget(ctx contractapi.TransactionContextInterface, collection string): this method let an admin and a user to read target data. As transient data is passed the id of target to read. Then, as normal argument it is passed the name of the collection where read the target. During the execution of the method, firstly it is checked that the collection in the argument is a target collection. Then, it is checked the correctness of transient data. At the end, the target data with that id is retrieved and returned by the method if it exists; ReadAllIDDevices(ctx contractapi.TransactionContextInterface, collection string): this method let an admin to read all devices id. As normal argument it is passed the name of the collection where read the devices. During the execution of the method, firstly it is checked that the collection in the argument is a device collection. Secondly, it is checked the caller is an admin. At the end, it is retrieved and returned a list of ids of all devices in the collection.
- ReadDevice(ctx contractapi.TransactionContextInterface, collection string): this method let an admin to read some specific device data. As transient data is passed the id of device to read. Then, as normal argument it is passed the name of the collection where read the device. During the execution of the method, firstly it is checked that the collection in the argument is a device collection. Secondly, it is checked the caller is an admin. Then, it is checked the correctness of transient data. At the end, the device data with that id is retrieved and returned by the method if it exists.

APPLICATION

-a Gateway application: This prototype application runs on a gateway and calculate the position of the target with the distances received from devices associated, through the contract “PositionContract". The language used is the Golang, for its simplicity and efficiency. All functionalities that are presented in the application must be run with admin account. For this reason it is created a grpc connection between the application with the blockchain as admin:

- Global: in the global part of the application there are the definition of a device entity, target entity, type for create a device and global parameters to customize the application for special case and global variables for different type of account. Device entity is defined with id of the device, key for the decryption of device data, x coordinate of the device in a predefined map, y coordinate of the device in a predefined map, target distance observed from the device, confidence on device observation, level of evidence that support the device, reputation of the device, trust given to the device observation and ids of the devices near to the device. Target entity is defined with Id of the target, x coordinate of the target in predefined map, y coordinate of the target in predefined map, timestamp of the update and a variable that indicate if the target is updated. The type for device initialization is defined with id of the device, key for the decryption of device data, x coordinate of the device in predefined map, y coordinate of the device in predefined map, ids of the devices near to the device and initial reputation of the device. As global variables there are: minimum confidence for a device; maximum confidence for a device; reward or penalty in case of high evidence and high confidence, and in case of low evidence and high confidence respectively; reward or penalty in case of high evidence and low confidence, and in case of low evidence and low confidence respectively; threshold that indicate when a confidence value is high or low; threshold that indicate when an evidence value is high or low; collection where are memorized the devices; collection where are memorized the targets; maximum error for the position calculation; maximum reputation; dimension of a batch of observations to read before compute the observations average. At the end, there are two variables that indicate the number of observation retrieve currently and the number of successful current evidence, reputation and trust respectively. This two numbers are used to take note of the updates number in the respective case so the main process can wait until the number have reached the value expected.
- Main: The main shows process that must be execute by the application. Firstly it connects with the blockchain. Secondly, initialize the network with the first devices and target. Then, it starts the loop. In this cycle the application check what are the devices in the system. One at the time it is started the process to read the data stream from the device. Then, for every devices it is run the function for updating the observation in devices entity. When this part ends for every device starts the updating of evidence and trust of observations, and

device reputation. When all devices are updated, begin the position calculation. At the end, the loop restart. If there are some error in updating, the main blocks until the problem isn't fixed. Except, if the position calculation gives some problems the target simply isn't updated because maybe there isn't enough precise measures. Moreover, for test purpose the prototype main calculates also the time interval of every loop.

- Functionalies:
  - initLedger(contract *client.Contract, devices []deviceTransientInt, idTarget string): this method initializes the ledger with initial devices and target data.
  - readStreamDevice(idDevice string, c chan string): In prototype case this function simulate the reading of stream of observations receive from the device. It reads these data from specific device file. This function run in parallel with the rest of program and insert the data in a channel variable that will be read from the updating observation process. If there is some error in the reading then it is restarted the read until all file is read.
  - insertObs(contract *client.Contract, idDevice string, idTarget string, c chan string): this method updates the confidence and distance of the observation for a specific device, indicating the maximum and minimum confidence. It is executed in a go routine to run in parallel with other devices updating. To do this it retrieves form blockchain the key for the decryption. Then, it is started the cycle for read the channel variable where the data encrypted are inserted. After decryption, it is computed the hash of this data and the digest is compared with the one retrieved from the blockchain. If there is a correspondence than the data aren't tempered. Then, it is checked that the data target and device ids are correct to assure the authenticity of the data. If all this condition are satisfied then the distances and the confidences are accumulated until it is reached a number of observation. If there is some errors in the execution the function restart by reading the key of decryption, because maybe there is some unexpected data received. It isn't updated the devices for every observation because of the difference of velocity between interval data arrival and processing of application. Then, for stabilize a bit the situation, the data are buffered and the devices are update with an average distance and confidence. Depend on application the tradeoff between accuracy and performance can be decided through a global parameter, the dimension of the batch of observations to read before compute the average. The dimension must be a pair integer because of the data received from devices arrive in couple: encrypted data and digest.
  - upDateDevices(contract *client.Contract, devices []string): this method updates the evidence and trust of the observations, and the reputation of devices. Moreover in the updating it is specified the global parameters of reputation reward and penalty and the threshold that indicates when an evidence value is high or low and maximum reputation. Every device is updated in a go routine in parallel and the upDateDevices finish to run only when all go routine finish. If there is some error in the go routine this

    one restart, because maybe there is conflict with other processes to access the blockchain.
    - upDateTarget(contract *client.Contract, idTarget string, devices []string): this method update the position of the target utilizing all observation information of devices indicated. For the updating is passed also the global parameter that represent the maximum measure error. If there is an error simply the target isn't updated, because maybe the measure error is too high.
    - seeAllIDDevice(contract *client.Contract): this method return all ids of devices in the collection. It is useful for the main process.

-b Admin application: this prototype application gives to an admin client the possibility to read device, read target, update device and create device in own organization through the contract "PositionContract". The language utilized is in Golang for its simplicity and efficiency. To run it is created a grpc connection between the application with the blockchain as admin.

- Global: in the global part of the application there are the definition of a type for the initialization of the device, target entity, type for the device update, device entity and global parameters. The type for device initialization is defined with id of the device, key for the decryption of device data, x coordinate of the device in predefined map, y coordinate of the device in predefined map, ids of the devices near to the device and initial reputation of the device. Target entity is defined with Id of the target, x coordinate of the target in predefined map, y coordinate of the target in predefined map, timestamp of the update and a variable that indicate if the target is updated. The type for the update of the device is defined with id of the device to update, the new key for decryption, and the new neighborhood. Device entity is defined with id of the device, key for the decryption of device data, x coordinate of the device in predefined map, y coordinate of the device in predefined map, target distance observed from the device, confidence on observation of the device, level of evidence that support the device observation, reputation of the device, trust given to the device observation and ids of the devices near to the device. As global parameters there are the name of the collection of the target and devices, and all info to connect to blockchain as admin.
- Main: Firstly it connects with the blockchain. Then, it is requested to the admin what operation wants to execute: read device, read target, update device and create device in own organization. After the choice the admin must insert the informations requested for the operation. When the operation finish the main process asks new choice to the admin.
- Functionalities:
    - insertDevice(contract *client.Contract, device deviceTransientInt): this method lets an admin to add the new device inidicated in devices collection.
    - DeleteDevice(contract *client.Contract, idDevice string): this method lets an admin to delete the indicated device in devices collection.
    - seeDataDevice(contract *client.Contract, idDevice string): this method lets an admin to read the device indicated from devices collection.
    - updateDevice(contract *client.Contract, device deviceTransientUp): this method lets an admin to update the device and data indicated.

    - seeDataTarget(contract *client.Contract, idTarget string): this method lets an admin to read the target indicated from target collection.

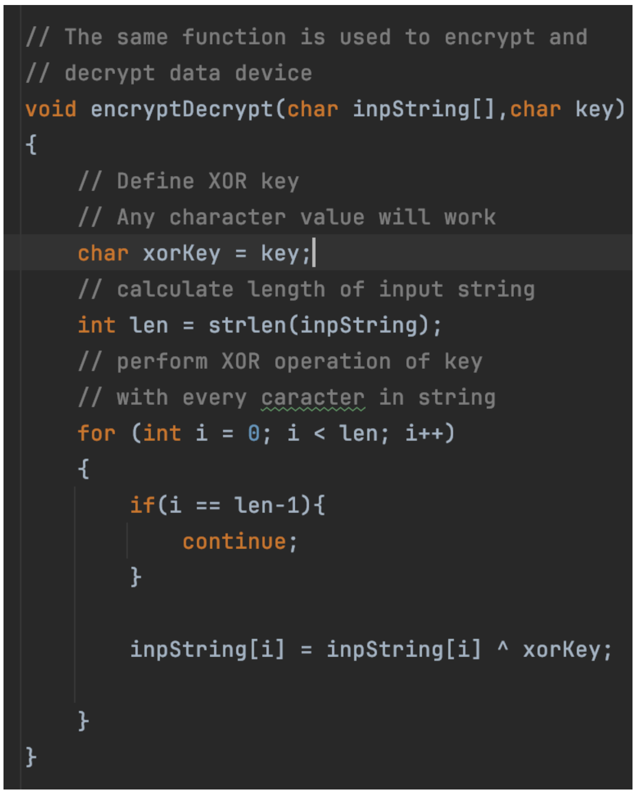
-c User application: this prototype application gives to a user client the possibility to read the target through the contract “PositionContract”. The language utilized is the Golang for its simplicity and efficiency. To run the application it is created a grpc connection between the application with the blockchain as user.

- Global: in the global part of the application there are the definition of type for the reading of the target data that is composed of id of the target, x coordinate of the target in predefined map, y coordinate of the target in predefined map, timestamp of the update and variable that indicate if target is updated. As global parameters there are the name of the collection of the target and all info to connect to blockchain as user.
- Main: Firstly it connects with the blockchain. Then, it is requested to the user if he wants to terminate the process or read the target. If it is chosen the second, the user must insert the id of the target. When the reading finishes the main process ask a new choice to the user.
- Functionalities:
    - seeDataTarget(contract *client.Contract, idTarget string): this method lets a user to read the target indicated from target collection. If the target isn’t updated then the function gives an error.

ALGORITHM

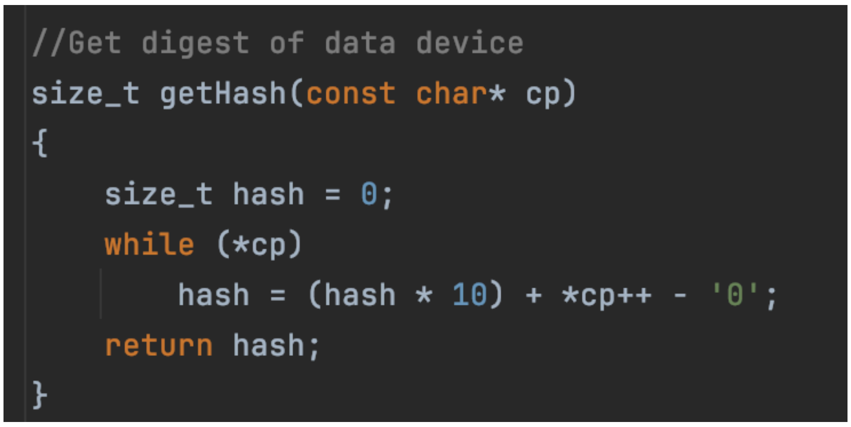
In this section it is explained in more detail certain operations executed in the system:

- Encryption of the observation from device (fig. 5): This algorithm consist in apply the xor operation between the character key and every character of the string in order to encrypt it. The xor operator has the characteristic that let to have the same flow of instructions also for the decryption. It makes this algorithm very simply and fast.
- Hashing of the observation from device (fig 6): the hash function is applied on the observation from device before encryption and the digest is sent with encrypted data to the apposite gateway.
- Two way ranging (fig. 2).
- Calculation of the confidence (fig. 7): The confidence is calculated basing on the RSSI (Received Signal Strength Indicator) from the target. The idea come from the base paper “A Trust Architecture for Blockchain in IoT” (1). If the signal power from target is upper then certain value RSSIUP than device gives maximum confidence 1. If the signal power from target is lesser then certain value RSSIUP than device gives minimum confidence 0.4. If it isn’t one of thiese cases, the confidence is calculated trough a formula 9/4 + rssval/40, where rssval is the RSSI calculated. The concept of these choices is that if the signal isn’t strong enough, then the measurement by the device can be less precise because of the large distance or obstacle between target and device.
- Calculation of the evidence of a device observation (fig. 8): The idea come from the base paper “A Trust Architecture for Blockchain in IoT” (1). It is used the



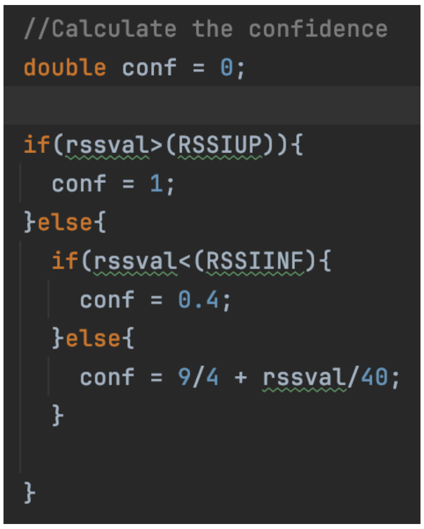
```
// The same function is used to encrypt and
// decrypt data device
void encryptDecrypt(char inpString[],char key)
{
    // Define XOR key
    // Any character value will work
    char xorKey = key;
    // calculate length of input string
    int len = strlen(inpString);
    // perform XOR operation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

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

    }
}
```

fig. 5



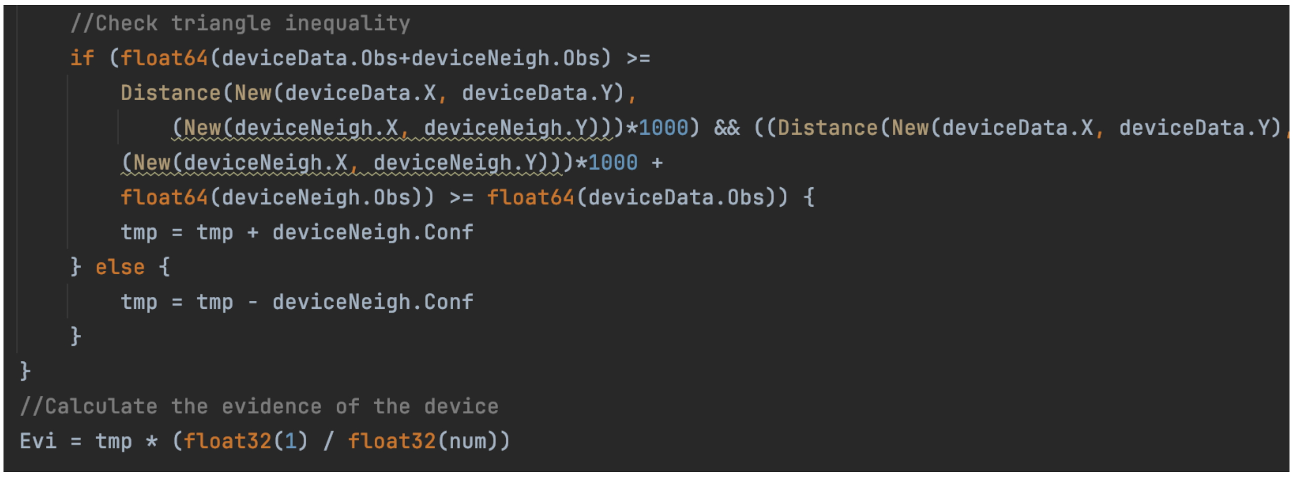
```
//Get digest of data device
size_t getHash(const char* cp)
{
    size_t hash = 0;
    while (*cp)
        hash = (hash * 10) + *cp++ - '0';
    return hash;
}
```

fig. 6



```
//Calculate the confidence
double conf = 0;

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

}
```

fig. 7

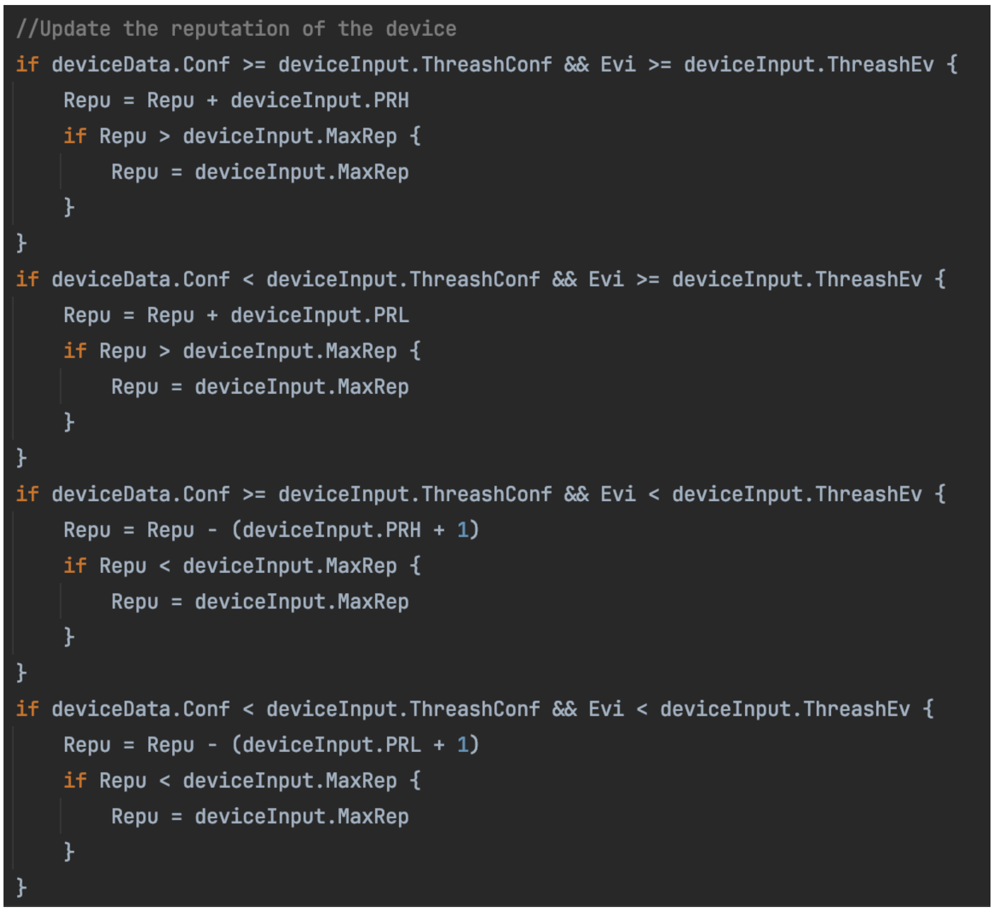


```
    //Check triangle inequality
    if (float64(deviceData.Obs+deviceNeigh.Obs) >=
        Distance(New(deviceData.X, deviceData.Y),
            (New(deviceNeigh.X, deviceNeigh.Y)))*1000) && ((Distance(New(deviceData.X, deviceData.Y),
        (New(deviceNeigh.X, deviceNeigh.Y)))*1000 +
        float64(deviceNeigh.Obs))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

fig. 8

correlation in device observations to calculate the evidence component for the observation trust. The evidence Evi is calculated for an observation of a device based on the data received from the neighboring devices. The neighborhood information is recorded in the device entity in the blockchain. If a neighbor observation support device observation, it increases the device evidence by a value proportional to neighbor's observation confidence. Otherwise, it decreases the device evidence by a value proportional to neighbor's observation confidence. The final device confidence is weighted for the number of neighbors (num). In the case of this project, a neighbor's observation, distance with the target, support the device observation if the triangle formed by the edge between the two devices and the two edges between the devices and the target respect the triangle inequality. If the propriety isn't respected then it is impossible that the distance calculated by these devices concern the same target.

- Calculation device reputation (fig. 9): The idea come from the base paper "A Trust Architecture for Blockchain in IoT" (1). There is a clear interplay between the trust level in an observation and the data source's long-term reputation. Higher reputation of a node leads to higher trust in the node observation. The reputation of a device evolves in time. The governing principle of reputation update based on the observation confidence and the evidence of other observations is the following: the reputation reward or penalty must be proportional to the reported confidence. If a device has high confidence in its observation (i.e. Conf ≥ ThreashConf) and the observation is substantiated by other nodes (i.e. Evi ≥ ThreashEv), the device should receive a significant increase PRH in its reputation. Conversely, if a device delivers observations with high confidence that are refuted by other nodes, its reputation should also drop significantly. Similarly, rewards and penalties PRL for observations with low confidence should be lower, i.e. PRL < PRH . In the case of the project to avoid too high reputation or too low reputation, it can't be possible decrease the reputation under 0 and can't be possible increase he reputation above a maximum value: MaxRep.
- Calculation of the device observation trust: The idea come from the base paper "A Trust Architecture for Blockchain in IoT" (1). Because of the connexion between a device observation confidence and evidence, and device reputation, the trust of a device can be calculate as following: Trust = Confidence * Reputation * Evidence.



```
//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
    }
}
```

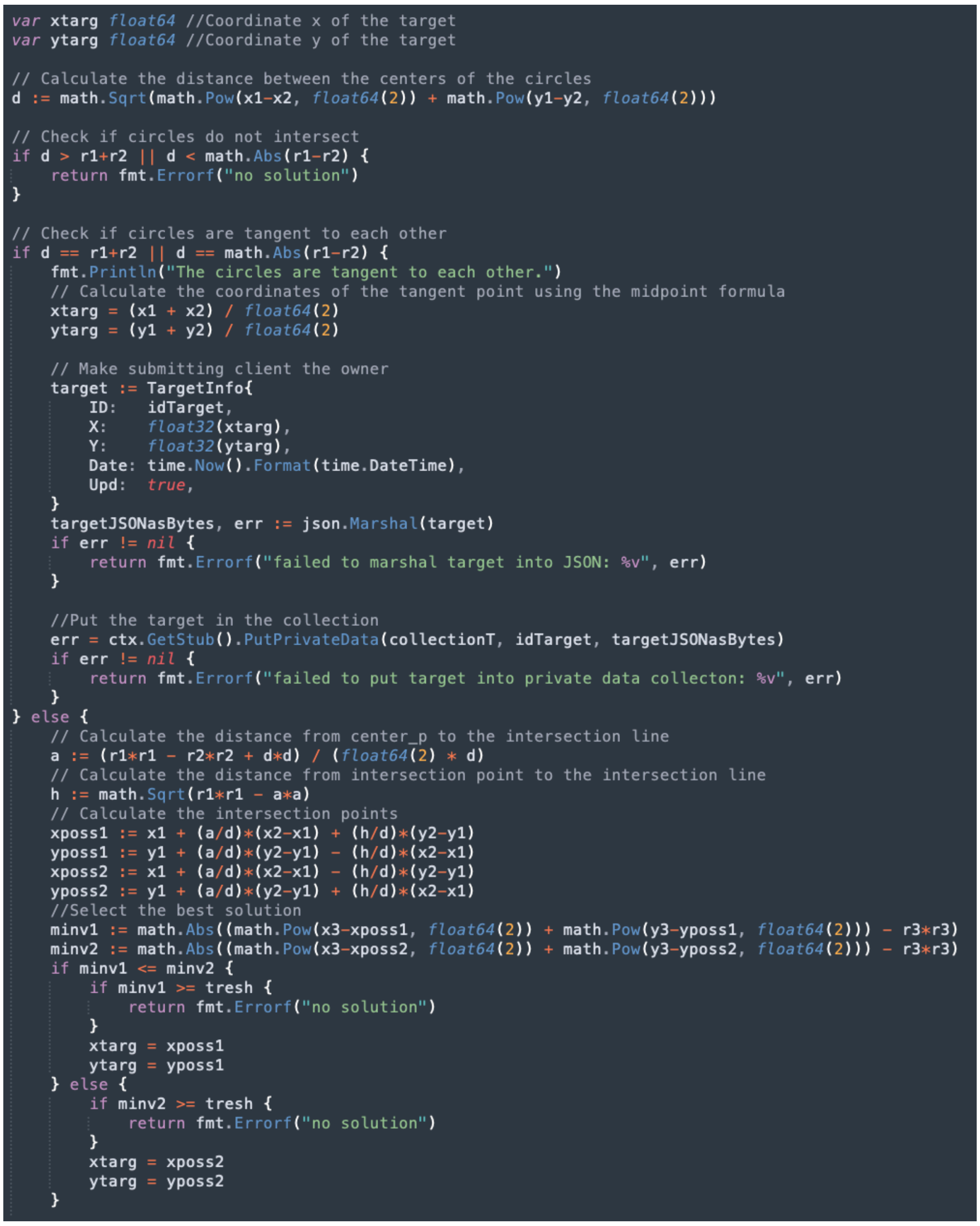
fig. 9

- Calculation of target position (fig. 11): To calculate the position of the target it is used a form of multilateration, that consists in finding the coordinates through the intersection of circumferences (fig. 10). In the case of the project it is considered the three devices with greater trust. The position of these devices can be considered as the centers of the circumferences and the calculated distances are the radius of the circumferences. Firstly, it is calculated the intersection between circumferences generated from the first two devices. If they don't intersect then can't be calculated the position of the target. If the intersection gives one point then it will be the position of target. If it gives two points then both are inserted in the analytic equation of third device circumference (x^2 + y^2 - r^2 = 0). Then, the point that gives the solution more near to 0 and it is less than the maximum error Thresh, it will be chosen as position of target. If the value obtained from expression x^2 + y^2 - r^2 is greater than maximum error the position isn't calculated because of the calculation error would be too high.

EXAMPLE OF REAL USE

In this section it is showed an example of deployment, implementation and execution of the system (fig. 12):

- Initialization: At first It is deployed the Hyperledger Fabric infrastructure with some nodes and a channel, and the contract "PositionContract" deployed. Then, a client register as Admin of own organization to the blockchain, receiving an admin account. In the organization the client creates the needed collections.



```go
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
    }
```

fig. 10

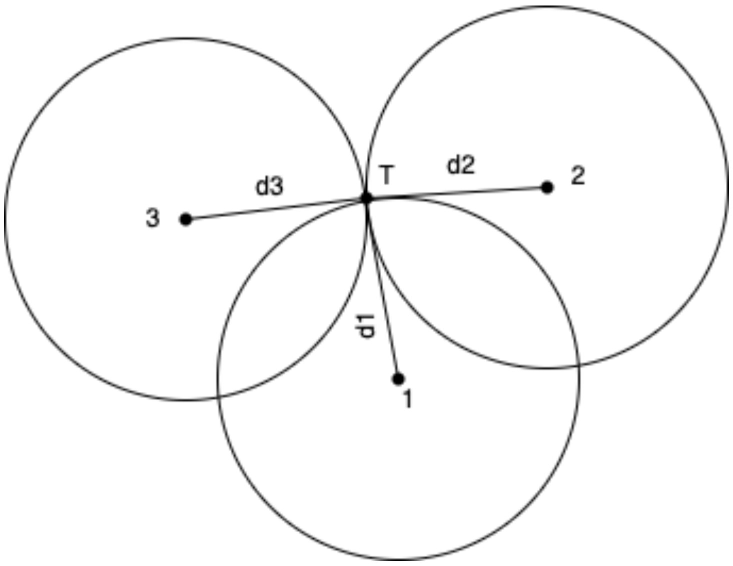


fig. 11

PROCESS

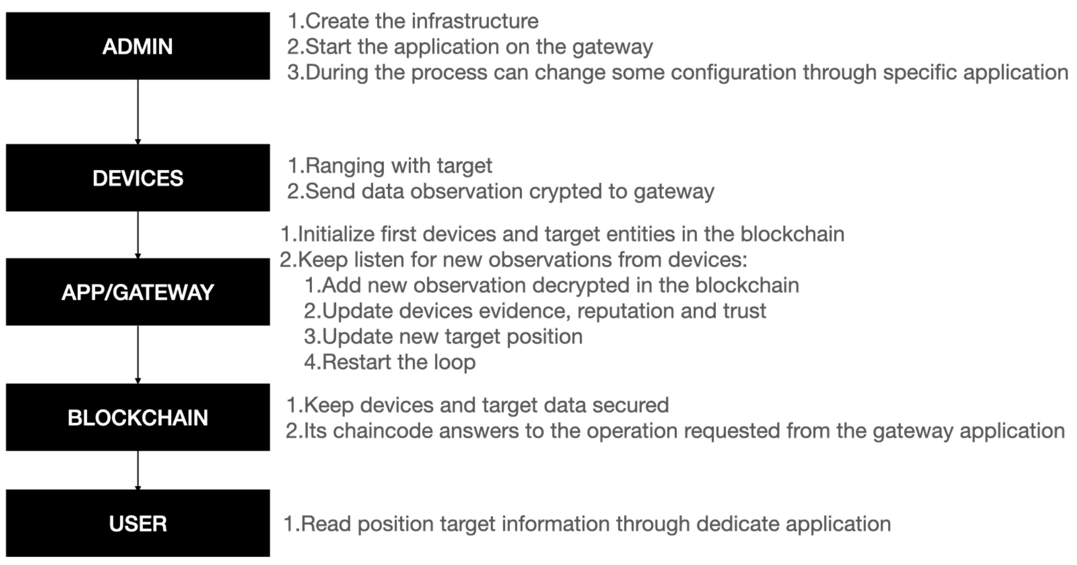

ADMIN
1.Create the infrastructure
2.Start the application on the gateway
3.During the process can change some configuration through specific application
DEVICES
1.Ranging with target
2.Send data observation crypted to gateway
APP/GATEWAY
1.Initialize first devices and target entities in the blockchain
2.Keep listen for new observations from devices:
1.Add new observation decrypted in the blockchain
2.Update devices evidence, reputation and trust
3.Update new target position
4.Restart the loop
BLOCKCHAIN
1.Keep devices and target data secured
2.Its chaincode answers to the operation requested from the gateway application
USER
1.Read position target information through dedicate application

fig. 12

The client gives the peers for the organization that will be connected to the blockchain, or take them from peers that already exist. The client also deploys the devices and target, and create an application based on prototype gateway application where indicate the initial devices and the target. The admin decides what peer or peers to be the gateway/s and connects it/them to the devices. A the end the admin starts the application on the gateway with own account and start the firmware for ranging deployed in devices. All operations in gateway refer to the contract already present in the blockchain.

- Processing: The devices do the two way ranging with the target and send their result encrypted to the gateway. The gateway checks what are the devices in system. For every devices starts to receive and decrypts every data that come from them. When the gateway captures in a buffer enough observations it compute the distance and confidence average and updates the blockchain. Then the gateway updates the devices reputation, and device observation evidence and trust. Finally, it computes the position of the target updating it in blockchain. At the end, the gateway returns to capture observations in the buffer so restarting the process.
- Admin client: An admin can create and run (with own account) an own application based on the prototype admin application. The application connects itself to the contract “PositionContract” in blockchain. Through this application the admin can manage a bit the blockchain infrastructure. The admin can read data device, read data target, update a device and create a new device.
- User client: An admin can register another client with a user account. A user can create and run (with own account) an application based on prototype user application. The application connects itself to the contract “PositionContract” in blockchain. Through this application the user can read the position of the target when it is updated.

TEST

In this section it is explained some test and their results from the point of view of correctness, security and performance of the prototype.
It can be possible to prove own-self the test following the indications in the GitHub page.
At first it is dealt with correctness and security tests. For these tests we utilize some test device with id 1,2,3,4,5 and test target with id 7. The first three devices are built to give corrects observation for localization of a specific target 7.
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 have the same position of device 1 for study the case device 1 with wrong data. 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.
In details, in case of User and Gateway application the tests are in “Position_test.go” file. All these tests are in the context of organization 1. The parameters of the gateway application are: 0.4 as minimum confidence; 1 as maximum confidence; 2 as reward or penalty in case of high evidence and high confidence, and in case of low evidence and high confidence respectively; 1 as reward or penalty in case of high evidence and low confidence, and in case of low evidence and low confidence respectively; 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 threshold that indicates when an evidence value is high or low; 20 as maximum reputation; 6 as dimension of a batch of observations to read before compute the average. The tests are:

- TestInitAsAdminOrg1: it is the test of the blockchain initialization as admin of the organization 1 with devices 1,2 and 3, and target 7.
- TestInitAsUserOrg1: it is the test of the blockchain initialization as User of the organization 1 with devices 1,2 and 3, and target 7. It should gives the message that a user can’t initialize a blockchain.
- TestInitAsAdminOrg2: it is the test of the blockchain initialization as Admin of the organization 2 with devices 1,2 and 3, and target 7. It should gives the message that an admin of another organization can’t initialize the blockchain of an organization that it doesn’t belong.
- TestAddObsAsAdminOrg1: it is the test of adding observations and confidences of devices 1,2,3 as admin of organization 1.
- TestAddObsWithErrConfAsAdminOrg1: it is the test of adding observations of devices 1,2,3,4 as admin of organization 1, where 4 has wrong confidence value. This test must gives the message that it cannot be possible insert the observation of 4 because the confidence isn’t between the min (0.4) and max (1) confidence.
- TestAddEvAsAdminOrg1: it is the test of updating evidence and trust of observations, and the updating the reputations of devices 1,2,3 as 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 reputation 7 and evidence 1.

- TestPosCalculateAsAdminOrg1: it is the test of position calculation of the target as Admin of organization 1, using devices 1,2,3. Considering the test application parameters, the initialization devices and the methodology to do this operation (see Algorithm section) the target position is updated.
- TestSeeTargetAsUserOrg1: it is the test of retrieving the data of a target updated as user of organization 1.
- TestSeeTargetNOTUpdatedAsUAdminOrg1: it is the test of retrieving the data of a target not updated as user of organization 1. The test must give the message that a user can’t read a target if it isn’t updated.
- TestSeeTargetAsAdminOrg2: it is the test of retrieving the data of a target not updated as admin of organization 2. The test must give the message that a admin of organization 2 can’t read a target of organization 1.
- TestAddEvWithIncorrectObsAsAdminOrg1: it is the test of updating evidences and trust of observations, and the updating of reputations of devices 5,2,3 as Admin of organization 1. The device 5 gives a distance that can’t correspond to same target of 2 and 3, because it doesn’t respect the triangular inequality. Considering the test application parameters, the initialization devices and the methodology to do this operation (see Algorithm section) the evidence and reputation of device 5 become respectively -1 and 2.
- TestPosCalculateWithIncorrectObsAsAdminOrg1: it is the test of updating the position of the target with the devices 5,2,3 as Admin of organization 1. The device 5 gives a distance that can’t correspond to same target of 2 and 3, because it doesn’t respect the triangular inequality. Considering the test application parameters, the initialization devices and the methodology to do this operation (see Algorithm section) the test must give the message that the position can’t be calculated because the calculus are wrong, that is the multilateration doesn’t gives acceptable result because of the device 5 fault observation.
- TestPosCalculateWith4devicesAsAdminOrg1: it is the test of updating the position of the target with the devices 1,5,2,3 as Admin of organization 1. The device 5 gives a distance that can’t correspond to same target of 2 and 3, because it doesn’t respect the triangular inequality. However, the target position will be updated because the devices with most elevated trust are considered, the devices 1,2,3.

In case of Admin application the tests are in “Conf_test,go” file. All these tests are in the context of organization 1. The parameters of the test Admin application are: ”DeviceAdmin1PrivateCollection" as collection where are memorized the devices; “TargetOrg1PrivateCollection" as collection where are memorized the target. These tests are:

- TestUpdateDeviceAsAdminOrg1: it is the test of updating the device decryption key or device neighborhood by an admin of organization 1.
- TestSeeDevicetAsAdminOrg1: it is the test of retrieving the data of a device as Admin of organization 1.

In the context of performance it is needed to consider all parameters of the test network that can be found in the various configuration files utilized. The tests are:

- The average time to do a ranging operation in the testbed environment. It is calculated from logs of testbed experiment.
- The times to do a cycle of computation of target position by the gateway application. It is calculated during a simulation of gateway application.
- The analysis of the chaincode operation performance is done through Hyperledger caliper tool:
    - Throughput and average time to do a cycle of target position computation by only considering chaincode operations.
    - Throughput and average time to do chaincode operation of reading the target.
    - Throughput and average time to do chaincode operation of reading a device.
    - Throughput and average time to do chaincode operation of updating a device by an Admin.

RESULT

From the tests executed come some results in terms of correctness, security, privacy and performance.
In terms of correctness and functionality the pro of the system are:

- All sequences of operations are be tested and works.
- The fact that the Gateway application waits for certain number of observations from every devices before go on with the computation, it lets an arrival desynchronization of device data.
- Ultra-wideband (UWB, ultra wideband, ultra-wide band and ultraband) is a radio technology that can use a very low energy level for short-range, high-bandwidth communications over a large portion of the radio spectrum. Most recent applications target sensor data collection, precise locating and tracking. Ultra-wideband is a technology for transmitting information across a wide bandwidth (>500 MHz). This allows for the transmission of a large amount of signal energy without interfering with conventional narrowband and carrier wave transmission in the same frequency band. Regulatory limits in many countries allow for this efficient use of radio bandwidth, and enable high-data-rate personal area network (PAN) wireless connectivity, longer-range low-data-rate applications, and the transparent co-existence of radar and imaging systems with existing communications systems. This means the protoype is tested on a network where measure are very precise and fast.

Instead the cons are:

- It isn't really simulated the connection between testbed network and the application.
- The network is a bit static, because of after the initialization of the system it can be possible only create new devices but no delete them. Moreover the target Its is only one and there isn't any way to add more.
- All the tests have been done in a prototype case so miss certain conditions of realistic case.
- The position it is calculated with a multilateration that assumes the devices and target are in same plane.

In terms of security and privacy the pro are:

- This system consider the possibility of malicious devices through trust calculation. It can work only if there are a lot of devices and more are benign.
- This system considers also the case of malicious architecture intern nodes through PBFT consensus protocol. It can work only if there are a lot of nodes and more are benign.
- This system gives the privacy of the data between different position experiments through the collection concept.
- This system gives also a level of privacy between user (only target data) and admin (target data and devices data) assigned data through the collection concept.
- The availability and integrity of the data are assured by the blockchain.
- The data transmitted between devices and gateway are encrypted and hashed to maintain integrity and confidentiality.

Instead the cons are:

- The system doesn't consider the security during the ranging between the devices and target.
- There isn't an efficiency way to obtain reliability of data transmitted between device and gateway.
- There is a problem of availability of service. To complete the target position calculation it is needed the presence of every necessary device data. If there is an attack ddos to the a device that block its transmission then the service of calculation is blocked. This let the service to be more secure and precise because it considers all devices, but can stall if a device doesn't send any data.

In terms of performance the pro are:

- Good performance come from utilizing the ultra wide band that lets a rapid calculation of ranging, in average 0,5 milliseconds.
- In the case of considering only the chaincode operations, it is obtained that for calculating the target position the average time is 2 second and throughput 12 TPS (transaction per second). It is a good updating time if you consider that the average times for 3 devices to calculate the ranging summed together is 1,5 milliseconds. This number can raise if you consider all process of hashing and encrypting. It also raises more if there are more devices.
- In the case of considering only the chaincode operations, it is obtained that the average time and throughput to read a target, update or read a device is 1 second and throughput 37.8 TPS (transaction per second).

The cons:

- The average time to update the target position through prototype application is 9 seconds. It is an internal very large. It can lead to a very large desynchronization in time between real position and calculated position of the target. The application try to sort of buffering the data that come from devices, decreasing the impact of the difference of rapidity between data arrival and target position calculation.

CONCLUSION

In this paper it is presented a prototype iot and blockchain infrastructure to do some experiment, in specific localization, in a trustful and privacy manner. It is started from the paper "A Trust Architecture for Blockchain in IoT" (1) that gives the base and the principal method to manage the malicious devices problem. What it is obtained, it is a good prototype that can be easily dowloaded, tested and utilized to develop a more realistic case. It contains a lot of limitation as the time for position target calculation, presence of only one target, the no realistic simulation of the communication between gateway and devices, no security between devices and target, no very flexible structure, the devices can not be eliminated from the blockchain, the main process can block if a device stop tp transmit. However, the goal of the project it is reached, that is adding privatization to the the paper “A Trust Architecture for Blockchain in IoT”(1) through the concept of collection in Hyperledger Fabric. It is added two level of privatization, that is between localization experiments, and between admin and user. Moreover, in this prototype it is implemented the device trust calculation of “A Trust Architecture for Blockchain in IoT”(1) to avoid malicious devices and other technique to make more trustful the system as adding an encryption of the data transmitted between devices and gateway. Then, it is implemented a verifiable process to calculate the position of the target, to let an admin administrate a bit the system and to let user read target data. All in all, the system has been presented isn’t a deploy version but a test version, that can be helpful for future similar projects as base to develop something more efficient, realistic and secure.