Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

Bubbles of Trust: A decentralized blockchain-based authentication system for IoT

Mohamed Tahar Hammi[a,b], Badis Hammi[a,*], Patrick Bellot[a,b], Ahmed Serhrouchni[a,b]

[a] LTCI, Telecom Paristech, France
[b] Paris Saclay University, France

ARTICLE INFO

Article history:
Received 14 February 2018
Revised 8 June 2018
Accepted 20 June 2018
Available online 30 June 2018

Keywords:
IoT
Security
Authentication
Blockchain
Smart city
Ethereum

ABSTRACT

There is no doubt that Internet of Things (IoT) occupy a very important role in our daily lives. Indeed, numerous objects that we use every time, are being equipped with electronic devices and protocol suites in order to make them interconnected and connected to the Internet. In IoT, things process and exchange data without human intervention. Therefore, because of this full autonomy, these entities need to recognize and authenticate each other as well as to ensure the integrity of their exchanged data. Otherwise, they will be the target of malicious users and malicious use. Due to the size and other features of IoT, it is almost impossible to create an efficient centralized authentication system. To remedy this limit, in this paper, we propose an original decentralized system called *bubbles of trust*, which ensures a robust identification and authentication of devices. Furthermore, it protects the data integrity and availability. To achieve such a goal, our approach relies on the security advantages provided by blockchains, and serves to create secure virtual zones (*bubbles*) where things can identify and trust each other. We also provided a real implementation of our mechanism using the C++ language and *Ethereum* blockchain. The obtained results prove its ability to satisfy IoT security requirements, its efficiency, and its low cost.

1. Introduction and problem statement

Currently, over the world, Internet of Things (IoT) is involved in almost all the fields of our daily life. According to a recent *Gartner* study, 50 billion connected devices[1] will be deployed by 2020 (Gartner, 2016).[2] Indeed, citizens are gradually equipping their homes with IoT devices such as smart TVs, Internet boxes, heating systems, home's remote control, lighting systems and so on. In factories and industrial environments, the cooperation of robots and other smart tools enhances the efficiency of automation systems and allows better productions. The IoT involvement did not stopped to these use cases, but, is widely adopted in many other areas such as health care, military, agriculture and smart cities.

* Corresponding author.
E-mail addresses: hammi@telecom-paristech.fr (M.T. Hammi), bhammi@telecom-paristech.fr (B. Hammi), bellot@telecom-paristech.fr (P. Bellot), serhrouchni@telecom-paristech.fr (A. Serhrouchni).

[1] In the remaining of this paper, we use indifferently the terms device, thing, object and smart thing in order to refer to a connected smart thing.

[2] A video that shows the realized implementation, development and functioning of the approach is available on: https://www.youtube.com/watch?v=XE13QGR1czE&t=169s.

https://doi.org/10.1016/j.cose.2018.06.004
0167-4048/© 2018 Elsevier Ltd. All rights reserved.

IoT represents the principal actor to make our cities smarter. This fact was extensively addressed during the last *United Nations conference on climate change (Cop21)* held in Paris in 2016. It was concluded that connected objects have the potential to considerably reduce CO_2 emissions.[3] Besides, IoT can bring other vital applications in the context of smart cities, such as: intelligent waste management, buildings' health, environmental monitoring, intelligent transportation systems, smart parking, traffic management, smart navigation system for urban common transport riders, smart grid, and multiple other applications (Hammi et al., 2017).

Besides, IoT allowed the evolution of many other areas such as: (1) factories to what is actually called industry 4.0 (Lee et al., 2015), (2) agriculture to smart agriculture (TongKe, 2013; Zhao et al., 2010), (3) health to smart health (Hassanalieragh et al., 2015; Yang et al., 2014) and many other examples.

The idea behind IoT and its different applications, is the omnipresence of a variety of things, where they are able to interact and cooperate with each other in order to provide a wide range of services. Thus, a huge number of devices will be included. Each physical or virtual device should be reachable and produce content that can be retrieved by users regardless of their location (Hammi et al., 2018). However, It is very important that only authenticated and authorized users make use of the system. Otherwise, it will be prone to numerous security risks such as information theft, data alteration and identity usurpation. Indeed, security issues remain the major obstacle to the large scale adoption and deployment of IoT since it is highly vulnerable to attacks for numerous reasons: (1) most of the communications are wireless, which makes the system more vulnerable to numerous attacks such as identity spoofing, messages eavesdropping, messages tampering and other security issues, and (2) multiple types of devices have limited resources in terms of energy, memory and processing capacity, which prevent them from implementing advanced security solutions.

Many researchers (Alur et al., 2016; Ma et al., 2016) qualify IoT as a system-of-systems, where, multiple use case scenarios require that only trusted users can use offered services. Thus, conventional security requirements such as authentication, confidentiality, and data integrity are critical to each part of these ecosystems, including things, networks, and software applications. However, due to limitations and heterogeneity of devices' resources, existing security solutions are not fully adapted to such an ecosystem. Besides, often, the combination of multiple security technologies and solutions is needed, which leads to extra high costs. Furthermore, efficient security solutions are often centralized e.g. Public Key Infrastructure (PKI), which can cause enormous scalability issues in an environment composed of thousands of nodes. Finally, each use case apply a different security approach, architecture and deployment, which causes multiple difficulties in the integration of new services and scenarios. Consequently, it is necessary to propose new security solutions for the system-of-systems as a whole. The latter must: (1) allow an easy integration of new devices as well as new services; (2) fully adapted to IoT requirements and needs; and (3) does not depend on the type of devices, nor on the use case architecture and design.

Contributions

We believe, as many researchers (Christidis and Devetsikiotis, 2016; Malviya, 2016; Ouaddah et al., 2016), that blockchains represent a very promising technology to meet security requirements in IoT context.

Benefiting from blockchains power and resiliency, in this work, we propose an efficient decentralized authentication mechanism called *bubbles of trust*. This mechanism was implemented upon the public blockchain *Ethereum*, and aims at the creation of secured virtual zones, where devices can communicate securely. Its evaluation shows clearly its ability in meeting IoT security requirements. In addition, we provide an extensive study on the computational and energy impact as well as the financial cost of our approach on different types of devices that can compose an IoT ecosystem. Finally, These costs are compared with some existing IoT authentication schemes.

This manuscript is organized as follows: Section 2 introduces the blockchain concept and its most known technologies. Section 3 analyzes security requirements and presents our threat model. Section 4 describes the existing works that aimed to integrate blockchains to IoT. Section 5 describes our blockchain based approach. Then, Section 6 discuss and analyzes our evaluation campaign. Section 7 describes the approach's open issues. Finally, Section 8 concludes the paper and introduces our future works.

2. Background

A blockchain is defined as a distributed database (ledger) that maintains a permanent and tamper-proof record of transactional data. A blockchain is completely decentralized by relying on a peer-to-peer network. More precisely, each node of the network maintains a copy of the ledger to prevent a single point of failure. All copies are updated and validated simultaneously.

Current blockchain functioning was created to solve the double spending problem in crypto-currency (Nakamoto, 2008). However, presently, numerous works explore blockchain applications in multiple use cases and use them as a secure way to create and manage a distributed database and maintain records for digital transactions of all types (Bahga and Madisetti, 2016; Dorri et al., 2017; Huh et al., 2017; Rouse, 2015).

The blockchain ledger is composed of multiple blocks, each block is composed of two parts. The first represents the transactions or facts (that the database must store), which can be of any type such as monetary transactions, health data, system logs, traffic informations, etc. The second is called the header and contains information about its block e.g. timestamp, hash of its transaction, etc. as well as the hash of the previous block. Thus, the set of the existing blocks forms a chain of linked and ordered blocks. The longer is the chain, the harder is to falsify it. Indeed, if a malicious user wants to modify or swap a transaction on a block, (1) it must modify all the following blocks, since they are linked with their hashs. (2) Then, it must change

[3] http://blogs.gartner.com/smarterwithgartner/cop21-can-the-internet-of-things-improve-organizations-sustainability-performance/.

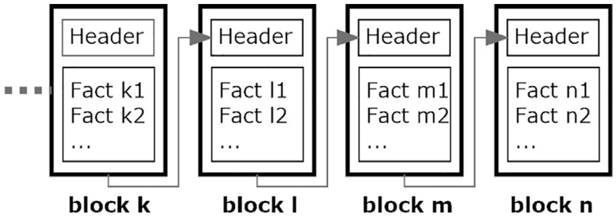


Fig. 1 – Example of a simplified blockchain.

the version of the block chain that each participating node stores. Fig. 1 depicts an example of a simplified blockchain.

There are two types of participating nodes: (1) nodes that can only read facts (passive mode); and (2) nodes that can read and write facts (active mode), commonly called miners. In order to add a new transaction to the blockchain, the following steps are performed:

1. The transaction is grouped with other transactions into what is called a block.
2. Miners verify that transactions within the block respect defined rules.
3. Miners perform a consensus mechanism to validate the added block.
4. A reward is given to the miner/miners that validate the block.
5. Verified transactions are stored in the blockchain.

In order to prove the honest validation of blocks, it exists numerous mechanisms. The most used ones are the Proof of Work (PoW) and the Proof of Stake (PoS) mechanisms.

Proof of Work: in the PoW, a miner must perform some predefined work, which is often a mathematical puzzle or challenge which is hard to compute but easy to verify. A PoW is requested for each block validation. The difficulty of the mathematical challenge can be adapted according to the time needed to validate a block and to the miners' computation power. From the one hand, PoW has the advantage of protecting transactions and blocks from being altered, since the attacker needs to validate all its fake requests and to change a part of the chain's blocks, to provide a new PoW for each altered block, as well as updating his version of the chain on all the nodes, which requires a huge computation power and energy. From the other hand, PoW suffers from some shortcomings which can have disastrous consequences. In fact, PoW requires a big energy waste in the puzzles' computations. Moreover, PoW can lead to a potential *Tragedy of Commons* (Bentov et al., 2014). Indeed, over time, mining rewards will decrease, which leads to the decrease of miners number, since the only fees that will be earned will come from transactions, which will also diminish over time as users opt to pay lower fees for their transactions. The decrease in miner's number make the blockchain ecosystem vulnerable to a 51% attack (Li et al., 2017). The latter occurs when a malicious miner (or a pool of malicious miners) controls 51%, or more, of the computational power of the network. Thus, he can create fraudulent blocks of transactions for himself, or another entity, while invalidating the transactions of others in the network. Finally, in some mechanisms, such as the *longest chain* mechanism applied in *Bitcoin*, numerous miners which validate blocks and realize the PoW, will not be rewarded, because they do not have enough power to construct the longest chain, which will cause them enormous loss.

PoW represents the widely adopted block validation method in blockchain systems e.g. *Bitcoin, Ethereum, BitShares* and *Litecoin*.

Proof of Stake: in order to solve the shortcomings of PoW, the PoS was proposed. In PoS, there is no mining, where power and time are spent in solving mathematical puzzles. The validators are called forgers. A forger can validate blocks according to how many money he owns. Which means that the more coins he owns, the more mining power he has. In a more simple way, we can compare PoS to betting games, where every forger places a bet on its block. Honest blocks which contain no fraudulent transactions, get appended to the chain and their forgers get rewarded. Each forger get the reward according to his bet. For example, if a forger bets 25% of the total bet amount, he obtains 25% of the reward amount. Finally, the forger whose block turns out to be dishonest get penalized and the amount of the bet that he had put get debited from his balance. Unlike the PoW, generally, in the PoS any forger is rewarded. However, rich forgers, are always the big beneficiaries. Furthermore, in PoW, dishonest miners are forgiven, however in PoS, they are penalized by paying their bet. There are numerous blockchain systems that use the PoS, and many others are moving from PoW to it. Examples of blockchains that use the PoS are *Peercoin, ShadowCash, Nxt, BlackCoin* and many others.

There are many other mechanisms to validate blocks like Delegated Proof-of-Stake (DPoS), Proof of Hold (PoH), Proof of Use (PoU), Proof of Stake/Time (PoST), Proof of Minimum Aged Stake (PoMAS) and Proof of Importance (PoI)[4] (Zheng et al., 2016).

Blockchains can be permissioned (private) or permissionless (public). The first category, makes restrictions on the consensus contributors. Only the chosen trustful actors have the rights to validate transactions. It does not require a lot of computation to reach a consensus, thus, it is not time neither energy consuming. Lastly, it enables the transactions' privacy, since only authorized participants can access them. The second type (public blockchains), uses an unlimited number of anonymous nodes. Based on the cryptography, each actor can securely communicate. Each node is represented by a pair of private/public keys. Any actor can read, write, and validate transactions in the blockchain. The blockchain is safe and the network consensus is reached, while 51% of the nodes are honest. Usually permissionless blockchains are energy and time consuming, because it includes a computation amount to strengthen security of the system.

2.1. *Bitcoin*

Bitcoin is a cryptocurrency and a digital payment system, based on a public blockchain. Each block of the *Bitcoin* blockchain

[4] https://cointelegraph.com/news/proof-of-importance-nem-is-going-to-add-reputations-to-the-blockchain.

contains a strong hash of its transactions called *merkle root* (Merkle, 1987) stored in the header. The latter contains also the hash of the header of the previous block.

Each node participating in the *Bitcoin* network can be a miner or not and each node stores a copy of the current blockchain.

In the mining process, transactions are ordered and timestamped, then saved into blocks. After, a consensus mechanism is executed. Indeed, in order to validate transactions, *Bitcoin* uses its defined consensus rules. More precisely, transactions have a version number, which informs *Bitcoin* nodes about the appropriate set of rules that should be used to ensure their validation (Bitcoin, 2017).

In order to share the same blockchain and avoid conflicts between miners, *Bitcoin* uses the *longest chain* rule. A conflict happens when multiples miners (in competition) generate blocks in the same time, and each miner considers its block as the legitimate one that should be added to the blockchain. For example, if two miners *A* and *B* try to add the block number *n, A* generates the block n_A and B generates n_B. Both blocks may contain a different set of transactions, and both contain the generator address for the block reward. Then, because blocks are not added and shared instantly in the network, each one assumes that its own block is legitimate. Thus, it add it to its chain and start building the next one (block $n + 1$). If *B* is faster then *A* and generates the block $n_B + 1_B$ before the $n_A + 1_A,$ then, based on the *longest chain* rule, *A* must take B's chain ($n_B + 1_B$) as the valid one and abandon the shorter chain (n_A), which will be called orphaned chain/block.

Bitcoin uses the PoW mechanism to make the system resistant against modification attacks. Thus, as described above, for each new block, the miner must provide its PoW which represents a data processing challenge, that is difficult (costly and time-consuming) to produce but easy for others to verify. More precisely, *Bitcoin*'s mining process and PoW are as follows: (1) each miner creates a block containing a header (timestamp, *merkle root* of the block's transactions, the previous block's hash, etc) and a body (transactions). Then, (2) the protocol generates a target "*t*", which represents a value $t \in]0, 2^{256} - 1]$. (3) Each miner must calculate the hash of (a) a chosen number *n* ($n \in]0, 2^{256} - 1]$) concatenated to (b) the hash of its block, in such a way that the resultant value should be $\leq t$. In other words, the miner changes *n*'s values until satisfying the equation $sha256\,(sha256\,(block) \,\|\, n) \leq t$. Once this equation is satisfied, the miner add this value to the block as a proof of work.

When a node sends a constructed block over the network, all the recipients verify the block's transactions as well as its PoW. If the major part of the network nodes agree on a block, the latter is validated and added to the blockchain. Consequently, all the other nodes update their blockchain copies, and the block creator receives its reward. The operation of the blockchain update occurs each 10 min. A miner of an orphaned block does not receive any reward, even considering that he provided the good PoW.

Theoretically, *Bitcoin* blocks can be falsified only if more than 51% of nodes are corrupted, which is currently almost impossible to realize. For example, if *Google*'s extant computing power in the cloud is used for *Bitcoin* mining, it will represents roughly 0.0019% of all of the worldwide *Bitcoin* mining operations.[5]

2.2. Ethereum

Ethereum is a public *blockchain* that provides a cryptocurrency called *Ether (ETH)* (after the fork that happened in July 2017, it exists a version of *Ethereum* called *Ethereum Classic* that uses a currency called *ETC*), used for paying financial transactions as well as applications processing. Miners replicate, validate, and store data in the blockchain network. Furthermore, they process programs called smart contracts which makes *Ethereum* a platform for decentralized applications. Smart contracts are executed by participating nodes using an operating system known as *Ethereum Virtual Machine (EVM)* (Ethereum, 2017).

As in *Bitcoin*, the mining operation consists of the creation and the validation of blocks. The block size is shorter than in *Bitcoin* and the validation time, takes only 14 s compared to *Bitcoin* which takes 10 min. Also, the reward system is different from *Bitcoin*. Indeed, *Ethereum* uses The *Ethereum Greedy Heaviest Observed Subtree (GHOST)* protocol for consensus and miners reward. A miner that validates a block that have been added to the main blockchain receives 5 *ETH*. Furthermore, according to the complexity of the executed smart contract, it receives an additional amount of *gas,*[6] paid by the sender of each transaction.

When a miner builds a block, it sends it with its PoW through the network. Within the 14 s of the consensus, each node will receive numerous blocks. Some of them are supposed to be generated at the same time. Thus, it keeps the first in its main chain and considers the others as *Uncles* (equivalent definition to orphaned blocks in *Bitcoin*). It is the chain that contains the more of Uncles (called the *heaviest chain*) that will be kept as main chain at the end of the consensus. Finally the miner gets a part of the reward of the *Uncles* (community, 2016). *GHOST* rewards also the Uncles of the accepted blocks in order to strengthen the system.

For blocks' validation, *Ethereum* uses a PoW mechanism called *Ethash*. As explained in Community (2017), participating nodes calculate a 16 MB pseudo-random cache, based on a seed computed from the block headers. From this cache, a 1 GB dataset is generated, knowing that each dataset item depends on only few cache items. The cache is stored by light clients, while dataset is stored by full clients and miners. By hashing random dataset pieces together, miners try to resolve a mathematical challenge. The verification requires only the cache to regenerate the specific dataset pieces.

Currently, there is a beta version of *Ethereum* that uses a protocol called *Casper* which relies on a PoS.

Ethereum can be used also as a private blockchain, thus the participating nodes are chosen and the proof of work mechanism is no longer required.

[5] http://www.zerohedge.com/news/2015-11-19/bitcoins-computing-network-more-powerful-525-googles-and-more-10000-banks.

[6] *Gas* represents the internal pricing for running a transaction or contract in *Ethereum*. 1 *gas* worths 0.01 microEther.

2.3. Hyperledger Fabric

Hyperledger Fabric is an open-source permissioned blockchain created by the *Linux Foundation*, more specifically by *IBM*. Unlike *Bitcoin* and *Ethereum, Hyperledger Fabric* does not provide a cryptocurrency. Transactions can be public or confidential, all depends on the nature of the stored information. *Hyperledger* uses the *Practical Byzantine Fault Tolerant (PBFT)* as a consensus mechanism. As explained in Castro et al. (1999), *PBFT* is a mechanism used in the distributed networks tolerating a certain degree of faults in order to allow the continuity of the system operations. All the participating nodes are trustful and know each other, and validating nodes are chosen randomly, but always at a majority, which protects the system against *Byzantine* imposters and Sybil attacks (Douceur, 2002).

Hyperledger Fabric allows also the development of smart contracts, called in this context *chaincodes*.

3. Security requirements and threat model

3.1. Security requirements

An IoT scheme must fulfill numerous security requirements in order to ensure the sustainability and resiliency of the ecosystem. Thus, in this section we describe the main security goals, and we introduce the criteria needed to evaluate the suitability of authentication schemes for securing IoT use cases.

Integrity: Maintaining integrity is the crucial requirement that each scheme must ensure. In our context, integrity is divided into two parts:

1. Messages (transactions/communications) integrity: an exchanged message must not be altered or modified during the network transit.
2. Data integrity: involves maintaining the consistency and trustworthiness of data over its entire life cycle. Thus, only authorized users can modify stored data.

Availability: the availability implies that resources must be accessible to legitimate users on demand. Thus, a system must be resilient against denial of service attacks especially those who target the authentication service.

Scalability: In our context, scalability represents the ability to ensure that the system size has no impact on its performances. For example, if the number of the used things explodes, the time needed for a system function such as the authentication service, must not be affected.

Non-repudiation: It refers to the ability to ensure that an entity cannot deny having performed a given action, e.g. a device cannot deny having sent a message.

Identification: The identification represents a main requirement in the majority of IoT use cases. It represents the contrary of the anonymity which ensures that any entity can make use of the system all within ensuring being anonymous to all system's entities. For example, in a smart parking scenario, when a sensor of a parking spot sends a notification, the management system must know exactly which sensor is communicating in order to update accurately the parking spots' state. Another example is the environment monitoring where a sensor monitors the level of a lake. When this sensor sends information to the monitoring platform, the latter must know exactly which sensor is communicating in order to decide about the actions to provide.

Mutual authentication: The authentication is the mechanism of proving identity. Mutual authentication represents the requirement where both communicating parties authenticate each other. This requirement is more than necessary to immune the system against spoofing the roles of entities.

3.2. Threat model

In this section we present our threat model. The latter is similar to the *model of* Dolev and Yao (1983).

3.2.1. Network model

The overall purpose of an authentication scheme is to allow multiple nodes to communicate in a trustworthy way over a non trusted network. In this work we consider a network that owns a set of things offering and using different IoT services in a centralized or a distributed architecture. Each thing communicates with a large number of other things. Exchanged messages pass through an unreliable and potentially lossy communication network, such as Internet. We also assume that all participants cannot be trusted. Indeed, the high number of smart things in the network, increases the risk of including compromised ones. Furthermore, the existing devices are of heterogeneous types and do not belong to the same use case. The network function consists in only forwarding packets and does not provide any security guarantee such as integrity or authentication. Thus, a malicious user can read, modify, drop or inject network messages.

3.2.2. Attacker model

In this work, we assume that an attacker or malicious user has a total control over the used network i.e he can selectively sniff, drop, replay, reorder, inject, delay, and modify messages arbitrarily with negligible delay. However, the devices can receive unaltered messages. Nonetheless, no assumptions on the rate of the altered messages are made. Besides, the attacker can benefit from a computation power and storage larger than the implemented devices.

However, we do not consider physical attacks on devices, where the attacker can retrieve some/all of the object's secrets such as private keys. We assume that objects are protected against physical attacks since it exists numerous methods to protect them from such attacks by making these information readable only by the device itself (Bong and Philipp, 2012; FIPS PUB, 2001).

3.2.3. Attacks

An attacker can have multiple goals, such as sending wrong informations in order to mislead system's decisions or the denial of system's services. Thus, it can conduct numerous attacks:

1) Sybil attack: in multiple cooperative use cases, the attacker simulates the existence of multiple entities (devices) that send wrong information to the service's server or management application, in order to elect decisions needed by the attacker. One example is the use case of a *Cooperative Intelligent Transportation System (C-ITS)*. In *C-ITS* the vehicles send continuously multiple information to a management infrastructure (e.g. *Cooperative Awareness Messages (CAM)* and *Decentralized Environmental Notification Messages (DENM)* in european based standards and *Basic Safety Messages (BSM)* in american based standards). These information concern the activity of the vehicles as well as their environment and are used by the management center in order to provide and enhance multiple services. For example, if the management center receives messages from multiple vehicles informing about a traffic jam or an accident, it will instantly diffuse these information to all the vehicles in the area and helps them to find better paths. Thus, an attacker can send wrong information on behalf of multiple existing or non existing vehicles in order mislead the decisions of the management center. This attack can be perpetrated in every use case that needs information from a certain number of devices in order to elect or to make a decision.

2) *Spoofing attack:* in contrary to sybil attack where the attacker try to create numerous false or virtual identities, in spoofing attack, the attacker tries to spoof the identity of a legitimate user in order to make use of his privileges.

3) *Message substitution attack:* In a substitution attack (Amoroso, 1994), the attacker intercepts valid messages during their transit and alter them in such a way that recipients accept the forged messages as if they had been sent by the original sender.

4) *Denial of service:* a Denial of Service (DoS) or a Distributed DoS (DDoS) attack is characterized by the explicit attempt by attacker to prevent the legitimate use of a service (Mirkovic and Reiher, 2004). There are two methods to conduct a DoS/DDoS attack (1) by the exploitation of a protocol flaw and (2) by flooding the target. DDoS and especially flooding attacks are among the most dangerous cyber attacks and their popularity is due to their highly effectiveness against any type of service, as they do not require identification and exploitation of protocols' or services' flaws, but just have to flood them (Hammi et al., 2014). A DDoS attack against the authentication mechanism will cause important damages such as paralyzing the whole system or allowing non legitimate users to make use of the system.

5) *Message replay attack:* An attacker can record selectively some messages and replay them without modification at a later time, since successful verification of a message does not certify the correctness of the message's sending time. In this way, inaccurate information can be intentionally provided to the objects or to the servers. Message replay attack is usually combined to a message removal attack.

In this work we are interested only in attacks related to authentication service. However, we do not consider other attacks such as DoS/DDOS to make a device out of service or message removal attacks where the attacker drops selectively messages during their transmission.

3.3. *Summary*

Table 1 summarizes the main security requirements and attacks considered during the evaluation of our approach.

Table 1 – Main evaluation criteria.

Evaluation criteria	Part of the evaluation
Mutual authentication	√
Data integrity	√
Communication messages integrity	√
Availability	√
Scalability	√
Non repudiation	√
Pseudonymity	✗
Confidentiality	✗
Sybil attack protection	√
Spoofing attack protection	√
Message substitution protection	√
Message replay protection	√
Message removal protection	✗
Protection against DoS/DDoS of authentication mechanism	√
Protection against DoS/DDoS of devices	✗

4. Related works

Recently, numerous works have been interested in the integration of blockchains into IoT ecosystems. However, very few works were interested in how blockchains can help in meeting IoT security requirements. In this section we survey almost all the works that intend to realize such an integration and show the rarity of works that realize the integration in order to meet security needs.

Christidis and Devetsikiotis (2016) provide a description of how blockchains and smart contracts can be integrated in IoT. They provide a list of the advantages and limits of blockchain use in IoT and conclude that using blockchains and smart contracts facilitates the sharing of IoT services and resources and allows the automation, in a cryptographically verifiable manner, of several existing, time-consuming workflows. However, the authors did not discuss how this integration can help in enhancing the security of IoT. Similarly, Malviya (2016) made a quick study on how blockchain features can secure the IoT. They also described some IoT platforms that rely on blockchains for numerous use case scenarios.

Huh et al. (2017) propose an approach to integrate blockchains to IoT. Their approach relies on the idea of configuring each object by a dedicated smart contract that defines its actions. However, their work still in a very embrionary state. In addition, no details about the considered use cases were provided. Finally, they consider a full anonymity of the used objects, which allows any user, even malicious to make use of the system. Similarly, Bahga and Madisetti (2016) propose a *Blockchain Platform for Industrial Internet of Things (BPI-IoT)*, that enhance the functionality of existing *Cloud-Based Manufacturing (CBM)* platforms, especially towards integrating legacy shop floor equipment into the cloud environment manufacturing. They also propose an architecture for IoT devices to support the proposed platform. However, to secure devices, they rely only on a key-pair, generated by the device itself, without any control. Thus, any user can exploit the system.

Ruta et al. (2017) propose a novel *Service-Oriented Architecture (SOA)* based on a semantic blockchain for registration,

Table 2 – Summary of the related works.

Approach	Identification consideration	Blockchain’s type	Implementation
Ouaddah et al. (2016, 2017)	Yes	Private	No
Hardjono and Smith (2016)	No	Public	No
Xu et al. (2018)	No	Not specified	No
Dorri et al. (2016, 2017)	Yes	Private	Simulation
Ruta et al. (2017)	Yes	Private	Yes
Bahga and Madisetti (2016)	No	Public	Partially
Huh et al. (2017)	No	Public	Yes
Christidis and Devetsikiotis (2016)	Not specified	Not specified	No
Malviya (2016)	Not specified	Not specified	No
Zhang and Wen (2015, 2017)	No	Public	No

discovery, selection and payment. Such operations are implemented as smart contracts, allowing distributed execution and trust. More precisely, their proposal is presented as a framework for *Semantic Web of Things (SWoT)* systems, where a semantic-based resource discovery layer is integrated in a basic blockchain infrastructure in a way that the blockchain adds verifiable records for every single transaction. However, their approach rely on a private blockchain, which restricts its use.

Dorri et al. (2016, 2017) propose a blockchain based architecture for IoT. Their approach relies on three interconnected blockchains: a local blockchain (private) for each use case, a shared blockchain (private) and an overly blockchain (public). Even if the solution resolve the problem of identification, it has multiple shortcomings like (1) each operation engender at least 8 network communications which can flood quickly the whole communication medium in case of high activity of nodes; and (2) the local blockchains are not distributed but centralized which is contrary to its principle because it can limit its power and availability.

Hardjono and Smith (2016) propose *ChainAnchor*, a privacy-preserving method for commissioning an IoT device into a cloud ecosystem. ChainAnchor supports device-owners being remunerated for selling their device sensor-data to service providers, and allow device-owners and service providers to share sensor-data in a privacy-preserving manner. However, Its goal is the full anonymity of the participating devices and is not adapted to numerous IoT use cases where the identification is needed.

Xu et al. (2018) propose *Saphire*, a blockchain-based distributed storage system for large-scale data analytics applications in the IoT. *Saphire* rely on the blockchain’s features to store IoT devices activity in a distributed way. Nonetheless, this work treats only the storage need, but, there was no consideration of IoT’s security requirements.

Ouaddah et al. (2016, 2017) propose *FairAccess*, a blockchain-based access control framework in IoT. More precisely, *FairAccess* works in the same way as *Role-Based Access Control (RBAC)* (Ferraiolo et al., 1995), where the policies are stored in a private blockchain. Thus, the authenticity, the authentication and the updates of policies are always guaranteed. However, it can handle only policy-based compatible systems and use cases, which cannot be applied to numerous IoT contexts.

To summarize, Table 2 describes the studied works. The majority of these works rely on the security mechanism as described in *Bitcoin* or *Ethereum*. In other words, each device uses a key-pair to make use of the system. However, this mechanism ensures a full anonymity, where each participant can exploit the ecosystem, even the malicious ones and cannot ensure the identification, a main requirement in the majority of IoT use cases. The sole works that ensured identification used private blockchains. Nonetheless, this solution suffers from the restriction of the used system, e.g. it is very hard to add a new service or a new device. Furthermore, the majority of the described works are in an embrionary phase, where only the description of the approach is quickly provided and no implementations or simulations were realized.

5. Proposed approach

The main goal of our approach is to create secure virtual zones in IoT environments. Each device must communicate only with devices of its zone, and considers every other device as malicious. We call these zones *bubbles of trust*. Thus, a *bubble of trust* is a zone, where all its members can trust each other. It is protected and inaccessible for non-member devices. In order to achieve such a system we rely on a public blockchain that implements smart contracts. We use a public blockchain instead of a private one in order to make the system open to any user. In other words, relying on a private blockchain, makes our approach applicable only by predefined users[7] and once the system is deployed it will be very difficult if not impossible to add new users, which limits considerably the scalability and the flexibility of the approach (for more details see Section 5.3).

Communications in the system are considered as transactions and must be validated by this blockchain in order to be considered. For example, if a device *A* sends a message to a device *B*, then (1) *A* sends the message to the blockchain, (2) if the blockchain authenticates *A*, it validates the transaction. Finally, (3) *B* can read the message.

In the following we describe the whole lifecycle of a device in an IoT ecosystem that implements *bubbles of trust* approach.

[7] Herein, “users” refers to the participants that own the rights to write on the blockchain.

5.1. *Initialization phase*

Our approach can be applied to a huge number of IoT use cases and does not require special hardware. Nonetheless, it needs an initialization phase. In the latter, a device is designed as *Master* of the *bubble* (It owns a private/public key-pair), which can be considered similar to a certification authority. Any given device can be the *Master*. Besides, each object, that makes part of the system is called *Follower*. Each *Follower* generates an *Elliptic Curve (EC)* private/public key-pair. Then, each *Follower* is provided by a structure called *ticket*, which represents a lightweight certificate of 64 bytes that contains: (1) a *groupID (grpID)*, which represents the *bubble* that the object will be part of, (2) an *objectID (objID)*, which represents the *Follower*'s identifier in the *bubble*, (3) *pubAddr*, which represents the *Follower*'s public address. It represents the first 20 bytes of the *Keccak(SHA-3)* hash (Chang et al., 2012) of the *Follower*'s public key. and (4) a *Signature* structure which represents the *Elliptic Curve Digital Signature Algorithm (ECDSA)* signature using the private key of the *bubble*'s *Master. ECDSA* represent multiple advantages over traditional signature algorithms such as *Rivest Shamir Adleman (RSA)* especially concerning key sizes and signature times and is more adapted to IoT contexts (De Win et al., 1998; Lauter, 2004). The *Signature* covers the *Keccack* hash of the concatenation of the *groupID*, the *objcetID*, and the *pubAddr*. The *ticket* structure is as follows:

```
========================================
GroupID : XX
ObjectID: YY
PubAddr : @@
========================================
Signature (keccakhash(XX||YY||@@))
========================================
```

5.2. *System's functioning*

The scheme in Fig. 2 details our proposed approach and all the phases of the ecosystem's lifecycle. Algorithm 1 describes its different parameters and functions. First, as shown in Fig. 2, phase (A), the connected things can belong to numerous areas (medical, industry, environment, etc.).

The phase (B) represents the initialization phase, where a group identifier (*groupID*) is chosen by the *Master*. Furthermore, each object is provided by a ticked, signed by the *Master*. Once the group is prepared, the step (C) consists of the creation of the *bubble* at the blockchain level. The *Master* sends a transaction that contains the *Master*'s identifier as well as the identifier of the group he wants to create. The blockchain checks the uniqueness of both of the *groupID* and the *Master*'s *objectID*. If the transaction is valid, then the *bubble* is created (eg. *bubble* F9, *bubble* 0A). Since the blockchain is public, any user can create a *bubble*.

After, in Fig. 2, phase (D), the *Followers* in turn, send transactions in order to be associated to their respective *bubbles*. At the blockchain level, the smart contract verifies the uniqueness of the *Follower*'s identifier (*objectID*), then checks the validity of the *Follower*'s *ticket* using the public key of the *bubble*'s *Master*. If one of the conditions is not satisfied, the object cannot be associated to the *bubble*. Algorithm 2 describes the association phase.

Algorithm 1: Parameters and functions definition.

```
parameter:
        bc: Blockchain
        obj: Object
        sender: Object
        receiver: Object
        const failed: State
        define master:  0
        define follower: 1
Function  : ObjIdExists(Integer objId, Blockchain b)
// check if the object identifier is used in
   the blockchain or not
Function  : GrpIdExists(Integer grpId, Blockchain b)
// check if the group identifier is used in
   the blockchain or not
Function  : AddrExists(Integer objAddr, Blockchain
            b)
// check if the object address is used in the
   blockchain or not
Function  : Error()
// returns and error message
```

Algorithm 2: The *smart contract bubbles'* association rules.

```
begin
    if (ObjIdExists (obj.id, bc) = true) then
        return Error ()
    if AddrIdExists (obj.grpId, bc) then
        return Error ()
    if (obj.type = master) then
        if GrpIdExists(obj.grpId, bc) = true then
            return Error ()
    else if (obj.type = follower) then
        if GrpIdExists(obj.grpId, bc) = false then
            return Error ()
        if (bc.TicketVerif(obj.ticket) = failed) then
            return Error ()
    else
        return Error ()
    // Association finished with success
```

Once the first transaction (association request) of a *Follower* is successful, the latter does no longer need to use its *ticket* to authenticate itself (sends it within the exchanged messages). For more details, an example is detailed in Fig. 3. In this example, we describe a *Follower* device called *F* which have been provided a *ticket* signed by the *Master M*. The *ticket* contains a $grpID = XX$, $objID = YY$ and a public key *PubKey_F*. The following operations are described:

1. the first client's transaction represents an association request. The sent message is signed with the *Follower*'s private key, and contains the *Follower*'s *ticket*;
2. when the blockchain receives the transaction it verifies its integrity by verifying the signature with the *Follower*'s

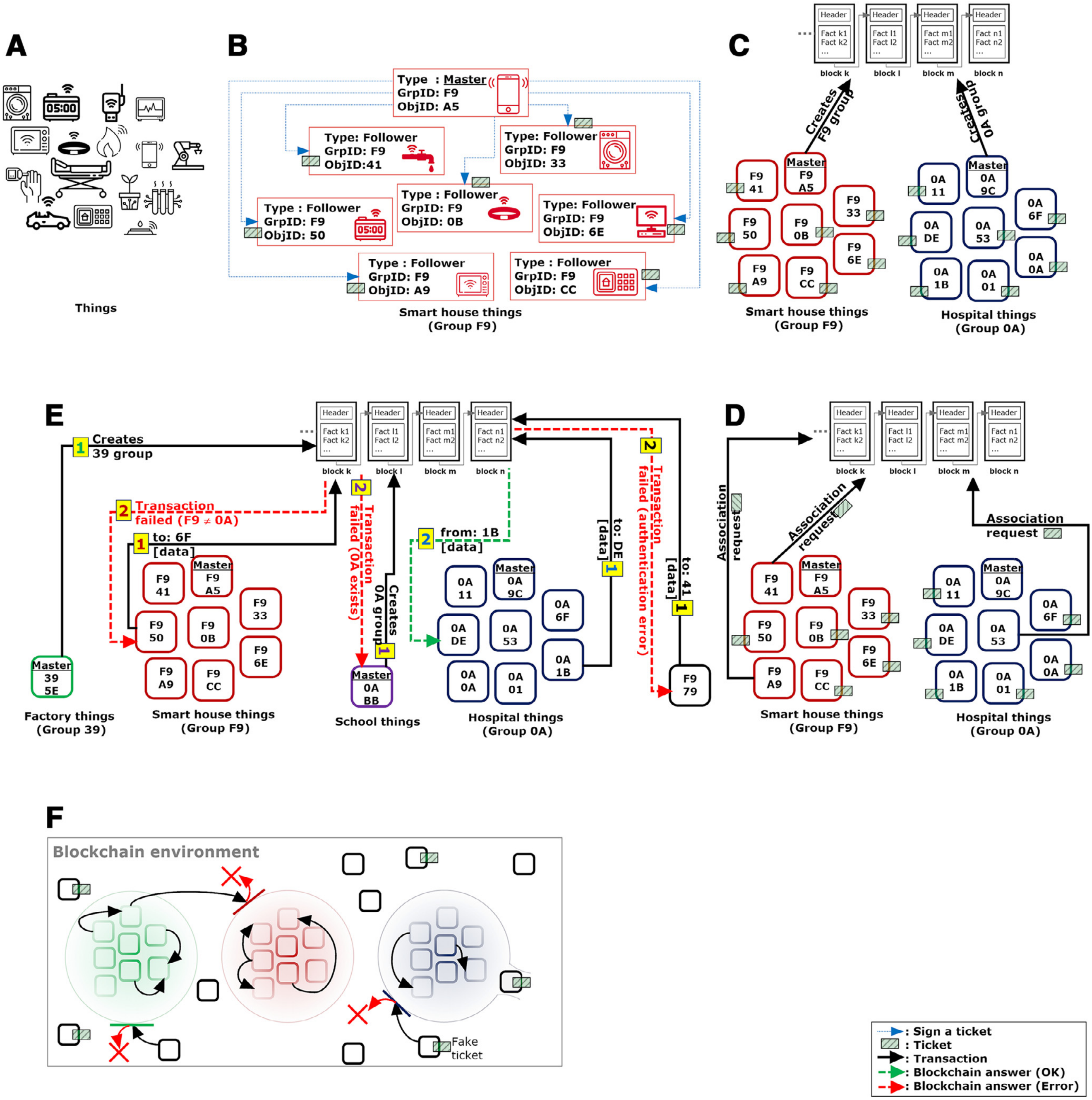


Fig. 2 – *Bubbles of trust* mechanism.

public key. Then, the *Follower*'s *ticket* is verified using the *Master*'s public key, since it represents the entity that signed it;

3. if the *ticket* is valid, then, the blockchain stores an association of its *grpID, objID* and the public key. Thus, it stores (*XX, YY and PubKey_F*);
4. the fourth step describes the case where *F* sends another transaction (transaction *n*) than the association request. This transaction contains: (1) the exchanged data, (2) *XX*, (3) *YY* and (4) the *ECDSA* signature of the concatenation of the previous fields using the *Follower*'s private key;
5. when the blockchain receives the transaction it verifies its integrity by verifying the signature with the *Follower*'s public key;
6. if the signature is valid, the blockchain verifies if the public key used for the transaction's verification is stored and associated to the *grpID* and *objID* sent within the transaction;
7. if the association is stored and is valid then;
8. the device is authenticated with success.

Fig. 2, phase (E), highlights how the blockchain makes the access control upon the objects and transactions. For example, (1) unlike the **Master 5E** which can create the group **39**,

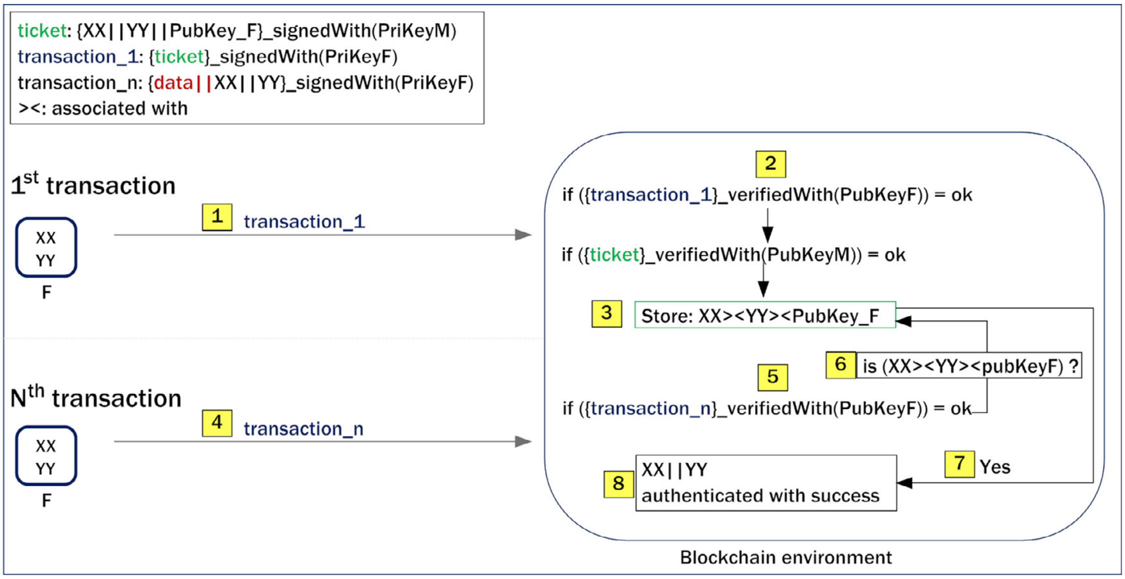

ticket: {XX||YY||PubKey_F}_signedWith(PriKeyM)
transaction_1: {ticket}_signedWith(PriKeyF)
transaction_n: {data||XX||YY}_signedWith(PriKeyF)
><: associated with
1st transaction
1 transaction_1
XX
YY
F
Nth transaction
4 transaction_n
XX
YY
F
2
if ({transaction_1}_verifiedWith(PubKeyF)) = ok
if ({ticket}_verifiedWith(PubKeyM)) = ok
3 Store: XX><YY><PubKey_F
6 is (XX><YY><pubKeyF) ?
5
if ({transaction_n}_verifiedWith(PubKeyF)) = ok
7 Yes
8 XX||YY authenticated with success
Blockchain environment

Fig. 3 – Communications within a *bubble of trust*.

the **Master BB** cannot create the group **0A**, because it already exists. And (2) unlike the accepted message exchanged from **1B** to **DE** which belongs to its group **0A**, the exchanged message from the object **50** belonging to the group **F9** to the object **6F** belonging to the group **0A** is rejected. Algorithm 3 describes the different implemented rules.

Algorithm 3: The *smart contract bubbles'* communication rules.

```
begin
    if (ObjIdExists $(sender.id, bc) = false$
    or (ObjIdExists $(receiver.id, bc) = false)$ then
        return Error ()
    if $(sender.grpId \neq receiver.grpId)$ then
        return Error ()
    if $(bc.SignVerif(sender.msg)) = failed$ then
        return Error ()
    // Secure data exchange finished with
       success
```

Finally, Fig. 2, phase (F) describes a global view of the ecosystem. The certified things (having *tickets*) can be added to their groups at any time. The number of things per group, theoretically, is unlimited, since it relies on completely decentralized architecture. Objects without *tickets* or with fake ones cannot be associated to *bubbles*, thus they cannot communicate with the *bubbles'* nodes. Thanks to the signature of transactions, the object's authentication and the integrity of the exchanged data is ensured. Finally *bubbles* are totally separated, and nodes of different *bubbles* cannot send or receive information of each other.

5.3. *Summary*

Once the smart contract is created and sent to the blockchain through a transaction, it must be validated by miners. If the validation is successful, then, the contract's owner receives an address (e.g. 0X7A62E5DC89FF47A0675EA74E8E445724610AEFEF), that references the contract in the blockchain. This address is public[8] and can be used by any user without any constraints.

To summarize, depending to the object's type, the smart contract's rules are applied as follow:

- *Master*: can create only one *bubble* using a unique group identifier, that does not exist in the blockchain. The *Master*'s role is only signing new tickets. If a *Master* is out of service, it does not disturb the functioning of the bubble (apart from adding new devices).
- *Follower*: (1) is associated only if its *bubble* exists; (2) cannot belong to more than one *bubble*; (3) cannot create a new *bubble*; and (4) its first transaction requires an authentication, using a *ticket* signed by the group's *Master* private key.
- *Both*: (1) the object identifier must be unique, (2) the object's public address and the key-pair must be unique; (3) messages must be exchanged between nodes belonging to the same *bubble*. (4) all the transactions must be signed and verified.

Our approach relies on a public blockchain, which brings enormous advantages:

- blockchains are very resilient decentralized systems, which makes our approach inherit those features;
- known public blockchains such as *Bitcoin* and *Ethereum* are very robust against falsification and alteration, thus, stored information about trustful nodes are reliable;
- public blockchains are autonomous in ensuring their own functioning (validation of blocks, consensus, etc.);

[8] The creator of the contract can advertise and publish this address via a web site for example.

- once a smart contract is deployed, users cannot modify it, since the contract was sent and validated through a transaction.
- using public blockchains instead of private ones makes the system scalable and open to any user. In contrary, if a private blockchain was used, adding a new *bubble* or sending messages (between things), can be performed only by specific validator nodes. This approach limits considerably the flexibility and openness of our solution.

The implementation of our approach was realized using *Ethereum* as blockchain. The choice behind *Ethereum* relies in: (1) it has the second greatest ledger in the world after *Bitcoin* (Nakamoto, 2008) (2) ensures secure transactions based on the *Elliptic Curves Cryptography*, which represents a robust and lightweight signature scheme for constrained devices; (3) uses smart contracts, which facilitates the implementation of the approach; (4) make easy the creation of decentralized applications, called *dApps*; and (5) followed by a big community.

The strong requirement behind the restriction of the communication between *bubbles* resides in the fact that the creation of a new *bubble* is given to any one. Indeed, if the communication between *bubbles* was authorized, if a malicious user creates a new *bubble* in the goal of communicating with a targeted *bubble* (e.g hospital *bubble*), he will succeed.

Knowing that the communication between different use cases can occur in IoT, in our future works, we will evolve our approach in order to support the cooperation between trusted *bubbles*.

6. Evaluation and discussion

6.1. *Context and use case scenarios*

As described above, the power of our proposed approach relies in its suitability to the majority of IoT scenarios, all within ensuring an easy integration of new devices, services and use cases. In this section we evaluate our approach regarding its execution time, energy consumption as well as the financial cost of some use cases. The use cases considered in the financial cost study are:

Smart house: is a house equipped with special structured wiring to enable occupants to remotely control or program a set of automated home electronic devices. In this work we consider a smart home equipped with (1) a device having the function to store a shopping list and to order it online following a certain programmed schedule. (2) a smart fridge, programmed to keep continuously certain food. If this food is consumed, it sends a message to add it to the shopping list. (3) a smart washing machine, which monitors the level of washing powder. If the level reaches a certain threshold, the washing machine sends a message to add it to the shopping list. (3) a remote watering system, which can be triggered by a smart phone. (4) a remote vacuum that can also be triggered by a smart phone.

Waste management: waste management is an increasing problem in urban living. One of the known problems is the garbage-truck route (Fujdiak et al., 2016). Indeed, the garbage-truck needs to pick up all garbage cans even when they are empty, which leads to fuel and time loss, as well as an increase of CO_2 in the ecological footprint. This problem can be more costly if we consider the trucks that go by the plastic, paper, glass or other special material collection, where the garbage cans are mostly underground and have a very complex procedure to empty them, which in numerous cases makes traffic jams that engender more pollution, time and money loss. Luckily, IoT can bring about multiple solutions. By using IoT devices inside the garbage cans, these devices will be connected to a management application, where they send information about can's filling state. Based on this information, the server decides on whether to add the can to the list of cans to be emptied. Next, the management application uses graph theory techniques in order to define a cheaper path for the garbage truck.

Smart factory: the advance in *Cyber Physical Systems (CPS)* introduces the fourth stage of industrialization, commonly known as industry 4.0 or smart factories (Lee et al., 2015). A smart factory is characterized by a self-organized multi-agent system assisted with big data based feedback and coordination (Wang et al., 2016). In other words, in a smart factory, numerous automated machines such as robotic arms, robots and autonomous driverless vehicles, which are equipped with communication devices to communicate between them and also between them and the outside world (customers, partners, other production sites) in order to provide a better organization and scheduling of the production.

Smart road radar: a smart road radar, is a road radar that can be controlled and set up remotely in order to avoid human interventions. Its main function is to measure the speed of road users. If it detects a speed violator, it sends a message containing its photographed license plate and its measured speed to the management system.

In all the described scenarios, multiple sensitive messages and informations are sent in the network. If a malicious user forge, modify or replay (in some cases) these messages, the consequences will be disastrous. Thus, the authentication and integrity of these messages are crucial.

6.2. *Evaluation framework*

In order to evaluate the time and power consumption of our approach, we used three end nodes: 2 identical laptops and 1 Raspberry Pi. One laptop was designed as *Master* and the other end-nodes as *Followers*. Table 3 describes their features. The end-nodes' applications are developed using C++ language.

As described in Section 5, we used *Ethereum* as blockchain. We developed the smart contract that ensures our approach functioning using *Solidity* language (Foundation, 2017). For the interactions between end-nodes and the blockchain, we created a C++ interface that encode/decode data toward/from *Ethereum*.[9] These interactions are realized using *JSON*[10] *Remote Procedure Call (RPC)*. Indeed, we used *TestRPC* (Tes, 2015), which represents a *Ethereum* tool for testing and development purposes and that emulates interactions to the blockchain

[9] The approache's source code including the C++ interface and the smart contract is available on: https://github.com/MohamedTaharHAMMI/BubblesOfTrust-BBTrust-/tree/master.

[10] *JSON* is a standard textual data format.

Table 3 – Experimentation nodes' features.

Node type	CPU architecture	CPU operation mode	CPU max speed	RAM	Operation system
Raspberry PI	armv6l	32-bits	700 MHz	450 MB	Raspbian 4.9.41
HP laptop	x86_64	64-bits	2600 MHz	8 GB	Ubuntu 14.04

without the overheads of running a real *Ethereum* node. An approach deployed using TestRPC, acts exactly in the same way on the public *Ethereum* blockchain. Thus, our approach can be deployed on *Ethereum* without any modifications.

In our study, we was not interested in communication transit time, because it depends on the used network technology and protocols. Our interest concerns the study of our approach's impact on devices, and since we use a public blockchain, where we do not have any control on its functioning, all our results are measured at devices level. The presented results concerns 100 experimentations where we measured:

1. the needed time to prepare an association request,
2. the needed time to prepare a data message,
3. the CPU power consumption to prepare an association request,
4. the CPU power consumption to prepare a data message,
5. the Network Interface Controller (NIC) power consumption to send an association (send request + receive response),
6. the NIC power consumption to send a data message (send message + receive a receipt).

In this work, we are only interested in *Followers* consumptions. Indeed, the *Master* needs only one transaction to create the *bubble*. This transaction is the same as for a *Follower* to associate itself, but without a ticket. Thus, it has smaller size, which leads to less communication costs. Once the *bubble* is created, apart from signing tickets, the *Master* can act as a *Follower* for sending and receiving messages.

6.3. *Evaluation results*

6.3.1. *Security requirements evaluation*

In this section, we discuss how our proposed approach meets the different security requirements presented in Section 3.1 and how it is protected against attacks presented in Section 3.2.3.

Mutual authentication and messages integrity: each object of the ecosystem uses a *ticket* (for the first transaction), which is, as described, a certificate equivalent. The *tickets* are only delivered to legitimate objects during the initialisation phase. All exchanged messages are signed with the private keys associated to those *tickets*, using *ECDSA* algorithm. Thus, signatures ensure the authentication of the device, as well as the integrity of messages. Besides, as explained in Section 2, the blockchain is considered as trustful.

Identification: Each object owns an identity (*objID* associated to a *grpID* and to its public address (generated from its public key)). The trustworthiness of this identity is ensured by the signature of the *Master* in the *ticket*. Each message of this object is signed by its private key which is associated to its identity. Consequently, the system can easily identify it.

Non-repudiation: since the messages are signed using the private key, which is known only by its owner object, its is only this owner who can use it. Thus, it cannot deny the fact of signing a message.

Scalability: our system relies on a public blockchain, which, in turn, relies on a peer-to-peer network. It is known that peer-to-peer networks are one of the best solutions to meet scalability at large scale (Lua et al., 2005).

Sybil attack protection: In our design, each object can have only one identity and each identity can have only one key-pair at a given time. Each communication message must be signed by the private key associated to this identity. Moreover, all identities must be approved by the system, thus, an attacker cannot use fake identities.

Spoofing attack protection: as described for the authentication or the sybil attack protection, an attacker cannot spoof another object's identity, since he needs its private key.

Message substitution protection: since all messages are signed, if an attacker alters or substitutes a messages, he must sign it with a valid private key. However, only trusted objects have been given *tickets* (valid key-pairs) at the initialization phase.

Message replay protection: all messages are considered as transactions. Each transaction has a timestamp and needs a consensus phase in order to be valid. Thus, an attacker cannot reply messages, since the consensus mechanism will reject them. Kshetri (2017) describes how blockchains are robust against replay attacks.

DoS/DDoS protection: the totally decentralized architecture of blockchains makes them robust against DoS/DDoS attacks. Indeed, services are duplicated and distributed over different network nodes. That is to say, even if an attacker manages to block a node, it cannot block all the other nodes. In addition, transactions are costly, which discourage an attacker from spending money by sending a big number of transactions. Furthermore, in some blockchains such as *Ethereum*, the transaction's price is related to the transmitted transaction packet size.

6.3.2. *Time consumption*

The columns 2–5 of Table 4 describes the average and standard deviation of the association request and data messages preparation times, computed over the 100 realized experimentations. The average needed time to realize an association request is 1.56 ms for the laptop. The Raspberry Pi needs more time with 28.03 ms. However in both cases the standard deviation is low which witnesses about the stability of computations. In comparison to these results, data messages sending consumes less time: 0.04 ms for the laptop and 0.82 ms for the Raspberry Pi. The reason of such a difference relies in the

Table 4 – Statistics (Average and Standard Deviation) of the obtained results.

Node type	Assoc time (ms)		Data msg time (ms)		CPU pow assoc (mWatt)		CPU pow data msg (mWatt)		NIC pow assoc (mWatt)		NIC pow data msg (mWatt)	
	Av.	SD	Av.	SD	Av.	SD	Av.	SD	Av.	SD	Av.	SD
Raspberry PI	28.03	0.045	0.82	0.029	64.16	8.19	16.29	1.10	89.24	14.01	31.22	15.11
Laptop	1.56	0.13	0.04	0.001	9.76	2.04	3.35	0.87	16.14	2.69	12.54	4.51

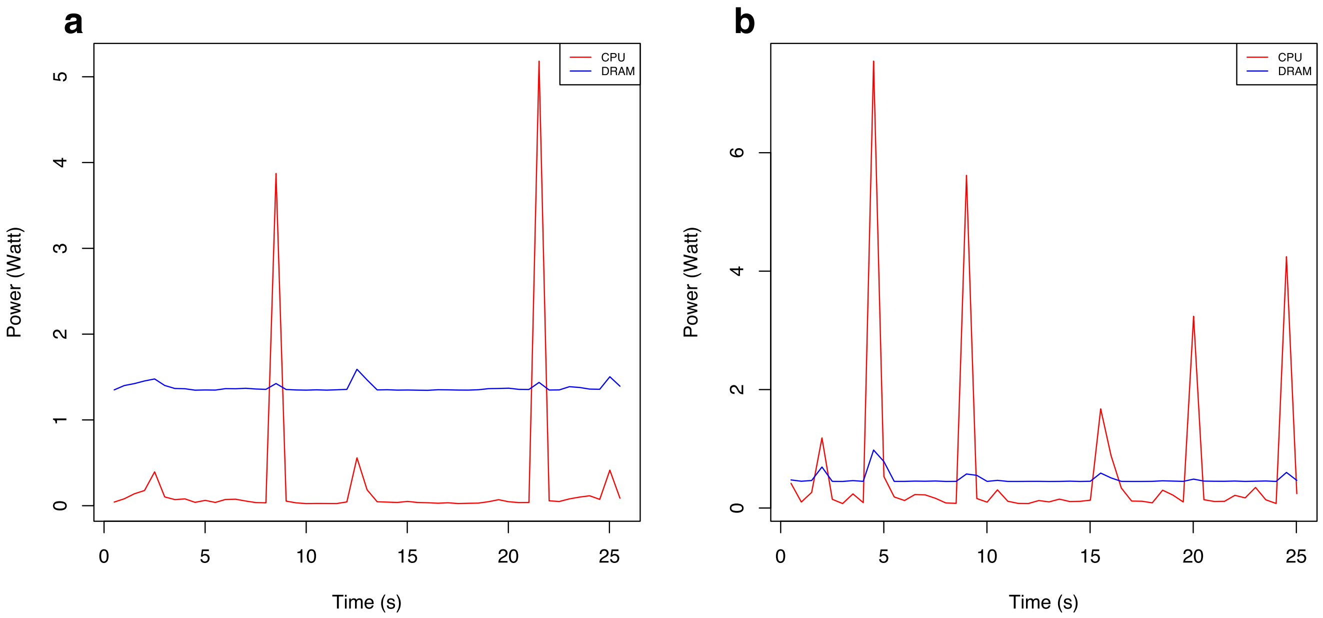


Fig. 4 – Impact of messages processing on: (a) Laptop; (b) Raspberry Pi.

complexity of the association request in comparison to the send operation of a data message. Moreover, in this case also, the standard deviation has a low value.

6.3.3. *Energy consumption*

The columns 6–9 of Table 4 describes the average and standard deviation of the energy consumption needed by the CPU in order to realize (1) the association request and (2) to send a data message. The Raspberry Pi consumes 64.16 mW to realize an association request while only 9.76 mW are consumed by the laptop. For message's sending, the Raspberry Pi needs 16.29 mW while the laptop needs 3.35 mW. These differences, are -as explained above- due to the complexity of association request in comparison to simple message's sending.

The columns 10–13 of Table 4 describes the average and standard deviation of the energy consumption needed by the Network Interface Controller (NIC) in order to realize (1) the association request and (2) to send a data message. The Raspberry Pi requires 89.24 mW to execute an association request while the laptop needs 16.14 mW. For message's sending, the Raspberry Pi consumes 31.22 mW and the laptop 12.54 mW. One can note the big difference between CI and CPU consumptions. This, confirms the well known fact that network communications are the most costly operations for a system.

Fig. 4 describes the impact of messages' sending on the CPU and the Dynamic Random Access Memory (DRAM) energy consumption for the tested devices. The figures describe three phases of the system functioning: (1) an idle phase; (2) the execution of a loop that sends 100 messages where there is a break of 100 ms between each message; and (3) the return to the idle phase. The measures where realized using RAPL[11] measurement tool. [12] Fig. 4a describes the laptop's results where the loop is executed at the 12th second and Fig. 4b describes the Raspberry Pi's results where the loop is executed at the 15th second. In both cases, one can note that the impact of the loop is really negligible. The other existing peaks are related to the operating system activity.

6.3.4. *Financial cost*

In this section we describe the financial cost of the use cases presented in Section 6.1.

Smart house: for the smart house scenario, we consider (1) the smart washing machine sends 1 request per week for adding the washing powder on the shopping list, which requires 1 transaction per week. This transaction involves 1 call operation in order to retrieve the information. (2) the smart

[11] https://github.com/kentcz/rapl-tools.

[12] RAPL measures the total CPU and DRAM activity and cannot isolate the measurements by a selected process.

Table 5 – Estimated financial cost of different IoT use cases (per month).

Use case scenario	Device/case	# Transactions	# Calls	Price (Gas)	Price (ETC)	Price (Euro €)
Smart house	shopping list	8	8	4160	0.0416	0.572
	smart fridge	8	8	4160	0.0416	0.572
	smart washing machine	4	4	2080	0.0208	0.286
	smart watering system	8	8	4160	0.0416	0.572
	smart vacuum	12	12	6240	0.0624	0.858
	Total	**40**	**40**	**20,800**	**0.208**	**2.860**
Smart factory	30 robotic arms	43,200	43,200	22,464,000	224.64	3088.80
	autonomous vehicles	2160	2160	1,123,200	11.232	154.44
	Total	**45,360**	**45,360**	**23,587,200**	**235.872**	**3243.24**
Smart road radar	radar	**58663,08**	**58663,08**	**30504801.6**	**305.048016**	**4194.41**

fridge orders twice a week. (3) the shopping list application makes two orders per week. (4) the smart watering system is used twice a week. Finally (5) the smart vacuum is used 3 times a week. Consequently, 40 transactions and 40 calls are triggered per month.

Smart factory: we consider a smart factory that works 12 h a day. It owns 30 robotic arms (divided into 2 types) and 10 autonomous vehicles. Each final product needs the contribution of two different robotic arms of different types. Once the first one finishes, it prepares the product for the second machine and sends a message to trigger it. When the second machine finishes, it sends a message to the autonomous vehicle, which transports the final product and sends a notification to a management application. Each product needs 30 min in order to be manufactured (15 min for each machine). Consequently, each autonomous machine of the factory sends one transaction each 15 min and triggers 1 call. Thus, 1920 transactions and 1920 calls are used per day (45,360 transactions/call per month).

Smart road radar: for this scenario, we take the case of Paris city. In 2015, radars detected 703,957 speed violations.[13] we consider this number in our study, thus, we consider 58663,08 speed violations per month.

Waste management: as described, certain garbage cans like those used for plastic or glass are often underground and have a complicated procedure to empty them. Generally, there is no gauge to indicate their filling level. Even if the gauge exists, the garbage truck must go through the can to verify it. It is clear that any decrease in cans number (which will occur using gauging objects) will represent considerable savings in the truck's travel time, its fuel consumption and the CO_2 footprint. It is very hard to estimate the financial benefits that our approach can bring, because it depends on the number of can's that the truck must go through, their localization and many other parameters (e.g. the city map and geography).

Table 5 describes the estimated financial cost of the previously presented scenarios, regarding the number of considered transactions for one month. The values described in Table 5 were obtained using Algorithm 4 . For this evaluation we use *Ethereum Classic* cryptocurrency.[14] However, other cryptocurrencies, less or more expansive than ETC can be considered.

Algorithm 4: Bubble of trust cost calculator.

```
const _trans_cost    = 500            // gas
const _call_cost     = 20             // gas
const _gas_in_eth    = 0.00001        // ETH
const _eth_in_euro = 13.75 // Euro

Function Cost (double trans_number, double
call_number)
begin
    return ((trans_number ×_trans_cost)
    +(call_number × _call_cost)) ×_gas_in_eth
    ×_eth_in_euro ;
```

6.3.5. *Comparison with related works*

Since there is no similar approach that provide authentication relying on blockchains, it is very hard to compare our approach with related works. Thus, we compare it with other authentication approaches that rely on an association phase.

Kothmayr et al. (2012) and Hartke and Tschofenig (2014) propose authentication schemes based on *Datagram Transport Layer Security (DTLS)* algorithm (Rescorla and Modadugu, 2012). In the *DTLS*, the association phase (*DTLS* handshake) require at least 5 messages. Moreover, other messages can be added like the Change Cipher Suite Message. Finally, the association phase can include 8 messages. Yeh et al. (2011) proposed an authentication protocol for Wireless Sensor Networks relying Elliptic Curves Cryptography. The association phase requires 5 messages. Besides, the use of a gateway is required, which can duplicate the number of messages. Jan et al. (2014) proposed a robust authentication scheme for IoT. The association phase behind this approach requires 4 messages.

It is known (and proved by tests described in Table 4) that Input/Output operations are the most costly ones. Thus, the less is the number of messages, the less is system's consumption, especially for constrained devices. Compared to the works described above, our approach requires only

[13] http://www.lefigaro.fr/actualite-france/2016/02/20/01016-20160220ARTFIG00011-les-50-radars-qui-flashent-le-plus-en-france.php.

[14] The considered ETC value during the writing of this paper (6 February 2018) is 1 ETC = 13.75 EUR.

2 messages: (1) the transaction sent from the device to the blockchain and (2) the blockchain's response, which makes it less energy and computation consuming if the approaches are implemented on the same hardware. Furthermore, Messages' authentication is realized through *ECDSA* algorithm. *Elliptic Curve Cryptography* is known to be lightweight and well suited for constrained devices (De Win et al., 1998; Lauter, 2004).

However, if we compare the needed time to realize a device association or a message authentication, our approach depends on the used blockchain (e.g. 14 s for Ethereum), while other approaches can realize it in some milliseconds.

7. Open issues

Our approach suffers from three main issues:

Not adapted to real time applications: our approach relies on a public blockchain. In the latter, according to the consensus protocol, the transactions (blocks) are validated each a certain defined period of time (consensus needed time), e.g. 14 s in *Ethereum*. Thus, transactions (messages) sent by devices will be validated only after this period. There are many IoT scenarios where this period is not tolerated. However, this issue can be resolved if a private blockchain is used.

Needs an initialization phase: our approach requires an initialization phase. The latter can be realized on a new or an old device. In both cases, it requires the intervention of the service vendor (initiator). Nonetheless, any user can create its own *bubble*, as long as he creates himself the *Master*.

Evolution of cryptocurrency rate: our approach relies on a public blockchain, which involves costs that depends on the cryptocurrency used by the blockchain system. Despite, we believe that each security service provided needs a cost, as long as it remains lower than the potential damages. Besides, according to studies like Saito and Iwamura (2018) and Mandeng (2018), the evolution of the cryptocurrencies rates will get more stable over time. Even better, *Ethereum* developers and community are working on regulating and stabilizing the amounts of fees related to smart contracts use.[15]

8. Conclusion and future works

IoT and its applications are quickly becoming part of our everyday life. Indeed, its usage is on the rise, which leads to the emergence of many IoT devices and services. Each device must be reachable and produce content that can be retrieved by any authorized user regardless of his location. In many cases, access to these devices and their communication exchanges should be secure.

In this paper, we have proposed an original approach called *bubbles of Trust*, in which secure virtual zones are created, where devices can communicate in a completely secure way. *Bubbles of trust* approach can be applied to numerous IoT contexts, services and scenarios. It relies on a public blockchain, thus, it benefits from all its security properties. Besides, we defined the security requirements that an IoT authentication scheme must ensure and built a threat model.

The evaluation of our approach shows its ability in meeting the requested security requirements as well as its resiliency toward attacks. Furthermore, we provided an extensive study of its time and energy consumptions, where we evaluated different devices.

For future works, we plan to (1) evolve the system in order to allow controlled communication between a chosen set of bubbles; (2) to implement a revocation mechanism for compromised devices; and (3) to study and design a protocol that aims the optimization of the miner's number in a defined system as well as how the selected miners can be placed.

REFERENCES

Alur R, Berger E, Drobnis AW, Fix L, Fu K, Hager GD, Lopresti D, Nahrstedt K, Mynatt E, Patel S, et al. Systems computing challenges in the internet of things. arXiv preprint arXiv:160402980 2016.

Amoroso EG. Fundamentals of computer security technology. Prentice-Hall Inc.; 1994.

Bahga A, Madisetti VK. Blockchain platform for industrial internet of things. J Softw Eng Appl 2016;9(10):533.

Bentov I, Lee C, Mizrahi A, Rosenfeld M. Proof of activity: extending bitcoin's proof of work via proof of stake. ACM SIGMETRICS Perform Eval Rev 2014;42(3):34–7.

Bitcoin. Technical Report. Bitcoin developer guide. Bitcoin; 2017.

Bong D, Philipp A. Securing the smart grid with hardware security modules. ISSE 2012 securing electronic business processes. Springer; 2012. p. 128–36.

Castro M, Liskov B, et al. Practical byzantine fault tolerance. Proceedings of symposium on operating system design and implementation, OSDI; 1999. p. 173–86.

Chang Sj, Perlner R, Burr WE, Turan MS, Kelsey JM, Paul S, Bassham LE. NIST Interagency Report 7896. Third-round report of the sha-3 cryptographic hash algorithm competition. NIST; 2012.

Christidis K, Devetsikiotis M. Blockchains and smart contracts for the internet of things. IEEE Access 2016;4:2292–303.

De Win E, Mister S, Preneel B, Wiener M. On the performance of signature schemes based on elliptic curves. Proceedings of international algorithmic number theory symposium. Springer; 1998. p. 252–66.

Dolev D, Yao A. On the security of public key protocols. IEEE Trans Inf Theory 1983;29(2):198–208.

Dorri A, Kanhere SS, Jurdak R. Blockchain in internet of things: challenges and solutions. arXiv preprint arXiv:160805187 2016.

Dorri A, Kanhere SS, Jurdak R, Gauravaram P. Blockchain for iot security and privacy: the case study of a smart home. Proceedings of the 2017 IEEE international conference on pervasive computing and communications workshops, PerCom Workshops. IEEE; 2017. p. 618–23.

Douceur JR. The sybil attack. Proceedings of international workshop on peer-to-peer systems. Springer; 2002. p. 251–60.

Ethereum community. Ethereum homestead documentation. Online http://wwwethdocsorg/en/latest/indexhtml 2016.

Ethereum community. Ethash. Etherium, wiki https://githubcom/ethereum/wiki/wiki/Ethash 2017.

Ethereum. Technical Report. Ethereum development tutorial. Ethereum; 2017.

Ethereum foundation. Solidity documentation; 2017.

Fast Ethereum RPC client for testing and development. Online Test RPC https://githubcom/ethereumjs/testrpc 2015.

[15] https://smartereum.com/6777/buterin-expresses-concern-over-stabilizing-ethereum/.

Ferraiolo D, Cugini J, Kuhn DR. Role-based access control (rbac): features and motivations. Proceedings of 11th annual computer security application conference; 1995. p. 241–8.

FIPS PUB. FIPS PUB 140–2. security requirements for cryptographic modules. Federal Information Processing Standards Publication; 2001.

Fujdiak R, Masek P, Mlynek P, Misurec J, Olshannikova E. Using genetic algorithm for advanced municipal waste collection in smart city. Proceedings of 2016 10th international symposium on communication systems, networks and digital signal processing, CSNDSP. IEEE; 2016. p. 1–6.

Gartner. Technical Report. Gartner says by 2020, more than half of major new business processes and systems will incorporate some element of the internet of things. Gartner, Inc; 2016.

Hammi B, Khatoun R, Doyen G. A factorial space for a system-based detection of botcloud activity. Proceedings of 2014 6th international conference on new technologies, mobility and security, NTMS. IEEE; 2014. p. 1–5.

Hammi B, Khatoun R, Zeadally S, Fayad A, Khoukhi L. Internet of Things (IoT) technologies for smart cities. IET Netw 2017;7(1):1–13.

Hammi MT, Livolant E, Bellot P, Serhrouchni A, Minet P. A lightweight mutual authentication protocol for the IoT; Springer. p. 3–12.

Hardjono T, Smith N. Cloud-based commissioning of constrained devices using permissioned blockchains. Proceedings of the 2nd ACM international workshop on IoT privacy, trust, and security. ACM; 2016. p. 29–36.

Hartke K, Tschofenig H. A dtls 1.2 profile for the internet of things. draft-ietf-dice-profile-01 (work in progress) 2014.

Hassanalieragh M, Page A, Soyata T, Sharma G, Aktas M, Mateos G, Kantarci B, Andreescu S. Health monitoring and management using internet-of-things (iot) sensing with cloud-based processing: opportunities and challenges. Proceedings of the 2015 IEEE international conference on services computing, SCC. IEEE; 2015. p. 285–92.

Huh S, Cho S, Kim S. Managing iot devices using blockchain platform. Proceedings of the 2017 19th international conference on advanced communication technology, ICACT. IEEE; 2017. p. 464–7.

Jan MA, Nanda P, He X, Tan Z, Liu RP. A robust authentication scheme for observing resources in the internet of things environment. Proceedings of the 2014 IEEE 13th international conference on trust, security and privacy in computing and communications, TrustCom. IEEE; 2014. p. 205–11.

Kothmayr T, Schmitt C, Hu W, Brünig M, Carle G. A dtls based end-to-end security architecture for the internet of things with two-way authentication. Proceedings of the 2012 IEEE 37th conference on local computer networks workshops, LCN workshops. IEEE; 2012. p. 956–63.

Kshetri N. Blockchain's roles in strengthening cybersecurity and protecting privacy. Telecommun Policy 2017;41(10):1027–38.

Lauter K. The advantages of elliptic curve cryptography for wireless security. IEEE Wirel Commun 2004;11(1):62–7.

Lee J, Bagheri B, Kao HA. A cyber-physical systems architecture for industry 4.0-based manufacturing systems. Manuf Lett 2015;3:18–23.

Li X, Jiang P, Chen T, Luo X, Wen Q. A survey on the security of blockchain systems. Future Gener Comput Syst 2017.

Lua EK, Crowcroft J, Pias M, Sharma R, Lim S. A survey and comparison of peer-to-peer overlay network schemes. IEEE Commun Surv Tutor 2005;7(2):72–93.

Ma M, Preum SM, Tarneberg W, Ahmed M, Ruiters M, Stankovic J. Detection of runtime conflicts among services in smart cities. Proceedings of the 2016 IEEE international conference on smart computing, SMARTCOMP. IEEE; 2016. p. 1–10.

Malviya H. How blockchain will defend iot 2016.

Mandeng OJ. Technical Report. Cryptocurrencies, monetary stability and regulation; 2018.

Merkle RC. A digital signature based on a conventional encryption function. Proceedings of the conference on the theory and application of cryptographic techniques. Springer; 1987. p. 369–78.

Mirkovic J, Reiher P. A taxonomy of ddos attack and ddos defense mechanisms. ACM SIGCOMM Comput Commun Rev 2004;34(2):39–53.

Nakamoto S. Bitcoin: a peer-to-peer electronic cash system. 2008.

Ouaddah A, Abou Elkalam A, Ait Ouahman A. Fairaccess: a new blockchain-based access control framework for the internet of things. Secur Commun Netw 2016;9(18):5943–64.

Ouaddah A, Elkalam AA, Ouahman AA. Towards a novel privacy-preserving access control model based on blockchain technology in iot. Europe and MENA cooperation advances in information and communication technologies. Springer; 2017. p. 523–33.

Rescorla E, Modadugu N. RFC 6347: datagram transport layer security version 1.2. Internet Engineering Task Force (IETF); 2012.

Rouse M. Blockchain. http://searchcio.techtarget.com/definition/blockchain; 2015.

Ruta M, Scioscia F, Ieva S, Capurso G, Di Sciascio E. Semantic blockchain to improve scalability in the internet of things. Open J Internet Things 2017;3(1):46–61.

Saito K, Iwamura M. How to make a digital currency on a blockchain stable. arXiv preprint arXiv:180106771 2018:1–15.

TongKe F. Smart agriculture based on cloud computing and iot. J Converg Inf Technol 2013;8(2):8.

Wang S, Wan J, Zhang D, Li D, Zhang C. Towards smart factory for industry 4.0: a self-organized multi-agent system with big data based feedback and coordination. Comput Netw 2016;101:158–68.

Xu Q, Aung KMM, Zhu Y, Yong KL. A blockchain-based storage system for data analytics in the Internet of Things; Springer International Publishing. p. 119–138.

Yang G, Xie L, Mäntysalo M, Zhou X, Pang Z, Da Xu L, Kao-Walter S, Chen Q, Zheng LR. A health-iot platform based on the integration of intelligent packaging, unobtrusive bio-sensor, and intelligent medicine box. IEEE Trans Ind Inform 2014;10(4):2180–91.

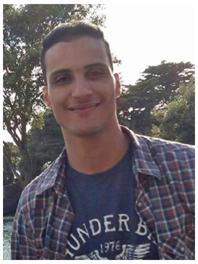
Yeh HL, Chen TH, Liu PC, Kim TH, Wei HW. A secured authentication protocol for wireless sensor networks using elliptic curves cryptography. Sensors 2011;11(5):4767–79.

Zhang Y, Wen J. An iot electric business model based on the protocol of bitcoin. Proceedings of the 2015 18th international conference on intelligence in next generation networks, ICIN. IEEE; 2015. p. 184–91.

Zhang Y, Wen J. The iot electric business model: using blockchain technology for the internet of things. Peer-to-Peer Netw Appl 2017;10(4):983–94.

Zhao Jc, Zhang Jf, Feng Y, Guo Jx. The study and application of the iot technology in agriculture. Proceedings of the 2010 3rd IEEE international conference on computer science and information technology, ICCSIT. IEEE; 2010. p. 462–5.

Zheng Z, Xie S, Dai HN, Wang H. Blockchain challenges and opportunities: a survey. Working Paper 2016.



Mohamed Tahar Hammi is a Ph.D. student at Institut MinesTelecom ParisTech. He received his Master's degree in "computer science and network security" at the University of Paris Descartes in 2015. His current research interests are about securing the Internet of Things.

Badis Hammi is a Post Doc fellow in Institut Mines Telecom ParisTech in France. He received his Master's degree in University of Valenciennes and of Hainaut Cambresis (2011) and his Ph.D. in Troyes University of Technology in 2015. His main research topics of interest are in Security in wireless environments, Intrusion Detection in Wireless Environments and in Cloud Computing.

Patrick Bellot is professor at Telecom ParisTech since 1992 and a member of the LTCI laboratory. During this time, he spent two years in Vietnam to manage the institute and to implement research at lInstitut de la Francophonie pour l Informatique in Hanoi, Vietnam. Before joining Telecom ParisTech, he spent five years in IBM where he was the project leader of the development of AD/Cycle IBM Prolog/2 Program Product. He got a Ph.D. from Paris 6 University and has been awarded Best French Young Researcher in Computer Sciences in 1987. He is a specialist of formal and programming languages. His research topics cover very theoretical matters such as theory of combinators, lambda-calculus and logic. His current research topics includes autonomous and self-healing overlay networks dedicated to smart routing and file storage. He is now currently developing an IoT middleware with robust security based on OPC-UA machine-to-machine communication protocol for industrial automation.

Ahmed Serhrouchni received his Ph.D. in computer science in 1989 and Habilitation Diriger des Recherches in 2010 both from the University Pierre & Marie Curie (UPMC). He is currently a Full Professor with Telecom ParisTech, CNRS-UMR 5141. He is/was leading or involved in many research projects in security Networking in France and Europe. His research focuses on computer network security, security for vehicular networks and security for Industrial Control System.