Q-Learning Based Method to Secure Mobile Agents and Choose the Safest Path in a IoT Environment

Badr Eddine Sabir[†] **Mohamed Youssfi**[††] **Omar Bouattane**[†††] **and Hakim Allali**[††††],

[†, †††] Laboratory of Watch for Emergent Technologies FST, Hassan I University of Settat, Morocco
[††, ††††] Laboratory SSDIA, ENSET Mohammedia, Hassan II University of Casablanca, Morocco

Abstarct
The Internet of Things (IoT) is an emerging element that is becoming increasingly indispensable to the Internet and shaping our current understanding of the future of the Internet. IoT continues to extend deeper into the daily lives of people, offering distributed and critical services. In contrast with current Internet, IoT depends on a dynamic architecture where physical objects with embedded sensors will communicate via cloud to send and analyze data [1-3]. Its security troubles will surely impinge all aspects of civilization. Mobile agents are widely used in the context of the IoT and due to the possibility of transmitting their execution status from one device to another in an IoT network, they offer many advantages such as reducing network load, encapsulating protocols, exceeding network latency, etc. Also, cryptographic technologies, like PKI and Blockchain technology, and Artificial Intelligence are growing rapidly allowing the addition of an approved security layer in many areas. Security issues related to mobile agent migration can be resolved with the use of these technologies, thus allowing increased reliability and credibility and ensure information collecting, sharing, and processing in IoT environments, while ensuring maximum autonomy by relying on the AI to allow the agent to choose the most secure and optimal path between the nodes of an IoT environment. This paper aims to present a new model to secure mobile agents in the context of the Internet of Things based on Public Key Infrastructure (PKI), Ethereum Blockchain Technology and Q-learning. The proposed model provides a secure migration of mobile agents to ensure security and protect the IoT application against malevolent nodes that could infiltrate these IoT systems.

Keywords:
IoT; q-learning; blockchain; ethereum; smart contract; solidity; PKI; multi-agent systems; mobile agents.

1. Introduction

IoT is growing exponentially in telecommunication [1]. Enabling heterogeneous devices to connect with one another to support various applications to serve users with different requirements [2-3]. It will be an indispensable part of the future Internet [4-5, 6]. It is referring to an approach where an extensive number of physical objects are interconnected and connected to the Internet [7]. It has become more and more real in today's life and is being part of pervasive and ubiquitous computing networks offering distributed and transparent services [8]. IoT is considered to be a good way to achieve Smart City [9-10].

Mobile agents are widely used in the context of the IoT and due to the possibility of collecting, sharing, processing and transmitting their execution status from one device to another in an IoT network, they offer many advantages such as reducing network load, encapsulating protocols and exceeding network latency. Agent-based systems enable cognitive management without constant human intervention [11-12]. An agent-oriented infrastructure enables flexible coordination between IoT devices including robots, smartphones and sensors. Functionality such as smartness, autonomy and dynamicity are extremely required for IoT based infrastructure and can be offered by the presence of agents [11]. Besides using agents, processing can be performed closer to actual data sources to reduce the cost of processing [13]. The use of mobile agents in an IoT network is highly recommended due to the various advantages they can offer [14].

Therefore, these main questions arise: During the migration process how to adopt the most secure and optimal path to reach the target while protecting the IoT application against malevolent platforms (nodes) and agents? The authors in [18] describe four kinds of threat scenarios faced while using mobile agents: Agent corrupts the Platform (AcP), Platform corrupts Agent (PcA), Agent corrupts Agent (AcA) other malicious entities (third-party programs) corrupts Agent (OcA).

In this article, we present a model architecture to secure mobile agents and protect them against different types of threats in the context of IoT. The proposed model aims to ensure a secure migration of mobile agents, security and protect IoT applications against malicious agents by using on the one hand the reinforcement learning algorithm Q-learning to take the safest path in a dynamic IoT environment, and on the other hand the Smarts Contracts [15] [16] or the PKI, depending on the application and its environment, to ensure that after the migration operation, agent n 'has not been altered.

After presenting the related works in the second section, the third section is devoted to the description of the Q-learning, Blockchain technology, the Ethereum network, Smart Contract JTW, the PKI, the X.509 Certificate and the PKCS #12 KeyStore form. In the fourth section, we

Manuscript received October 5, 2024
Manuscript revised October 20, 2024
https://doi.org/10.22937/IJCSNS.2024.24.10.8

will present the proposed model and its description. The last section gives some concluding remarks and perspectives for future related works.

2. Related Work

We describe related work reported in the literature with regards to the security of mobile agents in a multi-agent environment. Authors in [17] present a new technique of introducing Agent Identity in distributing the Symmetric Key to the newly attached Agents of the Platform thru USB. During the registration of the Agent and its Services at the Platform, it must obtain the key from the Key Distribution Process. The authors propose a novel idea in fixing the Identity of the Agent for getting its Shared Key from the Key Distribution process. Every Trusted Agent in a platform has tiny hardware called USB Dongle, which is password protected. It is configured during the environment formation initially. The work presented in [18] proposes a security framework that can be effectively used to protect agents from attacks by malicious hosts. The framework is based on restricting the access level of the agent according to the trust level that is assigned to the current host.

In [19], the authors illustrate the use of mobile agent systems in distributed applications in the domain of Ambient Intelligence. They focus on the ability to improve privacy by hiding information using the agent architecture. The shown scenario clarifies the necessity to consider the particular security requirements of mobile agents. In traditional security methods, discovering the determined private key is enough for message decryption that can be done through malicious attacks to the network nodes or listening to communication links. Accessing the private key can be considered as the endpoint of a malicious process. Authors in [20] propose an approach to improve private key security using two strategies: Encrypting the private key using an encryption algorithm (AES algorithm is used in this paper) and breaking the encrypted private key into different units.

The paper [21] described the general security requirements for mobile agent systems and existing security measures. Especially, they pointed out some weaknesses in the field of protecting the carried data of mobile agents. To mitigate this issue, they implemented a trust and reputation management to provide a secure path for mobile agent data protection. Our work aims to present a simple approach to guarantee the security of the migration of mobile agents in an IoT network through ensuring maximum autonomy by relying on reinforcement learning to allow the agent to choose the most secure and optimal path between the nodes of an IoT environment. And on the other hand, by ensuring the non-repudiation and their integrity using the Blockchain technology to ensure the non-repudiation and the integrity of the mobile agent.

3. Background

3.1 Reinforcement learning

Reinforcement learning is a technique of artificial intelligence that allows an agent to learn by interacting with an environment, he undertakes different actions by failure and by success to accumulate experiences. It is based on a system of rewards and penalties to allow the agent to learn to solve a problem independently [22].

The figure (Fig. 1) shows the main components of the standard model of reinforcement learning. Agent: it is the learner and the decision maker. It interacts with everything that is outside of the agent: it is the environment. Environment: In reinforcement learning, the "environment" is typically a set of states that the "agent" tries to influence through his choice of "actions" [23]. The reward (R), in an environment, indicates whether an agent makes the right or wrong choice (actions) and the agent according to these rewards tries to learn by maximizing the total rewards, which allows him to update his policy to arrive at the best and optimal policy [24].

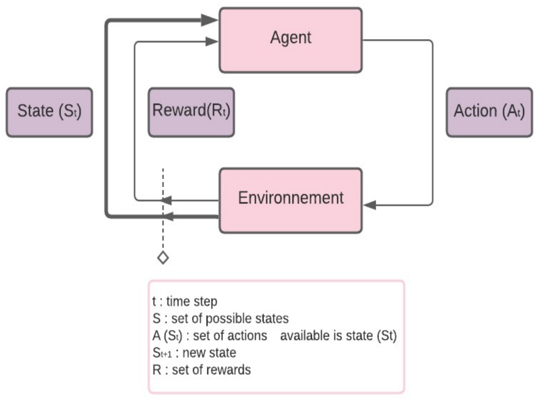

Fig. 1 Interaction agent-environment

The interaction between the agent and the environment represented by the (Fig. 4) occurs at each time step sequence t. At each time step t the agent receives a representation of the state of the environment (St) and following this representation the agent will choose an action (At) and is assigned a numerical reward (Rt+1) to change to a new state (At+1)

At each time step, the agent carries out a mapping to have all the probabilities of choosing all the possible future actions.

3.2 Mathematical formulation of reinforcement learning problems - Q-learning [25]

The mathematical formalism on which reinforcement learning algorithms is based is based on the Markov Decision Process (MDP). MDP is a means, which allows to formally describing a problem to be solved in RL, it is characterized by a quadruple (S, A, T, r) where:

- S : Set of finite states in the environment
- A : all actions in the environment
- T : $S \times A \times S \rightarrow [0,1]$ is the state transition function which describes the probability (1) of ending up in state s' when executing action a in state s

$$T(s, a, s') = P[S_{t+1} = s' \mid A_t = a, S_t = s]$$

- r : is a function of $S \rightarrow A$ in $\mathbb{R}$ which indicates the reward received by the system after each state transition, it is the expectation E of R_{t+1} knowing that $S_t = s$, $A_t = a$, $S_{t+1} = s'$.

$$r(s, a, s') = E[\, R_{t+1} \mid S_t = s, A_t = a, S_{t+1} = s' \,]$$

In order to resolve an MDP, we will define a policy that characterizes the behavior of the agent as well as a prediction of the reward obtained by this policy. This resolution consists in maximizing the return Gt that represents the sum of all the rewards while giving less importance to future rewards.

$$G_t = R_{t+1} + \gamma\, R_{t+2} + \gamma^2 R_{t+3} + \ldots \text{ avec } \gamma \in [0,1]$$

Or

$$G_t = R_{t+1} + \gamma\, G_{t+1}$$

Politics (π) is the way in which the agent, who perceives and acts in an environment, will behave in certain situations. The goal of reinforcement learning is to find the optimal policy [6] which seeks to maximize the sum of the rewards. The measure of the rewards for an action a in a state s following a policy π is defined by the action value function:

$$Q_\pi(s, a) = E_\pi\{G_t \mid S_t = s, A_t = a\}$$

The goal of reinforcement learning is to choose actions that lead to maximum rewards which can be represented by the action value function $Q_\pi(s, \alpha)$. The function optimal action value is the maximum action value function over all policies denoted by:

$$Q_*(s, a) = \max_\pi (Q_*(s, a))$$

Q-learning is a reinforcement learning algorithm without a model, it makes it possible to find an optimal policy in the sense of maximizing the expected value of the total reward on all successive stages, starting from the current state. Q-Learning assesses the quality of an action taken to move to a state rather than determining the possible value of the state to which it is moved. The heart of the Q-learning algorithm is a Bellman equation in the form of a simple value iteration update, using the weighted average of the old value and the new information:

$$Q(s,a)_{new} = Q(s,a)_{current} + \alpha * (R(s,a) + \gamma * \text{Max}(\text{next state, all actions})_{MRPA} - Q(s,a)_{current}) \ (1)$$

- γ : defines how much we will give importance to future rewards, it can have a value in 0 and 1. The closer it is to 1 the program will treat future rewards almost the same way, on the contrary more Gamma is less than 1 importance to diminished future rewards. In our model, we have chosen a value close to 1.
- MRPA: Maximum future reward expected given the new state and all possible choices in that new state.
- α : learning rate

Where α stands for the learning rate and γ for discount factor.[23]

Reinforcement learning to train an agent to solve a given problem, does not need data, an agent learn and make decisions depending on the environment, and consequently, by also modifying the system of rewards. The agent's goal is to seek optimal solutions while maximizing their total rewards.

3.3 Blockchain Technology

In 2008 the Blockchain was first propagated through bitcoin by Satoshi Nakamoto [24] to assure all parties that the payer had the means to satisfy the debt before concluding any transaction [25]. Basically, bitcoin was created with Blockchain technology to transfer money. But now Blockchain is used in many other areas.

A Blockchain [26] is a decentralized distributed database that maintains a continuously growing list of data in a public or a private peer-to-peer network. Duplicated to all the peer nodes of the network, Blockchain Technology offers a secured system, between the untrusted collaborators, which everyone on the network can check and interact with but no one can control or alter with. This allows the Blockchain to be a trustworthy source without the requirement of a third-party [27].

Is a series of block*s*; each block is connected to its previous blocks, every block has the cryptographic hash code, previous block hash, and its data [28]. As shown in Figure 1, each block in the Blockchain is connected to the previous block, containing a hash of the previous block as shown. As a result, the history of transactions on the Blockchain cannot be altered or deleted without completely changing the content of the Blockchain [29]. This is what makes the security force of the Blockchain technology. A Blockchain network is formed by one or more nodes. A node can be any electronic device (a computer, a telephone, etc.) if it is connected to the Internet and has an IP address, and each node created has a complete and separate copy of the Blockchain. All these Nodes connect to form a Blockchain network. A transaction is not sent to the network but rather to a network node that it communicates with the other network nodes.

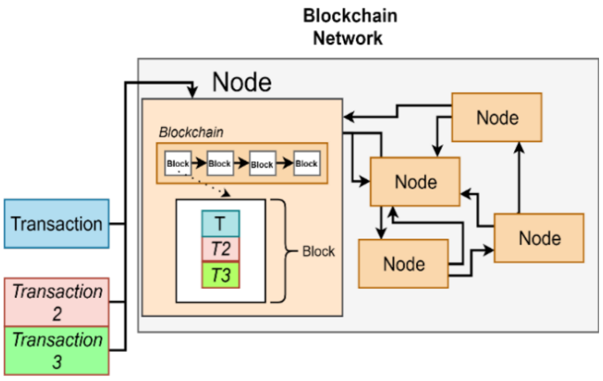

Fig. 2 Simplified diagram of a Blockchain network

The Blockchain has several characteristics, among which we can cite [30]:

- Decentralization: In the Blockchain, third parties are not required to verify transactions. Consensus algorithms and cryptographic mechanisms are used to maintain data consistency on Blockchain networks.
- Persistency: It is not possible to delete transactions that have already occurred.
- Auditability: Each transaction on the Blockchain refers to the previous transaction. This makes it easy to verify and track each transaction.
-

3.4 Ethereum

Ethereum is a Blockchain platform created by Vitalik Buterin and discusses some limitations of Bitcoin [31]. Allow users to run distributed applications in a decentralized manner. This means that applications running on Ethereum are available everywhere and every time [32-33].

Ethereum has several important elements. Among which we can cite [34]:

- Account: Every account on Ethereum has a 20-bytes address and consists of four parts, namely nonce- counter, storage, ether balance, and contract code.
- Transaction: Transaction in Ethereum refers to a signed data package that stores messages.
- Technology used: Ethereum uses several technologies including web technology, client/node implementation, and data storage.
- Consensus algorithm: Ethereum has 3 types of consensus algorithms, namely Proof Stake (PoS), Proof of Authority (PoA), and Proof of Work (PoW). The most common one is PoW. The underlying principle in this consensus algorithm is the complicated mathematical puzzles which consumes a certain amount of power to find the solution but whose verification is comparatively fast and easy. The process of finding a solution to the puzzle is known as Mining**,** while the nodes executing this process are known as Miners. If the Miner manages to find the solution (hash), the new block is formed, which is distributed on the network and if validated, the blocks get added, extending the chain. The protocol used for generating the hash for every block is cryptographic hash algorithms like SHA256 which computes the hash of the current block considering the metadata like hash of the previous block. This makes each hash unique and any attempt to change the content or metadata of a block will result in an entirely different hash generating a diversion in the chain [27].

A. Smart Contract

Nick Szabo introduced this concept in 1994 and defined a Smart Contract as "A computerized transaction protocol that executes the terms of a contract [35]. Nick Szabo suggested translating contractual clauses into code and embedding them into a property that can Self-enforce them [36]. Within the Blockchain context, Smart Contracts are scripts recorded on the Blockchain [37]. Since they reside on the chain, they have a unique address. We trigger a Smart Contract by addressing a transaction to it. It then executes independently and automatically in a prescribed manner on every node in the network, according to the data that was included in the triggering transaction [38]. The nodes in the network interact with the contract by requesting the functions of the contract code once it is deployed on the network. The Smart Contracts are invulnerable and cannot be alterable, even by the author, once deployed on the network.

Smart Contracts on Ethereum are written in a high-level language and compiled via the Ethereum Virtual Machine. The most used programming language is Solidity [39] which we will use to write our Smart Contract.

4. Proposed model

4.1 Components of the proposed model

The proposed model shown in Figure 2 aims to provide a secure migration of mobile agents to ensure security and protect the IoT application against malevolent agents by ensuring the non-repudiation and their integrity using Blockchain technology on one hand, and on the other hand taken the saftest path in a dynamic IoT environment using reinforcement learning algorithm Q-learning.

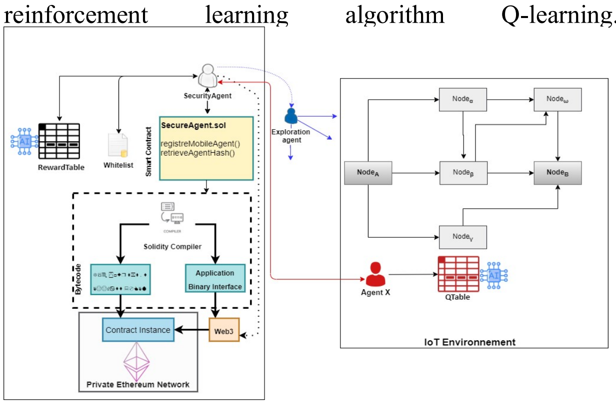

Fig. 3. model architecture based on Blockchain technology.

Here are the main components of our model:

- Agent X : An agent can be attached to a source device to collect information or perform actions on the source device. An agent can migrate to another device keeping its state to perform operations on the destination device.
- Nodes A,B,N : Nodes present in the IoT environment
- Whitelist: It contains hashes of transactions returned by the Ethereum network after the registration of a new agent and the hash of the exploration agent. An agent identifier (AID) is associated with a transaction hash.
- Security Agent: Mainly performs the operations of establishing the list of trustworthy nodes by preparing the rewards table, recording the hash of the source code of agents who wish to migrate to other devices in the Blockchain network and verifying the agents afterwards their migration.
- Exploration Agent: Agent sent by the security agent to detect the nodes presenting a threat and the agents deign to trust.
- Rewards Table: the rewards table representing the IoT environment in which the agent can moves and the consequence of each move.
- QTable: learning table calculate based on the Reward Table allowing the mobile agent to have visibility of the environment and move around independently while choosing the most optimal and most secure path.
- SecurityAgent.sol: SecurityAgent.sol is a Smart Contract that provides two functions:

registerMobileAgent (AID, AgentHash): this function allows us to send a transaction to the Blockchain network to register the source code of the agent. The function takes in two parameters, the hash of the source code of the agent (AgentHash) and the Agent Identifier (AID) and returns the identifier of the transaction recorded in the Blockchain (tranactionHash).

retrieveAgentHash (transactionHash): this function allows us to recover the hash of an agent's source code from the Blockchain network. The function takes the identifier of the transaction that allowed registering the hash of the source code of the agent and returns the hash of the source code of the agent.

This Smart Contra is developed in Solidity programming language. It's a Contract Oriented Language, used for writing Smart Contracts that can be deployed on Ethereum Virtual Machine. It follows an object-oriented approach and support features like inheritance and complex data types.

- Solidity Compiler: SecurityAgent.sol is what we call “Contract Definition”. This code will not be executed on the Ethereum network, we need to compile our “Contract Definition” using a “Solidity Compiler” which will produce two separate files. On one hand, a file which will contain“Byte Code” which will then be deployed on the Ethereum network in the form of a contract instance using

In our case Truffle which is a development environment and a testing framework that helps in the automatic compilation and deploying of contracts on Blockchain, and also used to deploy contracts on a private Ethereum Blockchain. And on the other hand, the compilation will produce what is called “Application Binary Interface (ABI)”. This ABI will be used to call the functions exposed by the Smart Contract instance deployed in the Ethereum network using the Web3 library. Ethereum provides an interface with the Ethereum network for developers in the web3.js API. This allows applications to interpret events sent from the Ethereum network and to submit transactions to the network [40].

4.2 Secure migration of mobile agents

The operation of securing agents is carried out in two stages:

Choose the safest path

The following diagram shown in Figure 3 illustrates the migration steps of an agent from a source node to a destination node.

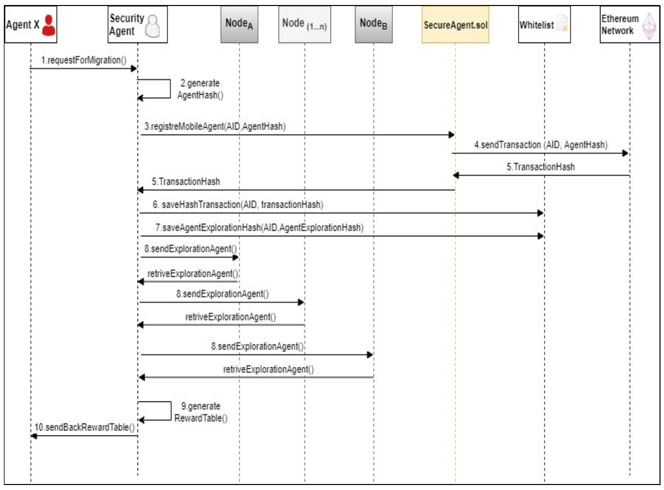

Fig.4 Sequence diagram: Migration of an agent

- Agent X needs to move from NoidA to NodeB. He requests Agent Security.
- The Security Agent generates the hash of the source code of the Mobile Agent.
- Invokes the registerAgent (AID, hash) method of the SecureAgent.sol smart contract
- The Security Agent invokes the "registerMobileAgent" function of the SecureAgent.sol Smart Contract, passing in parameter the hash of the Mobile Agent source code and the AID of the agent.
- The hash of the agent source code is registered in Blockchain Ethereum Network.
- The result of this operation is an identifier of the transaction validated on the Blockchain in the form of a hash of the transaction.
- The Security Agent registers the hash of the transaction in the whitelist by associating it with the AID of the Mobile Agent for which the source code has been saved in the Blockchain.
- The Agent Security sends an exploitation agent to the various nodes of the environment present at the moment. The agent generates the hash of the exploitation agent which keeps in the whitelist: registerExploitationAgent (AID, hash).
- After the exploitation agent returns, the Agent Security checks if the operating agent has been modified by calling the vrifyAgent (AID) method.
- Agent Security prepares the RewadTable based on the verification results after the agent returns. The rewards system defined by the RewardTable is represented by a square matrix of dimension of the number of possible states. This table defines for each state the possible reward for migrating to the other nodes according to the possible actions.
- The absence of a state transition from one node to another is represented by a reward value of -1. When the operating agent returns to the source without altering its code then the reward is zero to designate a normal passage). When the operating agent code is altered then a penalty is assigned (PENALTY = -10). The final step which represents the arrival at the target node, being awarded a maximum reward defined by the constant REWARD = 100
- The objective is to teach the agent how to evolve in the environment by taking the most secure and optimal path.
- In the learning stage, the agent will explore the environment randomly in several iterations which represent the number of learning epochs. For each iteration, the agent tries to repeat several attempts, starting from a random initial state, to act on the environment by randomly taking one of several actions. The environment gives him a reaction containing a reward allowing the agent to pass to another state. This allows him to update his QTable learning table allowing him to have visibility on the environment. The QTable is calculated using the Belman equations defined by the equation using the alpha and gamma parameters and the RewardTable
- In this operation, the agent tries in several iterations (number of learning epochs), to find an exit solution from the labyrinth. For each iteration, the agent starts from a random state of the environment, then repeats the random moves from one state to another taking possible actions. For each state transition, it calculates the value of the QTable corresponding to the transition from the current state to the next state. The operation is repeated until he finds the final step.
- After building the final QTable, the agent will be able to move from any node in the environment to reach the destination node.

To validate our model, we define an environment in the form of a network for which we can choose the number of rows and the number of columns by filling it in a random way using the following java implementation:

```java
this.map=new char[rows][cols];
for (int i = 0; i < rows; i++) {
    for (int j = 0; j < cols; j++) {
        double rnd=Math.random();
        if(rnd<0.75) this.map[i][j] ='0';
        else {
            this.map[i][j] = (Math.random() > 0.5)? 'P' : 'W';
        }
    }
}
```

Fig. 5 Generation of an environment in the form of a labyrinth

We randomly define different states of the environment by choosing 75% of the normal states associated with the '0' character (**trusted environments**) and 25% of the states that define either a 'P' penalty (**risky environment**) or an obstacle. 'W' (**environment not accessible**). An example of an environment generated in the simulator that we have developed is shown in the following figure:

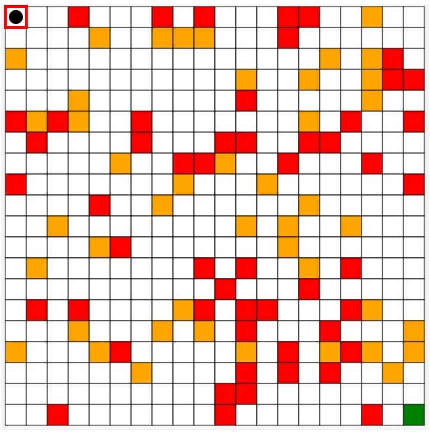

Fig 6. Proposed model architecture based on Blockchain technology

In the next step, the application calls the buildRewardTable operation which builds the reward table representing the IoT environment in which the agent travels. This operation is carried out using the following method:

```java
public void buildRTable(){

    for (int k = 0; k < statesCount; k++) {
        int row=k/gridWidth;
        int col=k%gridWidth;
        for (int s = 0; s < statesCount; s++) {
            System.out.println(s+"->"+k);
            R[k][s]=-1;
        }
        if(map[row][col]!='F'){
            for(int[] d :directions){
                int c=col+d[1];
                int r=row+d[0];
                if((c>=0 && c<gridWidth)&&(r>=0 && r<gridHeight)){
                    int st=r*gridWidth+c;
                    if(map[r][c]=='0')
                        R[k][st]=0;
                    else if(map[r][c]=='W')
                        R[k][st]=-1;
                    else if(map[r][c]=='F')
                        R[k][st]=EnvGrid.REWARD;
                    else if(map[r][c]=='P')
                        R[k][st]=EnvGrid.PENALTY;
                }
            }
        }
```

Fig 7. Preparation of the Reward table

After the initialization operation, the training operation can be started, allowing the QTable to be built. The algorithm for this step is described by the following code:

```java
public void train(){
    Random random=new Random();
    for (int i = 0; i <1000 ; i++) {
        this.currentStat=random.nextInt(envGrid.statesCount);
        while(!isTerminalState()){
            int[] possibleActionsFromState=possibleActionsFromState(currentStat);
            int index=random.nextInt(possibleActionsFromState.length);
            int nextState=possibleActionsFromState[index];
            double q=Q[currentStat][nextState];
            double maxQNextState=maxQ(nextState);
            double r=envGrid.R[currentStat][nextState];
            double value=q+alpha*(r+gamma*maxQNextState-q);
            Q[currentStat][nextState]=value;
            currentStat=nextState;

        }
    }
}

public int[] possibleActionsFromState(int state){
    List<Integer> states=new ArrayList<>();
    for (int i = 0; i <envGrid.statesCount ; i++) {
        if(envGrid.R[state][i]!=-1) states.add(i);
    }
    return states.stream().mapToInt(i->i).toArray();
}
```

Fig 8. Building of the QTable

Figures 9 and 10 represent graphs which show the evolution of the number of stages of the courses of each learning epoch respectively for the first 100 epochs and the last 100 epochs for a total of 1000 learning epochs.

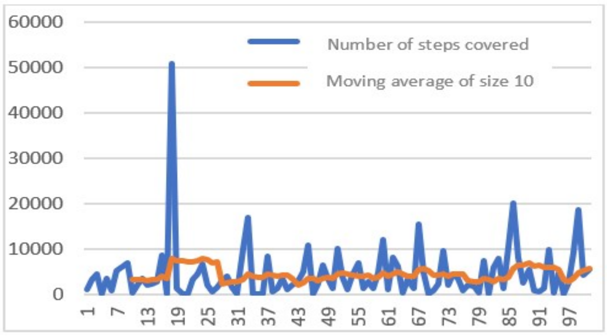

Fig 9. Number of states of the learning path for the first 100 epochs

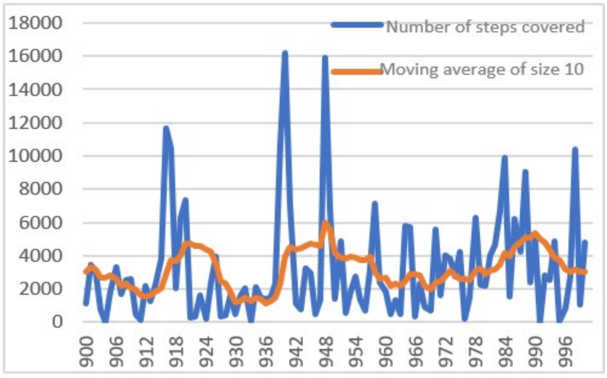

Fig 10. Number of states of the learning path for the last 100 epochs

After training, we ask the agent to find the optimal and most secure path starting from different positions. This time the agent uses his experience with a 100% Exploitation Rate and 0% Exploration Rate allowing him to maximize his rewards. Figures 11,12,13 and 14 show the trajectories taken by the agent for three trials at different initial positions for different executions (different environments). We find that the agent correctly takes the optimal and most secure path, sometimes with the preference to cross a penalty point with a minimum of trips.

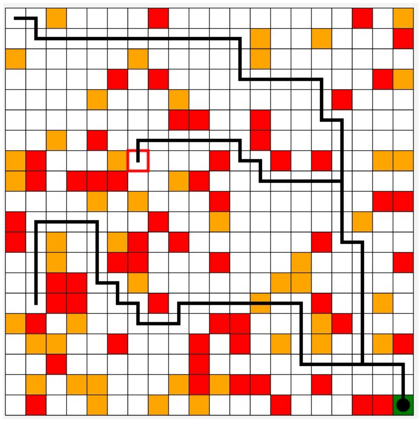

Fig 11. Execution 1

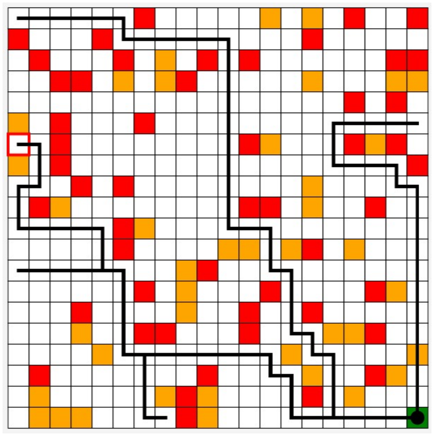

Fig 12. Execution 2

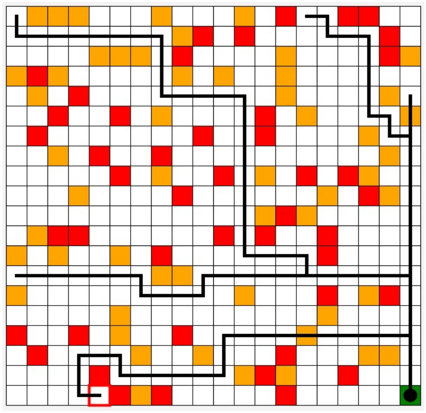

Fig 13. Execution 3

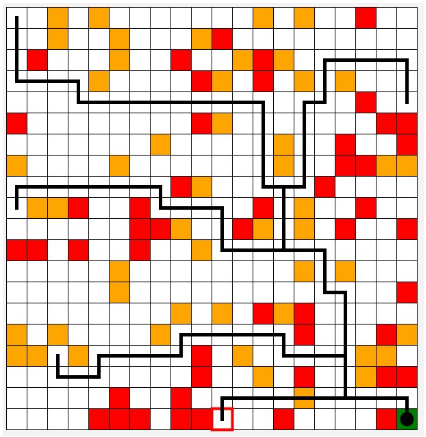

Fig 14. Execution 4

Verification of agent integrity after migration

Between the recovery time and creating a list of trusted nodes that are the most trusted, and migrating the agent, the risk of one of the trusted nodes being affected or attacked is minimal but not absent. To eliminate this risk, we will add a second layer of security to ensure non-repudiation and their integrity using Blockchain technology.

The following diagram shown in Figure 15 illustrates the migration steps of an agent from a Temperature Sensor to Smart Device in chronological order:

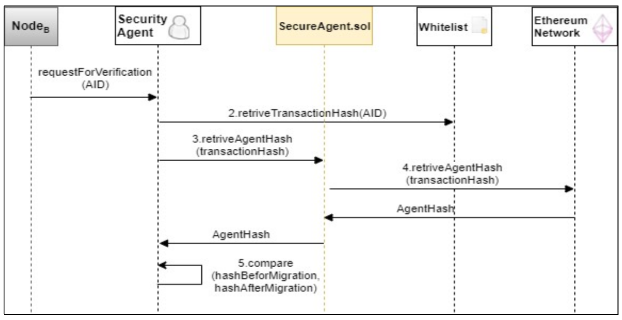
Node$_B$
Security Agent
SecureAgent.sol
Whitelist
Ethereum Network
requestForVerification (AID)
2.retriveTransactionHash(AID)
3.retriveAgentHash (transactionHash)
4.retriveAgentHash (transactionHash)
AgentHash
AgentHash
5.compare (hashBeforMigration, hashAfterMigration)

Fig 15. Migration of an agent

- After the migration of the agent, the host Node request for verification
- The Security Agent retrieves the transaction identifier from the whitelist passing in parameter the AID of the Mobile Agent.
- The Security Agent invokes the "retrieveAgentHash" function of the Smart Contrat “SecureAgent.sol”
- SecurAgent.sol Smart Contract passing the transaction identifier.
- The hash of the source code of the Mobile Agent stored in the Blockchain is returned to the Security Agent.
- The Security Agent compares the hash recovered from the Blockchain with the current Mobile Agent hash.
- If both hashes are similar, then the Security Agent allows the Mobile Agent to return to the Sensor Temperature.
- If both hashes are not similar, the Security Agent, in this case, destroys the Mobile Agent.

4. Conclusion

This document presents an overview of the current state of research in the field of mobile agent security in a multi-agent environment, as well the utility of employing mobile agents in IoT systems such as reducing network load, encapsulating protocols, and exceeding network latency.

An architecture using the Q-learning and Blockchain technologies has been offered in this document to secure mobile agents and protect them against different types of threats in the context of the IoT by opting for the most secure path during the migration.

Our future work aims to set up the private Ethereum network using before testing the implementation of the proposed model, based on the smart home use case with multiple IoT devices using mobile agents. To this end, we are in the process of developing a smart home testbed environment based on web technologies.

References

[1] Alam, Tanweer, and Mohamed Benaida. "CICS: Cloud–Internet Communication Security Framework for the Internet of Smart Devices.", International Journal of Interactive Mobile Technologies (iJIM) 12, No. 6 , pp. 74-84, 2018
[2] Lake, David, Ammar Rayes, and Monique Morrow. "The Internet of things.", The Internet Protocol Journal, Vol. 15, No. 3, pp. 10-19, 2012
[3] Lee, Gyu Myoung, and Jeong Yun Kim. "The Internet of Things-A problem statement." International Conference on Information and Communication Technology Convergence (ICTC), South Korea, 2010
[4] Li, Shancang, Li Da Xu, and Shanshan Zhao. "The internet of things: a survey.", Information Systems Frontiers , Vol. 17, No. 2, pp. 243-259, 2015
[5] Anithaa, S. K., S. Arunaa, M. Dheepthika, S. Kalaivani, M. Nagammai, M. Aasha, and S. Sivakumari. "The Internet of Things-A survey.", World Scientific News 41, pp. 150, 2016
[6] Weyrich, Michael, and Christof Ebert. "Reference Architectures for the Internet of Things." IEEE Software, Vol. 33, No. 1, pp. 112-116, 2016
[7] L. Järvenpää, M. Lintinen, A. Mattila, T. Mikkonen, K. Systä and J. Voutilainen, "Mobile agents for the Internet of Things," 17th International Conference on System Theory, Control and Computing (ICSTCC), Sinaia, Romania, October 11-13,2013.
[8] S. Bosse, "Mobile Multi-agent Systems for the Internet-of-Things and Clouds Using the JavaScript Agent Machine Platform and Machine Learning as a Service,", 4th International Conference on Future Internet of Things and Cloud (FiCloud), Vienna, Italy, August 22-24, 2016.
[9] Zanella, Andrea, Nicola Bui, Angelo Castellani, Lorenzo Vangelista, and Michele Zorzi. "Internet of things for smart cities.", IEEE Internet of Things journal, Vol. 1, No. 1, pp. 22-32, 2014
[10] Jin, Jiong, Jayavardhana Gubbi, Slaven Marusic, and Marimuthu Palaniswami. "An information framework for creating a smart city through internet of things." IEEE Internet of Things Journal, Vol. 1, No. 2, pp. 112-121, 2014
[11] G. Fortino, A. Guerrieri, W. Russo, and C. Savaglio, “Middlewares for Smart Objects and Smart Environments: Overview and Comparison”, in : Internet of Things Based on Smart Objects, 2014.
[12] F. Aiello, G. Fortino, A. Guerrieri, and R. Gravina, “Maps: a mobile agent platform for wsns based on java sun spots”, in : Proceedings of the ATSN, 2009
[13] H. Hasan et al., "Secure lightweight ECC-based protocol for multi-agent IoT systems," 13th International Conference on Wireless and Mobile Computing, Networking and

Communications (WiMob), Rome, Italy, October 9-11, 2017

[14] H. Yu, Z. Shen, and C. Leung, “From Internet of Things to Internet of Agents”, International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing, August, 2013

[15] V. Buterin, A next-generation smart contract and decentralized application platform, 2014.

[16] Tanweer Alam, “IoT-Fog: A Communication Framework using Blockchain in the Internet of Things”, International Journal of Recent Technology and Engineering (IJRTE), Vol. 7, No. 6, 2019

[17] R. Kumaravelu and N. Kasthuri, "Distribution of Shared Key (Secret Key) using USB Dongle based identity approach for authenticated access in Mobile Agent Security", International Conference on Communication and Computational Intelligence (INCOCCI), India, Erode, December 27-29, 2010

[18] P. J. Marques, L. M. Silva and J. G. Silva, "Establishing a secure open-environment for using mobile agents in electronic commerce," Proceedings. First and Third International Symposium on Agent Systems Applications, and Mobile Agents, Palm Springs, CA, USA, October 3-6, 1999

[19] F. Piette, C. Caval, A. El Fallah Seghrouchni, P. Taillibert, and C. Dinont, A multi-agent system for resource privacy: Deployment of ambient applications in smart environments (extended abstract), 2016

[20] A. Esfandi and A. M. Rahimabadi, "Mobile agent security in multi agent environments using a multi agent-multi key approach," 2nd IEEE

[21] G. Geetha and C. Jayakumar, “Implementation of trust and reputation management for free-roaming mobile agent security,”IEEE Systems Journal, Vol. 9, No. 2, pp. 556–566, 2015

[22] A. Das, S. C. Ghosh, N. Das and A. D. Barman, "Q-Learning Based Co-Operative Spectrum Mobility in Cognitive Radio Networks," 2017 IEEE 42nd Conference on Local Computer Networks (LCN), 2017, pp. 502-505, doi: 10.1109/LCN.2017.80.

[23] Q. Dang, D. Wu and B. Boulet, "An Advanced Framework for Electric Vehicles Interaction with Distribution Grids Based on Q-Learning*," 2019 IEEE Energy Conversion Congress and Exposition (ECCE), 2019, pp. 3491-3495, doi: 10.1109/ECCE.2019.8912298.

[24] Satoshi Nakamoto, Bitcoin: A Peer-to-Peer Electronic Cash System, 1997.

[25] I. Purdon, E. Erturk, “Perspectives of Blockchain Technology, its Relation to the Cloud and its Potential Role in Computer Science Education”, Engineering, Technology & Applied Science Research, Vol. 7, No. 6, pp. 2340-2344, 2017

[26] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system, 2009

[27] I. Ishita, D. Kulkarni, T. Semwal and S. B. Nair, "On Securing Mobile Agents using Blockchain Technology," Second International Conference on Advanced Computational and Communication Paradigms (ICACCP), Gangtok, India, February 25-28, 2019

[28] Tanweer Alam, “Blockchain and its Role in the Internet of Things (IoT)”, International Journal of Scientific Research in ComputerScience, Engineering and Information Technology, Vol. 5, No. 1, pp. 151-157, 2019

[29] Xiwei Xu et al., “A Taxonomy of Blockchain-Based Systems for Architecture Design”, International Conference on Software Architecture (ICSA), Gothenburg, Sweden, April 3-7, 2017

[30] Zibin Zheng et al., “An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends”, 6th International Congress on Big Data (BigData Congress), Honolulu, Hawaii, USA, June 25-30, 2017.

[31] Ethereum Community, A Next-Generation Smart Contract and Decentralized Application Platform.

[32] C. Dannen, Introducing Ethereum and Solidtty: Foundations of Cryptocurrency and Blockchain Programming for Beginner”, 2017

[33] D. Patel, J. Bothra and V. Patel, "Blockchain exhumed," ISEA Asia Security and Privacy (ISEASP), Surat, 29 January - 1 February, 2017

[34] Chinmay Saraf and Siddharth Sabadra, “Blockchain Platforms: A Compendium,” IEEE International Conference on Innovative Research and Development (ICIRD), Jakarta, Indonesia, 2018

[35] Tapscott Don, Tapscott Alex, The Blockchain Revolution: How the Technology Behind Bitcoin is Changing Money, Business, and the World, 2016

[36] N. Szabo, The Idea of Smart Contracts, 1997

[37] MySQL Reference Manual, Using Stored Routines (Procedures and Functions), 2016

[38] Seung Jae Pee, Jong Ho Nang, Ju Wook Jang, A Simple Blockchain-based Peer-to-Peer Water Trading System Leveraging Smart Contracts, 2018

[39] Maximilian Wöhrer and Uwe Zdun, “Smart Contracts: Security Patterns in the Ethereum Ecosystem and Solidity,” International Workshop on Blockchain Oriented Software Engineering (IWBOSE), March 20, 2018.

[40] V. P. Ranganthan, R. Dantu, A. Paul, P. Mears and K. Morozov, "A Decentralized Marketplace Application on the Ethereum Blockchain," IEEE 4th International Conference on Collaboration and Internet Computing (CIC), Philadelphia, PA, USA, October 18-20, 2018.