IoT and Blockchain for Smart Locks

Lucas de Camargo Silva
Computer Science
University of Saskatchewan
Saskatoon, Canada
lucas.camargo@usask.ca

Mayra Samaniego
Computer Science
University of Saskatchewan
Saskatoon, Canada
mayra.samaniego@usask.ca

Ralph Deters
Computer Science
University of Saskatchewan
Saskatoon, Canada
deters@cs.usask.ca

***Abstract*— Community-based online platforms for hospitality services have connected hosts and guests globally. With the increasing popularity of those platforms - e.g., Airbnb - some management issues have attracted the attention of researchers — for instance, granting access to properties and rooms remotely, without requiring hosts and guests to meet in person. Solutions have been proposed - e.g., locks with pin pad and vendor centralized smart locks - but they usually have shortcomings that compromise either security, privacy, or convenience. This research designs and proposes a blockchain-based system for smart door locks to provide the convenience of remote access control management, while security and privacy for both hosts and guests are not compromised. Moreover, to surpass current locks' functionalities, this research proposes a feature that enables the guests to cease the hosts' access to the lock, during their stay. This feature also guarantees to the guest that no one will be able to change that access rule without their explicit approval. The proposed solution integrates Ethereum blockchain as the foundation of the system and uses Infura API as the bridge to connect the IoT infrastructure to the blockchain network. Such architecture alleviates hardware demand from the equipment, which can favor the use of resource-constrained IoT devices. Once Ethereum is used to build the solution, and users are charged fees to perform some actions in the blockchain, the system is evaluated concerning operation costs. Three Ethereum test networks - Ganache, Ropsten, and Rinkeby - were used to run the smart contract and assess the charge to complete significant actions in the system. The results show that the cost to use the smart lock is low, especially if the benefits of security, privacy, and convenience are taken into consideration. Most of the actions yielded fees about cents of a US dollar, where the exception is a one-time payment of USD3.83 - worst-case scenario - to deploy the smart contract – i.e., setup the system.**

Keywords*— *Smart Lock, Internet Of Things, Blockchain, Ethereum, Access Control Management, Community-based Online Platforms, Hospitality Services.

I. INTRODUCTION

Community-based online platforms have enabled new ways to exchange goods and services in a peer-to-peer manner [1][2]. These online platforms have emerged from the rise of the Internet and the development of web systems and mobile applications [3].

According to a Forbes study [4], Airbnb [5] is the leading investment company in the hospitality economic sector nowadays. Airbnb offers a community-based online platform built on sharing, which allows property owners to share their homes or rooms in return for payment [6]. One challenge that emerges from that peer-to-peer hospitality service is how to manage access to the places in a secure, private, and convenient manner. Hosts who use standard key-based door locks must meet in person with the guests to hand them the key. Although convenience is a problem, the primary concern is security, since the guests can make copies of the key to access the place later, for example. If the host cannot meet with the guest for some reason, they may choose to leave the key with a third person or even hide it somewhere unattended, introducing even more risks to security. One possible solution is pin pad door locks, but to be used appropriately they need constant passcode change, which might be inconvenient, if the host lives at the place, or even infeasible if it is not possible to be regularly accessing the lock in place. In addition to that, managing the constant changing passcodes may also be challenging.

The Internet of Things (IoT) [7][8] and the development of pervasive systems [9] make it possible to implement automated solutions to manage access to shared properties. Seo et al. [10] present a door lock that integrates an Arduino micro-controller and an Android device for key sharing. Also, Verma et al. [11] present an RFID door locking solution that enhances access security in real-time and keeps a log of check-in and check-out of users.

Similarly to key-based door locks, these solutions still require in-person interaction between hosts and guests to grant/revoke access to properties and rooms. Moreover, these solutions do not consider guests' privacy as the host will always have open access to the shared space. To deal with these two emerging concerns in the community-based hospitality sector, we propose the integration of the Internet of Things (IoT) and blockchain technology.

Blockchain technology has attracted the attention of researchers in IoT for different purposes [12]. Blockchain has been proposed as a solution to manage the configuration of IoT devices [13][14][15], to detect suspicious behavior on smart spaces [16][17], and to handle data access control management [18][19][20][21]. According to Foroglou et al. [22], the combination of blockchain and IoT contributes to developing smart properties environments, for instance, homes and hotels.

This research presents a smart door lock solution that integrates IoT devices, blockchain smart contracts, and web technology to manage access to properties and rooms in a decentralized manner. This solution eliminates the need for hosts and guests to meet in person, as the blockchain network validates the information and enables a unique configuration of the smart door lock for every new guest. Additionally, this solution guarantees security and privacy for guests since it includes a feature that will not allow anyone else besides the guest to access the place while being rented, not even the host will be able to open the lock. If someone needs to access the place for some reason during that time, the guest must authorize it beforehand.

978-1-7281-2530-5/19/$31.00 ©2019 IEEE

The rest of the paper is organized as follows. Section two introduces smart locks. Section three explains blockchain technology to manage smart door locks. Section four presents the design of the proposed architecture. Section five presents preliminary evaluations. Finally, section six presents conclusions.

II. SMART LOCKS

Along the time, smart door locks have been designed to fulfill different expectations and needs. Roy et al. [23] present a prototype of a smart door lock that integrates an Arduino micro-controller and a camera-augmented mirror, which opens the door every time the user kisses the mirror. The primary purpose of this smart lock solution is to enhance the user's self-esteem.

Park et al. [24] present a smart door lock that manages access to spaces through an RFID authentication process and integrates sensors and actuators to monitor and manage the indoor environment. The primary purpose of this smart lock solution is not only to open the door but also configure the indoor environment through sensors.

Also, there are different smart lock options in the market, for instance, August Smart Lock [25], Friday Lock [26], Gate Smart Lock [27], among others. These smart locks integrate Bluetooth connection to enable the configuration through a mobile app creating a central profile. Although some of those products will provide the functionality to manage the locks remotely, the centralized architecture that they are built on top has some drawbacks. Since the company – or provider - has full control over the platform, a malicious actor – either an employee or a hacker - could access or expose user's data, monitor user activity, maybe even control the user's device, among other threats.

Until now, most of the smart locks' solutions have focused on the physical abstraction to perform specific tasks and monitoring through sensors and actuators. However, more research is needed at the conceptual level to implement new methods of authentication and validation of access permissions with high transparency and reliability, but less personal interaction required.

III. BLOCKCHAIN TO MANAGE SMART LOCKS

Blockchain is the distributed ledger technology that supports Bitcoin cryptocurrency [28]. In general, blockchain records blocks of transactions. These blocks are interconnected through a hash value and validated by the blockchain network (aka miners) through a proof-of-work mechanism [29].

Nowadays, there are blockchain technologies that implement features that the blockchain behind Bitcoin did not have. For instance, Ethereum [30] implements smart contracts. In the context of blockchain, smart contracts are programing functions stored across the blockchain network. Ethereum allows writing smart contracts in three programming languages, solidity, serpent, and LLL.

The integration of smart contracts into smart lock systems provides the following benefits

- High reliability. Before granting access to a property or room, smart contracts will verify the information about guests and renting conditions. Moreover, to change any information, the smart contracts will enforce compliance with the rules, from every user, for every action in the system.
- High security. When the ending renting conditions of a contract are met, the smart lock automatically blocks access to guests without requiring any manual action from the host. Guests will not be able to extend their stay or access the place in the future without the host's consent. Finally, the blockchain also ensures that only authorized people will interact with the smart contract and that no one can change the rules after deployment.

Some works have proposed implementations of smart contracts for IoT. Huh et al. [31] have developed smart contracts to manage the configuration of IoT devices through RSA public keys. Zhang et al. [32] have developed an access control framework for IoT based on smart contracts to manage access control policies and authorization decisions.

Other works have studied blockchain applied to smart home and smart living spaces. Joseph and Navaie [33] proposed a case study, to investigate the capabilities of blockchain when combined with IoT, that enabled tenants to pay for electricity using Ethereum, and the power would be provided automatically to the smart room. One of the proposed future works was to add a smart lock to the smart room. Aung and Tantidham [34] designed a smart home system (SHS) using Ethereum to gather data from sensors and other devices, i.e., temperature sensor and air conditioner. Xu et al. [35] developed an Ethereum-based smart home solution to acquire data from temperature and humidity sensors, with options to act automatically when some condition is met. Although all three smart home works use Ethereum, their approach runs a private network instead of using the public one, which results in different analysis, architecture, and limitations.

Han et al. [36] have proposed a smart door lock system based on blockchain. Their design includes other sensors, i.e., motion sensor, to monitor the surroundings of the house, therefore increasing the amount of information available to the system to make decisions. However, the work does not explain how the design would be implemented and what kind of blockchain would be used. Christidis and Devetsikiotis [37], and Zaparoli et al. [38], mentioned in their work a smart lock called Slock.it that would allow people to pay for a home or car rents through Ethereum and unlock the door after the payment. Nevertheless, the reference is no longer available, and the company Slock.it does not have any reference to this product on their website [39]. Finally, Zaparolli et al. [38] addressed a similar hospitality service problem as proposed in this work, also using the Ethereum blockchain combined with smart locks. The proposed design includes functionalities to manage the whole reservation workflow, from user registration to reservation management. On the other hand, their architecture includes a centralized server, which may lead to security and availability problems. Moreover, the authors do not describe and discuss the smart contract implementations.

Following these examples, this research proposes an Ethereum-based design to manage the operations of smart door locks and evaluates the cost feasibility of using them.

IV. Design of the Proposed Architecture

Using Airbnb, it is possible to rent a single room or a full house directly from the owner with little bureaucracies. However, it brings some challenges to the participants, especially related to management and security.

People who are renting their places using such platforms for accommodation services might not be doing that as a full-time job. It means that while renting an apartment, they still need to go to work, for example, which limits their availability to deal with renting related tasks. In some cases, people may be renting camping or beach houses which they are not even near to. When that happens, in addition to dealing with guests, they still need to manage the maintenance of the place remotely.

One of the first problems that arise from community-based online platforms for hospitality services is to check-in a guest or, in other words, to provide access to the place. Conventional door locks require handling a physical key to the guest. Therefore, someone must be available to deliver the key or open the door when the guest arrives. That exchange is not convenient nor secure since someone with bad intentions could make key copies to access the place in the future. Door locks with pin pads (e.g., [40]) are an alternative, though they suffer from similar security threats and low convenience to continually change the code. Recently, devices known as smart locks were launched, enabling the performance of access control and management of them over the internet. Although convenient, it may contain security threats as well. Most of those solutions follow a centralized architecture, which means that the manufacturer provides the user with a management platform, which they have 100% control over it. If an employee has bad intentions, if the company gets hacked, or even if IT providers encounter some of those problems, someone could potentially gain control over the lock without the owner consent or even without knowledge of it.

One more concern to address from the guest security and privacy perspective is the importance to ensure that, while renting a place, not even the owner could enter at any time during their stay.

The proposed design, therefore, address those management and security problems related to the use of community-based online platforms for hospitality services.

A. User Requirements

A set of high-level user requirements was developed from both the renter and guest perspective to capture the expectations regarding the lock functionalities.

As the renter, I would like to:

- Securely provide guests access to the place being rented (PBR), without the need to meet them personally
- Ensure that no one besides me - or people with my authorization - can provide access to the PBR
- Authorize others to manage the access to the PBR
- Ensure that no one besides me can do authorize others to manage the access to the PBR
- Ensure that past guests neither have nor can provide access to the PBR once their stay is over, without requiring me to go physically there
- Immediately revoke someone's access to the PBR if needed
- Ensure that no one can track when the lock is opened or closed

As the guest, I would like to:

- Ensure that no one besides me has access to the PBR during my stay - be it a single room or a whole place
- Check-in – have access to the PBR - without requiring meeting someone physically
- Ensure that no one can track when the lock is opened or closed

B. Use Cases

Based on the requirements presented, a collection of use cases was developed, and a diagram can be found in Fig. 1.

Setup lock: Configure ownership and provide a device name to start using the lock.

Provide access: Give temporary permission to up to 10 people to open/close the lock - each of them with an expiry date set.

Revoke access: Terminate someone's permission to open/close the lock before expected - meaning anytime previously the expiry date.

Delegate access management: Give management privileges to up to 5 people. Managers can: open/close the lock at any time, excluding when the exclusive access feature - explained below - is turned on; provide and revoke access permissions; request and manage exclusive access permissions.

Revoke delegated management: Terminate a specific manager privilege.

Request exclusive access: Anyone with access permission to the lock can request exclusive access (owner, managers, and guests). If approved, up to the expiry date, that person will be the only one able to open/close the lock. It will temporarily terminate the access permission of the owner and managers.

Manage exclusive access: Owner and managers can approve exclusive access requests and propose changes to an active one. The current exclusive access holder must approve a proposed change. Managers cannot approve their exclusive access requests to avoid a hostile takeover of the lock from the owner. The owner, however, has the power to do so.

C. Systems Architecture

The system is composed of the components shown in Fig. 2. The decision to use a blockchain was due to its decentralized architecture, mainly to overcome the potential problems related to system centralization. Furthermore, Ethereum was chosen as the blockchain platform because of its smart contracts. Using them makes it possible to leverage the blockchain to enforce and secure all access control rules.

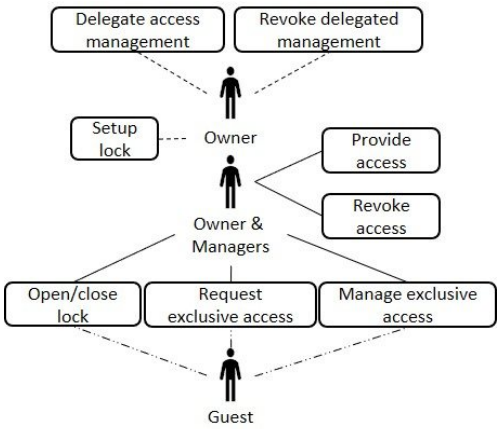

Fig. 1. Use case diagram

For this system, all the code regarding the use cases is written in a unique smart contract. It is important to note that, based on the requirements, the smart contract was carefully developed to hide any information regarding the lock being opened or closed.

A management website was created to hide most of the blockchain complexity from the users, providing them with an interface to manage the lock. The interface can query the smart contracts to retrieve information directly from the page using the web3.js library [41]. However, when the action requires sending a transaction, a blockchain wallet is necessary, e.g., when providing access to anyone.

When someone tries to open/close the lock, the device must check in the blockchain if that person has permission to do it. One possibility to do so would be running an Ethereum node at the lock or the gateway. This approach, however, would require a lot of energy, computing power, and network bandwidth from the device. Infura [42] was chosen since it provides an API to interact with Ethereum so that the solution would be more efficient in terms of resource requirements. In this scenario, the device only needs a simple internet connection, and the ability to handle HTTP requests, which is a lot lighter than running the Ethereum node.

It is out of the scope of this work to discuss the lock and gateway architecture and implementations.

V. Preliminary Evaluation

Transactions on Ethereum are not free and, therefore, such architecture requires its users (owner, managers, and guests) to spend some money to operate the lock. Read-only operations to the smart contract (known as queries), on the other hand, are free of charge. Once this work proposes a solution to be used frequently, it must be evaluated under the critical consideration of operating costs.

A. Testing Environment

Ethereum uses Gas as the measurement unit of how much an operation cost. While the transaction's confirmation time may vary depending on various factors, the Gas calculation follows a fixed set of rules, which do not depend on the network dynamics [43][44]. That characteristic is perfect for this investigation since the results of the experiments will hold valid independently of what might happen to the Ethereum network in the future, which is not valid when performing response time analyses.

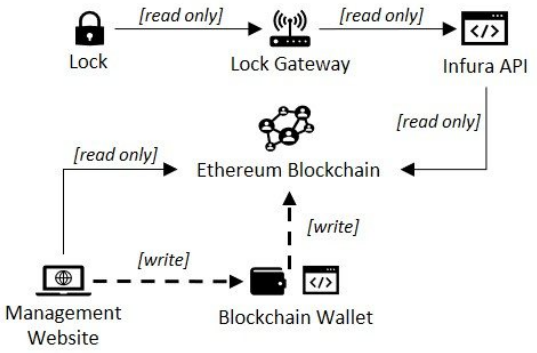

Fig. 2. System architecture

The use of Gas to evaluate the solution also allows the execution of the experiments using test environments (often referred to as *testnets*) – which emulates the Ethereum network behavior - instead of using the main network (often referred to as the *mainnet*). Two examples of such *testnet* networks are Ropsten (ROP) and Rinkeby (RIN) [45], which are commonly used to test and debug smart contracts and Ethereum applications before they are published. While both networks are public, meaning that anyone can join it, it is also possible to run a local private *testnet* using the Ganache software (GAN) [46]. In this work, ROP, RIN, and GAN were used to investigate the operating cost of the lock.

Two tools are used to deploy the smart contract and to interact with it. First, Metamask [47] wallet is used to manage the Ethereum accounts and to provide a connection to all networks. Second, Remix Integrated Development Environment [48] is used to write, test, debug, and deploy the smart contract. Fig. 3 shows the test environment architecture.

B. Gas to USD Conversion

As stated before, the Gas amount was used to evaluate the solution due to its stability over time and the possibility to use the *testnets*. However, it is crucial to understand how that cost translates to USD.

The first step for that conversion was to determine how much ether (ETH – Ethereum currency) is required to pay for a Gas unit. Then, the equivalent ETH can be converted to USD. The cost to buy 1 ETH considered for this work was USD164.11, price on the 25th of April, 2019 [49].

In Ethereum, the user creating a transaction establishes how much ETH he or she is willing to pay as a fee for each unit of Gas. The higher the offer, the more interesting it is for a miner to process that transaction, and the faster it is confirmed. The Ethereum Gas Station website [50] monitors Gas prices in the network and provides real-time recommendations of prices based on the expected transaction confirmation time. The so-called "safe low" price, which may take up to 30 minutes to confirm a transaction, is usually 1 Gwei (1x10-9 ETH) and was used in this work as the best-case scenario for fees, i.e., the cheapest. On the other hand, the "fast" price, which leads to confirmation times below 2 minutes, is more volatile. The recommended price for it on the 26th of April 2019 was 3 Gwei (3x10-9 ETH). Given that, the worst-case scenario for fees, where a fast transaction is desired, was chosen as twice that price, 6 Gwei (6x10-9 ETH).

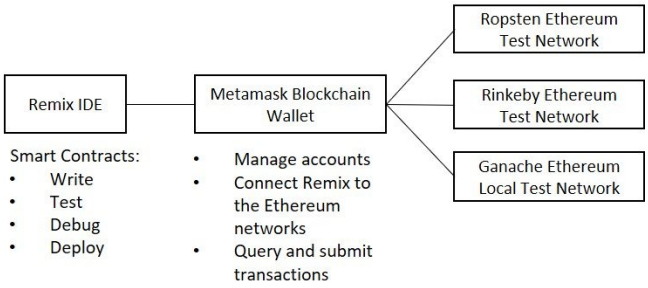

Fig. 3. Test environment architecture

C. *Evaluation Scenarios Selection*

In order to determine the lock's feasibility, the most common and essential operations must be investigated. Those operations are lock/unlock the door, provide permission, get the current permission list, and revoke someone's or everyone's access. In addition to that, the cost related to deploy the smart contract is also considered. The reason is that even being a one-time-only operation, its cost cannot be prohibitive. Delegate access management and exclusive access are useful features but do not happen as usual as those previously mentioned. Moreover, provide and revoke access should give a good baseline for the operating cost.

D. *Smart Contract Creation Scenario*

The cost associated with the creation of the smart contract occurs once, only at its deployment. In the real use case, a lock owner would do this at the first setup of the device and never again. Therefore, to test its cost, the smart contract was deployed three times in each *testnet* - to make sure the value would not differ after the first or second deployment for any reason.

As expected, the three deploys had the same cost in any given network. However, they were not the same between the networks. ROP and GAN yielded a Gas cost of 3897951, meaning USD0.64 in the best-case and USD3.83 in the worst-case prices, following the conversion established earlier. On the other hand, RIN presented a Gas cost 2.7% lower, of 3792351 Gas, which represents USD0.62 in the best-case, and USD3.73 in the worst-case prices.

In summary, the most expensive deploy identified was USD3.83, which is reasonable for a one-time setup operation. Notice that, if the owner does not have an urgency to complete the lock set up, this cost could get as low as USD0.64.

Once the assumption of the same Gas amount between networks did not hold valid, it was necessary to perform all the following tests in all networks and use the highest value as the baseline cost (so to adopt a worst-case scenario price).

E. *Lock and Unlock Scenario*

Regarding a door lock, lock and unlock must be the most performed action. The device owner and managers can do it almost at any time – excluding when the exclusive guest feature is turned on – and any guest can do it at any time up to their permission expiry date.

In the smart contract implementation, both operations are performed by the same function, *unlock*. A call to this function will tell if someone has permission or not to lock or unlock the device. All of this is done as a read-only operation in the smart contract and, as explained before, calls to this function are free of charge.

F. *Get Current Permission List Scenario*

Retrieve the lock permission list, with expiry dates, is probably going to happen often by the management website.

At the smart contract implementation level, complete this action takes two separate calls from the application. First, the list of all guests must be retrieved using the *getAllAccess* function. Second, in order to consult expiry dates, the *getAccessExpireDate* function must be called for each guest. Both steps, however, require read-only operations. Therefore, they are free of charge.

G. *Allow Access Scenario*

Every time the owner wants to give someone permission to lock and unlock the device, he or she will perform this operation, informing who the guest is and when is the desired expiry date. The access will be valid up to the date set. If it is necessary to change the expiry date, a new call to this function is required. As stated before, the smart contract allows up to 10 guests simultaneously with access.

At the implementation level, the *allowAccess* function uses an array of guests of size ten, along with a map structure of guests to expiry dates. Therefore, the cost to permit the first guest might not be the same as for the tenth guest, once more computation is required. The function is coded as after an empty position is found in the array, the guest is saved there, the map is set with the expiry date, and it stops the execution – see Fig. 4. Consequently, the test performed was to provide permission for ten guests, one at a time, and record the cost for each operation. The results found are in Fig. 5, where the x-axis is the guest permission, and each y-axis bring the cost of the operation, one in USD, and the other in Gas units.

For this operation, ROP, GAN, and RIN yielded the same Gas cost for every permission. Using the fast price as the worst-case once again, the most expensive single permission, which is for the tenth guest, costs USD0.074. The cumulative cost of providing all ten permissions was approximately USD0.70, considered feasible. Without the time constraint, this cumulative cost could be as low as USD0.12.

```
/** @dev Give access permission to a given address.
  * Works only if there is less than 10 valid permissions.
  * @param guest Address to give permission.
  * @param expiredate Unix timestamp (in sec - GMT-0600 CST) -
  * When access expires.
  */
function allowAccess(address guest, uint expiredate) public onlyAuthorized {

    // Verify that expiredate is less than 10 digits (valid timestamp in sec)
    require (expiredate<10000000000, "Wrong date format.");
    // It does not make sense to have owner in access_permission
    require (guest != owner, "Cannot add owner as guest.");
    // It does not make sense to have a delegated user in access_permission
    require (!delegated(guest), "Cannot add delegated address as guest.");

    for (uint i=0; i<10; i++) {
        if (access_permission_list[i]==0x0 ||
              access_permission[access_permission_list[i]]<now) {
            access_permission_list[i] = guest;
            access_permission[guest] = expiredate;
            break;
        }
    }
}
```

Fig. 4. *allowAccess* function implementation – Solidity language

H. *Revoke a Single Access Scenario*

While *allowAccess* requires a call for every new guest, *revokeAccess* does not. Once the permission was set with an expiry date, the owner is not required to do anything, unless it is required to terminate the permission before expected. This function is designed to terminate access immediately, meaning that the guest will lose access at that moment. If that is not the case, an *allowAccess* function call is the best option to change the access expiry date.

At the implementation level, differently from the *allowAccess* function, *revokeAccess* does not need to go through the array of guests. The function is designed with the ability to access the map of guests to expiry dates directly. Thus, it was expected the same Gas cost to revoke the access for any guest, independently of the order in which they were given permission. The results are found in Fig. 6, where the x-axis is the guest permission, and each y-axis bring the cost of the operation, one in USD, and the other in Gas units.

Once again, ROP, GAN, and RIN yielded the same Gas cost for every call. It is possible to see that nine out of ten revoke calls had the same Gas amount of 28559, while one of them was of 28495, a negligible difference of 0.22%. One possible explanation for this difference is a change in the size of the call payload to the smart contract. Using the "fast" price and the higher Gas cost as the worst-case scenario, the cost for each permission revoked is USD0.028, which corresponds to 6% of the most expensive *allowAccess* call.

I. *Revoke All Accesses Scenario*

The *revokeAccess* function, as previously explained, is a good option if only a few specific guests must have their permission terminated. However, sometimes a full reset might be needed, where all permissions would be erased, and the owner could start over as new.

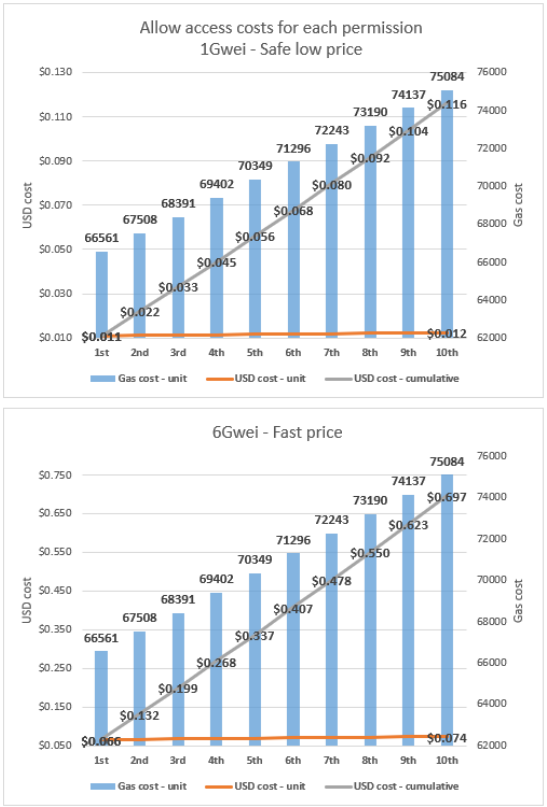

Fig. 5. *allowAccess* costs for safe low and fast prices

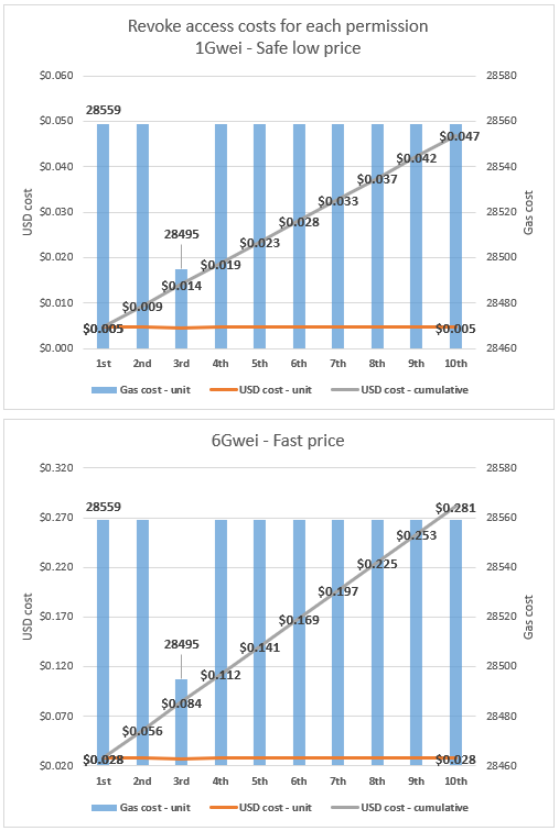

Fig. 6. *revokeAccess* costs for safe low and fast prices

In the smart contract implementation, the *revokeAllAccess* function will erase all the guests from the array and expire their permission date in the map structure. The function is designed with every call doing the same work, meaning that it will go through the entire array of guests independently if the position was used before or not, and expire their access in the map structure. Given that, it was expected the same Gas cost to revoke all the accesses disregard how many guests had permission before. Even so, the cost was investigated in two scenarios, first having a single guest with permission, and second having ten guests. The results are found in Fig. 7, where the x-axis is the number of guests with access permission and each y-axis bring the cost of the operation, one in USD, and the other in Gas units.

First, the Gas cost was different between the two scenarios – one and ten guests - and, moreover, the *testnets* did not agree on the Gas cost between them. For a single guest with permission, ROP and GAN yielded a cost of 129300 Gas, while RIN charged 47700, which represents a significant reduction of 63.1%. For ten guests with access granted, ROP, GAN, and RIN yielded a cost of 64650 Gas. As happened with the smart contract creation, the worst-case cost is chosen. For the "safe low" price, 129300 Gas corresponds to USD0.02, while the "fast" price amounts to USD0.13. The later represents a 92% increase in the most expensive *allowAccess* cost for the first guest, which was USD0.066. Even though the value almost doubles the cost of giving permission, the total cost in dollars is still low. In addition to that, to revoke single access, the *revokeAccess* function is the best option.

Although the two scenarios, for one and ten guests, did not cost the same amount of Gas, they represent the lower

and upper boundary values of *revokeAllAccess*. It means that the cost for a different number of guests with permission must be anywhere between those two values. This hypothesis was put to the test in GAN since it yielded the worst-case cost considered before, and the results are also in Fig. 7. The result cost behavior followed the expected pattern, and the worst-case cost considered before, 129300 Gas, was held valid.

VI. Conclusions

This work addresses management, security, and privacy concerns encountered in community-based online platforms for hospitality services, i.e., Airbnb. In particular, the focus is to avoid the need to meet personally to check-in a guest and to exchange door keys or access codes. Some alternative solutions are available, like door locks with pin pads and some smart locks connected to the internet, but they still have limitations and security threats for both hosts and guests.

The proposed solution uses the Ethereum blockchain to address such concerns, building a system capable of remotely providing access to someone, while preserving the security and privacy for all parties involved. Furthermore, the proposed design includes a feature to enable guests to enforce exclusive access to the lock during their stay, meaning that not even the owner of the device would be able to unlock it during that period. The smart contract in Ethereum blockchain handles all the access rules of the system in a completely decentralized pattern.

Since Ethereum blockchain charges fees for performing some actions, the proposed solution was investigated in terms of operating cost. The use cases expected to be the most common to control the lock were executed using three different Ethereum test networks – Ropsten, Rinkeby, and Ganache. Since the networks presented disagreement to evaluate the cost of some operations, the resulting cost was considered as the most expensive value in those cases.

The results show that the overall cost to use the lock is low, especially if compared with the benefits – convenience, security, and privacy - of using the system. The most expensive action is to set up the lock, a one-time operation, which costs in the worst-case scenario USD3.83. Besides that, all actions investigated cost a few cents of a dollar. Even if their use is needed daily, the amount is too low to be prohibitive. In addition to that, since the solution aims the users of community-based online platforms for accommodation services, guests are paying for the rent, and the operating costs could be included there.

The proposed solution and study have some limitations. First, the use of *testnets* should, in theory, present the same operating costs of the *mainnet*. However, since a few disagreements were found around the Gas cost of some actions, it is desired to do the same evaluation using the main Ethereum network, where the system is meant to run in production. Second, the volatile relationships between the cost of Gas in ETH, and the value of ETH in USD force the constant revaluation of the operating cost at different times. Third, since Ethereum is a public blockchain, meaning that anyone can see its transactions, future studies will investigate how much information is available about the smart contract and its operations.

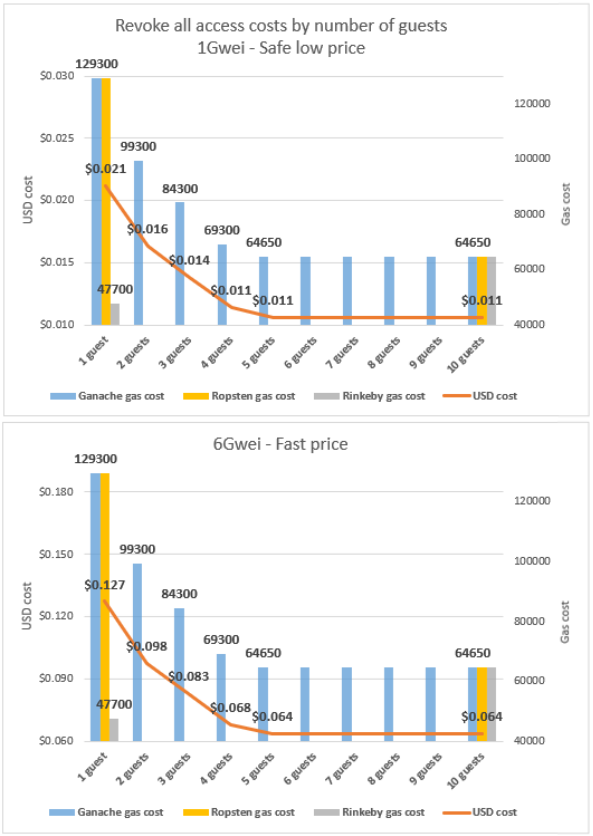

Fig. 7. Revoke all access costs by the number of guests for safe low and fast prices

This work could be applied in the future in many forms. To start with, although the design is focused on the community-based online platforms for hospitality services, hotels, and other accommodation services could benefit from its use as well. The same architecture could be applied to operate gym or school lockers, self-storage facilities, control access in workspaces, and many others. Another possibility would be to expand the use cases and functionalities of the system. For instance, a reservation platform could be integrated to automatically setup guests' permission to locks – taking this action from the host - once a reservation has been completed. Moreover, the same architecture could be used to integrate other IoT devices that do not require real-time control to manage a smart space. For example, a property manager could check and control remotely the lights, or air conditioner, or any other device connected to the internet, of a recently vacant place, or after the maintenance people work there.

References

[1] L. L. P. PricewaterhouseCoopers, "The sharing economy: Consumer intelligence series," 2015.

[2] P. A. Pia A. Albinsson and B. Y. Perera, "From trash to treasure and beyond: the meaning of voluntary disposition Pia," *J. Consum. Behav.*, vol. 8, no. Nov.-Dec., pp. 340–353, 2009.

[3] C. Shah and G. Marchionini, "The Sharing Economy: Why People Participate in Collaborative Consumption," *Int. Rev. Res. Open Distance Learn.*, vol. 14, no. 4, pp. 90–103, 2013.

[4] S. Sharf, "Airbnb Leads $160 Million Investment Into Hospitality Startup Lyric," *Forbes*, 2019. [Online]. Available: https://www.forbes.com/sites/samanthasharf/2019/04/17/airbnb-leads-160-million-investment-into-hospitality-startup-lyric/#70e8c2f04c24. [Accessed: 23-Jul-2019].

[5] "Airbnb." [Online]. Available: https://www.airbnb.ca/. [Accessed: 22-Jul-2019].

[6] "What is Airbnb and how does it work?" [Online]. Available: https://www.airbnb.ca/help/article/2503/what-is-airbnb-and-how-does-it-work. [Accessed: 23-Jul-2019].

[7] M. Samaniego and R. Deters, "Management and Internet of Things," *Procedia Comput. Sci.*, vol. 94, no. MobiSPC, pp. 137–143, 2016.

[8] M. Samaniego and R. Deters, "Zero-trust hierarchical management in IoT," Proc. - 2018 IEEE Int. Congr. Internet Things, ICIOT 2018 - Part 2018 IEEE World Congr. Serv., pp. 88–95, 2018.

[9] S. Xue, R. K. Lomotey, and R. Deters, "Enabling Sensor Data Exchanges in Unstable Mobile Architectures," in *2015 IEEE International Conference on Mobile Services*, 2015, pp. 391–398.

[10] D. G. Seo, H. S. Ko, and Y. D. Deok, "Design and implementation of digital door lock by iot.," in *Transactions on Computing Practices*, 2015, vol. 21, no. 3, pp. 215–222.

[11] G. K. Verma and P. Tripathi, "A Digital Security System with Door Lock System Using RFID Technology," *Int. J. Comput. Appl.*, vol. 5, no. 11, pp. 6–8, 2010.

[12] M. Samaniego and R. Deters, "Blockchain as a Service for IoT," *Proc. - 2016 IEEE Int. Conf. Internet Things; IEEE Green Comput. Commun. IEEE Cyber, Phys. Soc. Comput. IEEE Smart Data, iThings-GreenCom-CPSCom-Smart Data 2016*, pp. 433–436, 2017.

[13] M. Samaniego and R. Deters, "Hosting virtual IoT resources on edge-hosts with blockchain," *Proc. - 2016 16th IEEE Int. Conf. Comput. Inf. Technol. CIT 2016, 2016 6th Int. Symp. Cloud Serv. Comput. IEEE SC2 2016 2016 Int. Symp. Secur. Priv. Soc. Networks Big Data, Soc. 2016*, pp. 116–119, 2017.

[14] M. Samaniego and R. Deters, "Virtual Resources & Blockchain for Configuration Management in IoT," *J. Ubiquitous Syst. Pervasive Networks*, vol. 9, no. 2, pp. 1–13, 2017.

[15] M. Samaniego and R. Deters, "Pushing Software-Defined Blockchain Components onto Edge Hosts," Proc. 52nd Hawaii Int. Conf. Syst. Sci., pp. 7079–7086, 2019.

[16] M. Samaniego and R. Deters, "Detecting Suspicious Transactions in IoT Blockchains for Smart Living Spaces," vol. 11407, Springer International Publishing, 2019, pp. 364–377.

[17] M. Samaniego, C. Espana, and R. Deters, "Smart virtualization for IoT," Proc. - 3rd IEEE Int. Conf. Smart Cloud, SmartCloud 2018, pp. 125–128, 2018.

[18] M. Samaniego, C. Espana, and R. Deters, "Access Control Management for Plant Phenotyping Using Integrated Blockchain," in *International Conference on Machine Learning for Networking*, 2018, pp. 364–377.

[19] M. Samaniego and R. Deters, "Supporting IoT Multi-Tenancy on Edge Devices," Proc. - 2016 IEEE Int. Conf. Internet Things; IEEE Green Comput. Commun. IEEE Cyber, Phys. Soc. Comput. IEEE Smart Data, iThings-GreenCom-CPSCom-Smart Data 2016, vol. 7, pp. 66–73, 2017.

[20] U. U. Uchibeke, K. A. Schneider, S. H. Kassani and R. Deters, "Blockchain Access Control Ecosystem for Big Data Security," 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Halifax, NS, Canada, 2018, pp. 1373-1378.

[21] H. Guo, E. Meamari, and C. C. Shen, "Multi-authority attribute-based access control with smart contract," ACM Int. Conf. Proceeding Ser., vol. Part F148153, pp. 6–11, 2019.

[22] G. Foroglou and A. L. Tsilidou, "Further applications of the blockchain," *Conf. 12th Student Conf. Manag. Sci. Technol. Athens*, no. MAY, pp. 0–8, 2015.

[23] M. Roy, F. Hemmert, and R. Wettach, "Living interfaces: the intimate door lock," *... 3rd Int. Conf. ...*, no. June 2008, pp. 2008–2009, 2009.

[24] Y. T. Park, P. Sthapit, and J. Y. Pyun, "Smart digital door lock for the home automation," *IEEE Reg. 10 Annu. Int. Conf. Proceedings/TENCON*, pp. 1–6, 2009.

[25] "August Smart Keypad." [Online]. Available: https://august.com/products/august-smart-keypad. [Accessed: 25-Jun-2019].

[26] "Friday Lock." [Online]. Available: https://www.fridaylabs.net/views/home-page.html. [Accessed: 25-Jul-2019].

[27] "Gate Smart Lock." [Online]. Available: https://getgate.com/. [Accessed: 25-Jul-2019].

[28] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," *Www.Bitcoin.Org*, p. 9, 2008.

[29] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, "On the Security and Performance of Proof of Work Blockchains," pp. 3–16, 2016.

[30] "Ethereum." [Online]. Available: https://ethereum.org. [Accessed: 26-Jul-2019].

[31] S. Huh, S. Cho, and S. Kim, "Managing IoT devices using blockchain platform," *Int. Conf. Adv. Commun. Technol. ICACT*, pp. 464–467, 2017.

[32] Y. Zhang, S. Kasahara, Y. Shen, X. Jiang, and J. Wan, "Smart contract-based access control for the internet of things," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1594–1605, 2019.

[33] J. Joseph and K. Navaie, "Blockchain Enabled Rooms Implementation For Internet of Things," no. IoTSMS, 2019.

[34] Y. N. Aung and T. Tantidham, "Review of Ethereum: Smart home case study," Proceeding 2017 2nd Int. Conf. Inf. Technol. INCIT 2017, vol. 2018-January, pp. 1–4, 2018.

[35] Q. Xu, Z. He, Z. Li, and M. Xiao, "Building an Ethereum-Based Decentralized Smart Home System," Proc. Int. Conf. Parallel Distrib. Syst. - ICPADS, vol. 2018-December, pp. 1004–1009, 2019.

[36] D. Han, H. Kim, and J. Jang, "Blockchain based smart door lock system," Int. Conf. Inf. Commun. Technol. Converg. ICT Converg. Technol. Lead. Fourth Ind. Revolution, ICTC 2017, vol. 2017-December, pp. 1165–1167, 2017.

[37] K. Christidis and M. Devetsikiotis, "Blockchains and Smart Contracts for the Internet of Things," IEEE Access, vol. 4, pp. 2292–2303, 2016.

[38] M. X. Zaparoli, A. D. de Souza, and A. H. de Oliveira Monteiro, "SmartLock: Access Control Through Smart Contracts and Smart Property," no. Itng, pp. 105–109, 2019.

[39] "Slock.it." [Online]. Available: https://slock.it/. [Accessed: 05-Sep-2019].

[40] "Weiser Smart Locks." [Online]. Available: https://ca.weiserlock.com/en/smart/. [Accessed: 23-Jul-2019].

[41] "web3.js." [Online]. Available: https://web3js.readthedocs.io. [Accessed: 20-Jun-2019].

[42] "Infura." [Online]. Available: https://infura.io/. [Accessed: 20-Jun-2019].

[43] G. Wood, "Ethereum: a secure decentralised generalised transaction ledger," *Ethereum Proj. Yellow Pap.*, pp. 1–32, 2014.

[44] "Ethereum Homestead: account types, gas, and transactions." [Online]. Available: http://www.ethdocs.org/en/latest/contracts-and-transactions/account-types-gas-and-transactions.html. [Accessed: 20-Jun-2019].

[45] "Ethereum: testnets and faucets." [Online]. Available: https://www.ethereum.org/developers/#testnets-and-faucets. [Accessed: 20-Jun-2019].

[46] "Truffle Suite: Ganache." [Online]. Available: https://www.trufflesuite.com/ganache. [Accessed: 20-Jun-2019].

[47] "Metamask." [Online]. Available: https://metamask.io/. [Accessed: 20-Jun-2019].

[48] "Remix web-based IDE." [Online]. Available: https://remix.ethereum.org. [Accessed: 20-Jun-2019].

[49] "Coinbase: Ethereum price (ETH)." [Online]. Available: https://www.coinbase.com/price/ethereum. [Accessed: 25-Apr-2019].

[50] "Ethereum gas station." [Online]. Available: https://ethgasstation.info/. [Accessed: 26-Apr-2019].