Data Privacy Based on IoT Device Behavior Control Using Blockchain

FAIZA LOUKIL, University of Lyon, University Jean Moulin Lyon 3, CNRS, LIRIS, France
CHIRINE GHEDIRA-GUEGAN, University of Lyon, iaelyon school of Management, University Jean Moulin Lyon 3, CNRS, LIRIS, France
KHOULOUD BOUKADI, Mir@cl Laboratory, Sfax University, Tunisia
AÏCHA-NABILA BENHARKAT, University of Lyon, INSALyon, CNRS, LIRIS, France
ELHADJ BENKHELIFA, Staffordshire University, Stoke on Trent, UK

The Internet of Things (IoT) is expected to improve the individuals' quality of life. However, ensuring security and privacy in the IoT context is a non-trivial task due to the low capability of these connected devices. Generally, the IoT device management is based on a centralized entity that validates communication and connection rights. Therefore, this centralized entity can be considered as a single point of failure. Yet, in the case of distributed approaches, it is difficult to delegate the right validation to IoT devices themselves in untrustworthy IoT environments. Fortunately, the blockchain may provide decentralization of overcoming the trust problem while designing a privacy-preserving system. To this end, we propose a novel privacy-preserving IoT device management framework based on the blockchain technology. In the proposed system, the IoT devices are controlled by several smart contracts that validate the connection rights according to the privacy permission settings predefined by the data owners and the stored record array of detected misbehavior of each IoT device. In fact, smart contracts can immediately detect the devices that have vulnerabilities and have been hacked or pose a threat to the IoT network. Therefore, the data owner's privacy is preserved by enforcing the control over the own devices. For validation purposes, we deploy the proposed solution on a private Ethereum blockchain and give the performance evaluation.

CCS Concepts: • **Security and privacy → Privacy-preserving protocols**; **Distributed systems security**; **Privacy protections**;

Additional Key Words and Phrases: Behavior control, blockchain, smart contract, Internet of Things

ACM Reference format:
Faiza Loukil, Chirine Ghedira-Guegan, Khouloud Boukadi, Aïcha-Nabila Benharkat, and Elhadj Benkhelifa. 2020. Data Privacy Based on IoT Device Behavior Control Using Blockchain. *ACM Trans. Internet Technol.* 21, 1, Article 23 (January 2021), 20 pages.
https://doi.org/10.1145/3434776

Authors' addresses: F. Loukil, University of Lyon, Bat. Blaise Pascal, 7 av. Jean Capelle, 69621 Lyon, France; email: faiza.loukil@liris.cnrs.fr; C. Ghedira-Guegan, University of Lyon, iaelyon school of Management, University Jean Moulin Lyon 3, 1C avenue des Fréres Lumiére, 69372 LYON CEDEX 08, France; email: chirine.ghedira-guegan@univ-lyon3.fr; K. Boukadi, Mir@cl Laboratory, Sfax University, FSEGS, Route de Aeoport Km4, BP 1088 Sfax, Tunisie; email: khouloud.boukadi@fsegs.usf.tn; A.-N. Benharkat, University of Lyon, INSALyon, CNRS, LIRIS, Blaise Pascal, 7 av. Jean Capelle, 69621 Lyon, France; email: nabila.benharkat@insa-lyon.fr; E. Benkhelifa, Mellor Building, Staffordshire University, College Road, Stoke-on-Trent ST4 2XE, United Kingdom; email: e.benkhelifa@staffs.ac.uk.

1533-5399/2020/01-ART23 $15.00
https://doi.org/10.1145/3434776

1 INTRODUCTION

The Internet of Things (IoT) connects and shares data collected from smart devices in several domains, such as smart home, smart grid, and healthcare. According to Cisco [6], the number of connected devices is expected to reach 500 billion by 2030. Such a rise will undoubtedly improve the quality of people's lives by providing them with better facilities on various daily applications. However, the low computing capability of the IoT devices may incur security and privacy issues in the IoT systems. Thus, several adversaries can violate data owners' privacy by compromising the existing IoT devices to gain illegal access to sensitive resources.

There is no single universally accepted definition of privacy. For instance, [7] introduced four dimensions to describe privacy. First, the privacy of personal information, which involves the right to control when, where, how, and with whom, the data are shared. The second dimension is the privacy of the personal behavior, which involves the right to keep secret any knowledge of the activities and choices. The third dimension is the privacy of the communication, which involves the person's right to communicate without surveillance, monitoring or censorship. The last dimension is the privacy of the person, which includes the right to control the integrity of the body, including the medical devices. Although the privacy of personal information, behavior, and communication are the most addressed dimensions by privacy laws, the last dimension is very important and therefore, should be considered in the IoT context. Thus, privacy should be preserved by enabling IoT device management instead of trying to preserve it at the consumer's side.

However, traditional IoT device management models are conducted by a centralized entity, such as a cloud server the role of which is to manage the right validation of each IoT device to communicate with other IoT devices. However, these centralized models are not suitable for the IoT domain due to the difficulty of scale and the centralized entity, which can be considered as a single point of failure. In this context, multiple distributed approaches have been proposed to tackle the IoT device management issue with centralized solutions. Therefore, the right validation is performed by the IoT devices rather than by a centralized entity. However, well-known security and privacy techniques tend to be very expensive when running on devices with limited computing capabilities in the IoT domain. Moreover, adversaries can easily compromise IoT devices, as they intrude into the IoT network, and take control of the IoT systems. Thus, such a distributed right validation cannot be trusted due to the resource-constrained IoT devices. Consequently, the IoT device management requires a distributed and trustworthy right validation. However, the blockchain technology has the potential to overcome the aforementioned challenges thanks to its distributed, secure, and private nature however its application in the IoT domain is not straightforward due to the high bandwidth overhead and the delays involved by classic blockchains. To this end, a lightweight blockchain that eliminates the proof-of-work used for mining new blocks into the classic blockchain can be exploited in the IoT context.

Motivated by the drawbacks mentioned above, we focus on the IoT device management to preserve privacy in a private area network using the blockchain technology. The objective of this work is to propose a smart contract-based IoT device management solution that enables the data owner to control the IoT devices first by defining the privacy permission settings about how each device must behave, then by logging the communication between the devices in a private area in a private blockchain, and, finally, by checking the IoT device behavior before making it communicate with other devices.

This article is organized as follows. Section 2 analyses the existing solutions to the problem of preserving privacy in the IoT domain. Then, Section 3 deals with preliminaries, while Section 4 defines the proposed system model. Section 5 presents the proposed privacy-preserving IoT device management framework. Security analysis and performance, which are illustrated by experiments,

are detailed in Sections 6 and 7, respectively. Finally, Section 8 concludes the article and presents some future endeavors.

2 RELATED WORK

There are many researchers who have studied how to preserve privacy in the IoT domain by using the management of the IoT device access right validation. Reading through the related work, we categorized the proposed solutions in three aspects focusing on (i) managing the IoT devices through a centralized cloud server [2], (ii) delegating the access right validation to the requested IoT devices [3, 11–13], and (iii) managing the IoT devices using the blockchain technology [8, 10, 14, 16].

According to Reference [2], current IoT ecosystems rely on centralized and brokered communication models. Thus, all devices are identified, authenticated, and connected through cloud servers that support huge processing and storage capacities. Therefore, the management of the connections between the IoT devices is conducted by one centralized entity through the internet. Despite its high computing capability, a cloud server can turn out to be a single point of failure and disrupt the entire network, especially with the increase of the expected number of connected devices in the years ahead. Moreover, the centralized solutions are not well suited for IoT due to the difficulty of scale and the many-to-one nature of the traffic.

Therefore, to overcome the centralized model issues, some solutions [3, 11–13] were proposed to delegate the access right validation to the requested IoT devices themselves. For instance, Hernandez-Ramos et al. [12] proposed a set of lightweight authentication and authorization mechanisms to embed authentication and authorization functionality on constrained IoT devices. After that, a Distributed Capability-based Access Control (DCapAC) model [11] was proposed, which was directly deployed on resource-constrained devices. Meanwhile, DCapAC was extended to a flexibility trust-aware access control system for IoTs (TACIoT) [3]. The DCapAC allows smart devices to autonomously make decisions on access rights, based on authorization policy, and shows advantages in scalability and interoperability. However, neither the capability revocation management and delegation were discussed, nor the granularity and context-awareness were considered. For their part, Hussein et al. [13] proposed an access control framework using a community-based structure to define the notion of access rights in a distributed IoT environment. However, the IoT devices can be easily compromised due to their limited memory and energy resources and thus cannot be trusted as access right validation entities.

Consequently, an IoT device management requires a distributed and trustworthy right validation. In fact, several solutions [8, 10, 14, 16] have been proposed to address this issue using the blockchain technology. For instance, Dorri et al. [8] proposed a local blockchain with a policy header, which stores access control policies to control all the access requests related to a smart home. The authors proposed a custom, blockchain technology, where the home gateways hold the role of the miners. Such a solution is hard to be deployed, since it requires a "critical mass." As it seems relevant to new IoT solutions, it is worth building on the existing technologies to be compatible with the already available libraries and wallets. For their part, Maesa et al. [16] proposed an approach based on blockchain technology to publish the policies expressing the right to access a resource and allow the distributed transfer of such right among users. According to the authors, the policies and the rights exchanges are publicly visible on the blockchain. Consequently, any user can at any time know the policy paired with a resource and the subjects that currently have the rights to access the resource. Although this solution enables auditability, the blockchain analysis allows adversaries to deduce personal behaviors, habits and preferences. However, in both [8, 16], the blockchain is only served as an immutable storage for access control policies and therefore cannot provide a dynamic right validation according to the behavior of each IoT device. For their

part, Košťál et al. [14] proposed an architecture for the management and monitoring of IoT devices using a private blockchain. Their proposed network has administrators that authenticated in the network by using digital signatures. In fact, the administrators modify the configuration of the devices in the blockchain. After the configuration is added to the blockchain, all the managed devices get informed then, the device applies the downloaded modifications by decrypting them using its private key. However, for smart city networks, Gong et al. [10] proposed a blockchain-based device management framework for efficient device management and firmware updating. Then, the whole management history of each device is stored in the blockchain and the firmware transmission between the vendor and the management node is conducted through a smart contract for security and resilience against an attack.

Unlike all the prior research studies, our work does not embed security and privacy into the IoT devices but instead it moves it to the blockchain network managed under external control using smart contracts. The reason behind the smart contract use is to (i) enforce a common agreement between several untrusted parties without the involvement of a trusted third party, (ii) verify the privacy permission settings predefined according to the IoT device owner's privacy choices before allowing any IoT device to communicate with other devices, and (iii) prevent any malicious intrusion attempts by analyzing the IoT device behaviors to detect any malicious attempt and rapidly block the detected devices.

3 PRELIMINARIES

As mentioned, the proposed solution is based on the blockchain technology and smart contract, which are introduced in this section.

3.1 Blockchain Technology

The blockchain technology is a distributed computing paradigm that successfully overcomes the problem related to the trust of a centralized party. Thus, in a blockchain network, several nodes collaborate among them to secure and maintain a set of shared transaction records in a distributed way without relying on any trusted party. Moreover, specific nodes in the network, which are known as miners, are responsible for collecting transactions in blocks, solving challenging computational puzzles to reach a consensus, and adding the blocks to a distributed ledger known as the blockchain.

The first proposed system based on this technology was Bitcoin [17], which enables users to securely transfer the cryptocurrency (bitcoins) without a centralized regulator. Bitcoin uses a stack-based bytecode scripting language that offers a very limited ability of creating a smart contract with rich logic [15]. Indeed, a smart contract is an executable code hosted in the blockchain, which stores information, processes inputs, and writes outputs thanks to its predefined functions.

Since then, several blockchain-based development platforms have been proposed offering the ability to host/use smart contracts, such as NXT [18], Ethereum [4], and Hyperledger Fabric [1]. For instance, NXT [18] is an open-source blockchain platform that relies entirely on a proof-of-stake consensus protocol. It has barebones support for smart contracts. However, it is not Turing-complete, meaning that only the existing templates can be used and no personalized smart contract can be deployed. Currently, Ethereum [4] is the most popular blockchain platform for the development of smart contracts. It supports advanced and customized smart contracts with the help of Turing-complete virtual machine, called Ethereum virtual machine (EVM). The EVM is the runtime environment for smart contracts where every node in the Ethereum network runs an EVM implementation and executes the same instructions. Moreover, Hyperledger Fabric [1] is an open-source enterprise-grade distributed ledger technology platform, proposed by IBM and supports the smart contracts. The main differences between Ethereum and Hyperledger Fabric smart contracts

are the used programming languages as well as how and by whom the smart contract code is executed.

3.2 Smart Contract

A smart contract is likely to be a class that contains state variables, functions, function modifiers, events, and structures [4]. Besides, it can even call other smart contracts. We represent the smart contract, which is denoted as *SC*, as a tuple that has the following form:

$$SC =< states, functions >.$$

- **States:** they are variables that hold some data or the owner's wallet address (i.e., the address in which the smart contract is deployed). We can distinguish between two state types, namely *constant states*, which can never be changed, and *writable states*, which save states in the blockchain.
- **Functions:** they are pieces of code that can read or modify the states. We can distinguish between two function types, namely *read-only functions*, which do not require *gas*[1] to run and *write functions* that require *gas*, because the state transitions must be encoded in a new block of the blockchain.

Moreover, a smart contract is hosted in the blockchain by invoking its constructor function through a transaction submitted to the blockchain network, then the constructor function is executed, and the final code of the smart contract is stored on the blockchain. Once deployed, the creator of the smart contract got the returned parameters (e.g., contract address), then users can invoke any available smart contract's function by sending a transaction. Based on the immutable blockchain technology concept, smart contracts cannot be modified once added to the blockchain. Once started, all running of the contract is based on its code. No one can affect it, even the creator [5]. The only way to remove the bytecode from Ethereum is by using the self-destruct function. Usually, only the smart contract owner can remove the contract by invoking this function. The remaining cryptocurrency stored at the address of the smart contract is sent to a designated target and then the code is removed from the state, but the contract remains part of the blockchain history [19].

4 SYSTEM MODEL

This section includes both the main goals of the system model and its description.

4.1 System Model Main Goals

Several researchers adopted the blockchain for non-monetary applications, such as managing IoT devices to enhance the data owner's control over the own smart objects. However, applying the blockchain technology to the IoT context is not straightforward, therefore, several challenges need to be addressed. First, the proof-of-work needs to be eliminated to decrease the transaction processing overhead. Indeed, this computationally expensive consensus is important for cryptocurrency to prevent double spending, which is not considered for IoT device management. For this purpose, a private blockchain that restricts who is allowed to participate in the network, can execute the consensus protocol, and maintain that the shared ledger can be used to eliminate the proof-of-work while maintaining most of the classic blockchain security and privacy benefits. Because the network of a private blockchain is usually not exposed to a hostile public internet environment, the requirements on cost of immutability are weaker. Moreover, the blockchain does not have to

[1] gas: it is a unit that measures the amount of computational effort that it will take to execute certain operations.

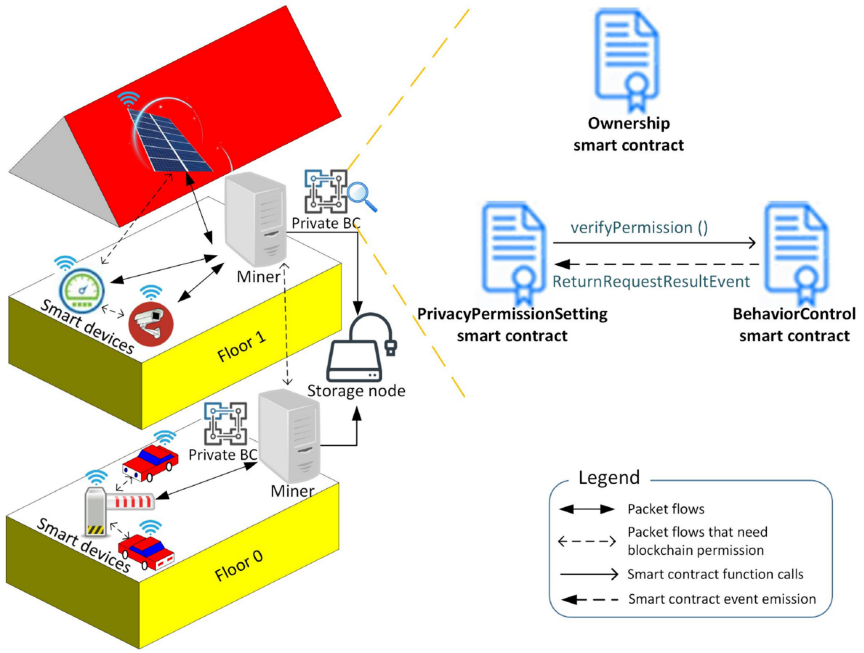


Fig. 1. The System Model: A smart space network, such as a smart home, a hospital, or a smart building that is considered as a private area network. It consists of several smart devices owned by a smart home owner or a hospital manager and monitored by multiple miner nodes through the replicated private BC, which hosts three types of smart contract, namely Ownership, PrivacyPermissionSetting, and BehaviorControl. The role of the private blockchain is to log the communication of the smart devices, while the storage node stores the collected IoT data (e.g., location, energy consumption, etc.) to increase the data owner's privacy.

be guarded by the the proof-of-work thus, the hash chains and replicas owned by different parties are sufficient for ensuring immutability integrity. Second, to enforce the data owner's control over the own IoT devices, a behavior tracking is required to detect any possible misbehavior. In this context, a smart contract can be explored to enforce the data owner's privacy preferences about how the IoT devices must behave.

4.2 System Model Description

To manage the IoT devices, a data owner can introduce some privacy permission settings that define how each IoT device must behave according to the produced data. However, these permission settings need to be enforced to keep the data owner's control over the own devices. We aim at addressing this dilemma by proposing a system model that monitors the IoT devices by allowing or blocking actions according to the device behaviors. Thus, our system model is based on (i) a lightweight blockchain, called Private BC that eliminates the proof-of-work to be supported by the resource-constrained IoT devices, (ii) a smart contract, called Ownership that stores the addresses of the IoT devices possessed by a data owner, (iii) a smart contract, called PrivacyPermissionSetting that verifies the privacy permission settings before allowing any device to communicate with other devices, and (iv) a smart contract, called BehaviorControl that analyzes the IoT device behaviors to detect any malicious attempt and rapidly block the detected devices.

As shown in Figure 1, the system model is a smart space network, such as a smart home, a hospital, or a smart building that is considered as a private area network. It consists of several

smart devices owned by a smart home owner or a hospital manager and monitored by multiple miner nodes through the replicated private BC, which hosts three types of smart contract. The role of the private blockchain is to log the communication of the smart devices, while the storage node stores the collected IoT data (e.g., location, energy consumption, etc.) to increase the data owner's privacy. It consists of four components, namely smart device, miner, private BC, and smart contract.

Therefore, the detailed description of these components is as follows:

- **Smart device:** it is an IoT device equipped with sensing and communication capabilities that allow it to collect environmental data, communicate with other devices, or connect to the Internet. In an IoT environment, we distinguish several devices, such as RFID readers, sensors, actuators, embedded computers, and mobile phones. However, the memory and storage capabilities differ from one device to another. In this work, the smart devices are considered as IoT devices with low memory and storage capabilities. Thus, they only store the relevant information, such as the addresses of the miner nodes and smart contracts, unlike the miner nodes, which store the whole blockchain.
- **Miner:** it is a smart device that does not rely on a battery power, such as a computer or a cloud server. Typically, the miners receive the collected data from smart devices, like sensors and actuators, to remotely analyze them and take appropriate decisions. To enable better control over the smart devices, the data owners require a more flexible way to define permission settings and guarantee their enforcement. For this purpose, each miner node aims at computing complicated treatments, such as logging the communication of IoT devices in a private ledger, monitoring the misbehavior of IoT devices, and blocking the malicious ones. In this work, the miner nodes are considered as smart devices with high memory and storage capabilities. Thus, they process every transaction and store a copy of the entire private blockchain.
- **Private BC:** it is a local private blockchain that enables the data owner to control his/her own smart devices. This blockchain logs only the data owner's smart device communication. The collected IoT data are stored in the storage nodes to increase the data owner privacy. In the private BC, blocks are chained together using the hash of the previous block to keep the blockchain immutable. However, classic blockchains are computationally expensive and involve high bandwidth overhead and delays, which are not suitable for most IoT devices. For this reason, we propose to use a lightweight private blockchain that eliminates the proof-of-work used for mining new blocks into the classic blockchain. To maintain the correctness, the system instantly validates a new block for every new transaction. Therefore, we aim at reducing the block validation processing time by creating a new block for each transaction (i.e., one block only includes one transaction). Thus, each new block will be validated faster while maintaining most of the classic blockchain security and privacy benefits. Moreover, by considering off-chain data storage mechanisms, the system reduces the transaction data size and the redundant storage requirements. The advantages of this approach is that it can reduce both the transaction fee and the chain size. Hence, throughput and scalability of the overall system are enhanced. Furthermore, to enforce the data owner's privacy preferences on how the own IoT devices must behave, the private BC hosts a set of smart contracts.
- **Smart contract:** it can be seen as a published agreement within the blockchain that ensures the compliance of a set of conditions shared between untrusted parties. Therefore, we propose three types of smart contracts, which aim at addressing the data owner's control enforcement over the own smart devices within an environment in which there is no need for participants to be trusted and no centralized or single point of failure is feared.

After defining the system model core components, we detail below the proposed privacy-preserving IoT device management framework.

5 PRIVACY-PRESERVING IOT DEVICE MANAGEMENT FRAMEWORK

According to Reference [7], the privacy of the person is the right to control the integrity of the body and the wearable IoT devices. To guarantee this right, we propose to use smart contracts to define a blockchain-based IoT device management framework, which aims at enforcing the data owner's control over the IoT devices by detecting any possible misbehavior while blocking the detected devices.

In this section, we describe our proposed smart contracts and the main framework functionality.

5.1 Smart Contract Description

To enable the data owner to define and enforce the privacy permission settings, three smart contracts are proposed as shown in Figure 1, namely PrivacyPermissionSetting, Ownership, and BehaviorControl. These contracts enforce the data owner's privacy preferences on how the smart devices must behave according to each data output.

PrivacyPermissionSetting smart contract: It is created by the data owner and hosted on the private BC where each smart device that knows this smart contract address can use it by invoking its defined functions. The PrivacyPermissionSetting smart contract is designed to enable the smart devices to ask for permission before communicating with other devices. This smart contract defines a set of functions, namely: (i) LocalStore function, which enables to verify the smart device permission and locally stores its collected data, (ii) ExternalStore function that verifies if the smart device has the permission to send the collected data to be stored on an external storage node, (iii) Read function that verifies if the smart device has the permission to request data from other internal or external smart devices after verifying the smart device permissions, (iv) Write function that enables a smart device to add and/or modify a requested data collected by other internal or external smart devices if the smart device is permitted, and (v) Monitor function that enables to verify the smart device permission to receive periodic data from another smart device.

Ownership smart contract: it is created by the data owner to store its own IoT device addresses. In fact, for each IoT device, a set of IoT data outputs is added and a PrivacyPermissionSetting smart contract is associated. Moreover, the Ownership smart contract is designed to enforce the data owner's control over the IoT devices and their outputs. It defines a set of functions, namely: (i) addNewIoTDevice function, which enables to add a new IoT device by indicating an IoT device address, an IoT device output, and the address of the associated PrivacyPermissionSetting smart contract, (ii) modifyIoTDevice function, which enables to modify the description of an existing IoT device except for the set of its outputs, (iii) removeIoTDevice function, which enables to remove an existing IoT device, (iv) addIoTDeviceOutput function, which enables to add a new output to an existing IoT device by indicating a description of the new output, (v) modifyIoTDeviceOutput function, which enables to modify the description of an existing IoT device output, and (vi) removeIoTDeviceOutput function, which enables to remove an existing IoT device output from an existing IoT device.

BehaviorControl smart contract: it is created by the data owner to define the privacy settings for each smart device output, verify the permissions before allowing any device to communicate with other devices, and block a smart device access to a resource in case of sending too many requests during a very short time. The BehaviorControl smart contract is designed to rapidly detect any malicious attempt by analyzing the smart device behavior. It defines a set of functions, namely: (i) privacySettingAdd function, which enables to add a new privacy setting to a smart device according to its data output by introducing the action to be handled on the data, its permission, and

its allowed frequency threshold, (ii) privacySettingUpdate function, which enables to modify the permission associated with the action on the output of one smart device, (iii) privacySettingDelete function, which enables to revoke the permission from an existing smart device, (iv) misbehaviorPenalty function, which enables to maintain a record array of the detected misbehavior of an IoT device and compute the duration time of a smart device penalty when any misbehavior is detected, and (v) verifyPermission function, which enables to check the smart device behavior before allowing the transaction sender to access the requested data output.

After introducing the smart contracts, we explain below the main framework functions.

5.2 Framework's Main Functions

Based on the proposed smart contracts, our privacy-preserving IoT device management framework includes the following functions: (i) registering a new smart device to the IoT system, (ii) adding privacy permission settings to each smart device, and (iii) enforcing the privacy permission settings. The dynamic aspect of the framework that relies on these functions is detailed what follows.

5.2.1 Registering a New Smart Device in the IoT System. To facilitate the management of the own smart devices, an owner can register new smart devices through the following steps:

- Step 1: Create (i.e., write and compile) a PrivacyPermissionSetting smart contract.
- Step 2: Send a transaction to deploy the created contract onto the private blockchain.
- Step 3: Create (i.e., write and compile) an Ownership smart contract.
- Step 4: Send a transaction to deploy the created smart contract onto the private blockchain.
- Step 5: Send a transaction to call the function addNewIoTDevice defined in the Ownership smart contract to add a new smart device by indicating an IoT device address, an IoT device output, and the address of the associated PrivacyPermissionSetting smart contract.

5.2.2 Adding Privacy Permission Settings for Each Smart Device. To define the privacy preferences on how each smart device must behave, the owner can define the privacy permission settings for each smart device through the following steps:

- Step 1: Create (i.e., write and compile) a BehaviorControl smart contract.
- Step 2: Send a transaction to call the function privacySettingAdd defined in the BehaviorControl smart contract to add a new privacy permission setting by indicating the action to be handled on the data, its permission, and its allowed frequency threshold, which is the maximum allowed request number in a short time period.

Once added, the new smart device received the blockchain address of the PrivacyPermissionSetting smart contract.

5.2.3 Enforcing the Privacy Permission Settings. To receive the authorization to execute the needed operation, the smart device can communicate with its associated PrivacyPermissionSetting smart contract through the following steps:

- Step 1: Send a transaction to call any function defined in the PrivacyPermissionSetting smart contract to receive the authorization to execute the needed operation by indicating the needed target (e.g., IoT device output) and the requested action to be handled on the target.
- Step 2: Call the verifyPermission function defined in the BehaviorControl smart contract internally by the invoked function in Step 1.
- Step 3: Emit the ReturnRequestResult event with the appropriate decision after verifying the smart device permission and behavior.
- Step 4: Receive the appropriate decision.

As mentioned, each IoT device behavior is tracked to authorize the requested action or detect any possible misbehavior. For this purpose, we introduce the algorithms of both misbehaviorPenalty and verifyPermission functions defined in the BehaviorControl smart contract, which are detailed below in Algorithm 1 and Algorithm 2.

Algorithm 1 aims at computing the duration time penalty when any misbehavior is detected (Line 3) and pushing the received misbehavior into a dynamic array that stores the detected misbehavior records of each smart device (Line 4). The misbehaviorPenalty Algorithm takes as input the subject (i.e., smart device blockchain address), the requested smart device output, the asked action, the misbehavior type, and the time when the misbehavior occurred and returns the computed penalty. In fact, for each subject, a record array of misbehavior, namely *MisbehaviorList* is stored and used to compute the penalty, which is the block duration opposed to the subject in terms of number of minutes, during which the subject cannot invoke any operation using its smart contract. Then, the penalty is computed according to the subject record array of misbehavior and its frequency threshold of invoking a specific action on one device output.

ALGORITHM 1: IoT device misbehavior judge.

Input: *subject*, *deviceOutput*, *action*, *misbehavior*, *time*
Output: *penalty*

1: **Function misbehaviorPenalty**(*subject*, *deviceOutput*, *action*, *misbehavior*, *time*):
2: *length* =MisbehaviorList[*subject*].length + 1
3: *penalty* =*length* / privacySettings[*subject*][*deviceOutput*][*action*].*frequencyThreshold*
4: MisbehaviorList[*subject*].push(Misbehavior(*subject*, *deviceOutput*, *action*, *misbehavior*, *time*, *penalty*))
5: **return** *penalty*
6: **End Function**

Algorithm 1 is used by Algorithm 2 in case of any misbehavior detection. Indeed, Algorithm 2 aims at checking the smart device behavior before allowing it to handle the requested action on the device output. Thus, it is executed each time a smart device invokes a function of its PrivacyPermissionSetting smart contract. The verifyPermission Algorithm takes as input the subject, the device output, the asked action, and the time when the smart contract function is invoked. It also returns the request result and the authorization message. First, a defined privacy permission setting that introduces the action for the couple of one subject and the device output needs to exist. Otherwise, one misbehavior is detected, stored on the subject's record array of misbehavior, while its request is denied (Lines 2–6). If a privacy permission setting exists, then both the subject and the output are verified to see if they are blocked or not (Lines 8–13). In case of unblocking, both the smart device privacy permission setting and the smart device behavior are checked. If the permission is allowed and no misbehavior is detected, then the verifyPermission Algorithm authorizes the action. Otherwise, a penalty is computed, the subject is blocked, and the permission is denied (Lines 38–42). Several misbehavior types can be detected by the verifyPermission Algorithm, such as sending requests to invoke unauthorized action on a device output (Lines 4–6), sending requests during the penalty duration time (Lines 11–13), and sending multiple requests in a short period of time (Lines 20–24). The device output can also be momentarily blocked to be protected from a possible attack when receiving multiple requests from multiple subjects in a short period of time (Lines 28–34).

ALGORITHM 2: IoT device misbehavior detection.

Input: *subject*, *deviceOutput*, *action*, *time*
Output: *requestResult*, *authorizationMessage*

Function verifyPermission(*subject*, *deviceOutput*, *action*, *time*):
privacySetting= privacySettings[*subject*][*deviceOutput*][*action*]
outputSetting= outputSetting[*deviceOutput*]
if (! privacySetting.exists) **then**
  behaviorcheck = false ; *authorizationMessage*= "Wrong subject specified"
  penalty = **misbehaviorPenalty**(*subject*, *deviceOutput*, *action*, "Unauthorized action attempt", *time*)
else
  if (TimeofDeviceOutputUnblock[*deviceOutput*] ≥ *time*) **then**
    authorizationMessage= "Device Output are still blocked"
  else
    if (behaviors[*subject*].TimeofSubjectUnblock ≥ *time*) **then**
      behaviorcheck = false ; *authorizationMessage*= "Subject is still blocked"
      penalty = **misbehaviorPenalty**(*subject*, *deviceOutput*, *action*, "Successive failure", *time*)
    else
      if (privacySetting.permission == "*allow*")) **then**
        privacySettingCheck = true
      end if
      if (*time* - privacySetting.lastRequest ≤ privacySetting.minInterval) **then**
        privacySetting.frequentRequestsNumber++
        if (privacySetting.frequentRequestsNumber ≥ privacySetting.frequencyThreshold) **then**
          behaviorcheck = false ; *authorizationMessage*= "Subject is blocked"
          penalty = **misbehaviorPenalty**(*subject*, *deviceOutput*, *action*,"Too frequent request", *time*)
          behaviors[*subject*].TimeofSubjectUnblock = time + *penalty*
        end if
      end if
      privacySetting.lastRequest = time
      privacySetting.requestResult = (*privacySettingCheck* **and** *behaviorcheck*)
      if (*time* - outputSetting.lastRequest ≤ outputSetting.minInterval) **then**
        outputSetting.frequentRequestsNumber++
        if (outputSetting.frequentRequestsNumber ≥ outputSetting.frequencyThreshold) **then**
          TimeofDeviceOutputUnblock[*deviceOutput*]= *time* + outputSetting.frequencyThreshold
          authorizationMessage= "Data output are blocked"
        end if
      end if
      outputSetting.lastRequest = *time*
    end if
  end if
  if (*privacySettingCheck* **and** *behaviorcheck*) **then**
    authorizationMessage= "Action authorized"
  else if (!*privacySettingCheck* **and** *behaviorcheck*) **then**
    authorizationMessage= "Permission Denied"
  end if
end if
return ((*privacySettingCheck* **and** *behaviorcheck*), *authorizationMessage*)
End Function

6 SECURITY AND PRIVACY ANALYSIS

After detailing the privacy-preserving IoT device management framework, we highlight and analyze in this section both the security and privacy properties.

6.1 Anonymity and Pseudonymity

Each smart device has a blockchain address used to communicate with other devices. Thus, the anonymity aims at tying the smart devices to obfuscate the data owner's habits and personal behaviors.

To break this anonymity, an attacker may try to link anonymous transactions and other available information to find the data owner's real identity. However, to protect itself against such linking attack, the blockchain addresses of all the smart devices are periodically updated. Indeed, by using different pseudonyms, an attacker will be prevented from linking the real world identities and the pseudonyms.

6.2 Authentication and Privacy Permission Setting Control

Each smart device has a blockchain address and a set of privacy permission settings, which define how each smart device must behave, such that where it can store its produced data, with which devices can communicate, and with which frequency per second executes each operation.

To enforce the privacy permission settings, each smart device has a set of permissions that include the authorized operations defined according to the privacy preferences of the data owner.

Moreover, to break up authentication and smart device control, an attacker may take control of one smart device and start to use the predefined functions on the smart contract to attack the network. However, to address this attack, our design employs behavior monitoring that detects smart devices misbehavior thanks to the BehaviorControl smart contract. Moreover, the miner node controls all the transactions in the network. Then, to protect the smart devices from malicious requests, the transactions are filtered and limited to the authorized transactions by the BehaviorControl smart contract. Therefore, the miner node forwards only the requests sent to the devices by the accepted transactions to be executed.

Moreover, only the data owner's blockchain address can update the privacy permission settings of the own smart devices. Thus, the miner nodes execute only the smart contract code but cannot modify it or alter the smart device authorizations.

Furthermore, an attacker may introduce many misbehaving IoT devices to misguide the environment. To address this attack, our design is based on a private blockchain where users are selected and chosen before joining such a private environment. Each user only controls his/her own smart devices. Thus, if a user introduces many misbehaving IoT devices, only his/her devices will be blocked and not the whole system. Besides, such a user risks to be excluded from the private blockchain network if the introduced IoT devices are malicious.

6.3 Availability

Each smart device or IoT resource (i.e., produced data) should be available to legitimate the data owners. The availability means that the target is accessible when it is needed.

To break up the availability, an attacker may take control of one smart device and send multiple transactions to one IoT resource. Then, to protect against such a denial of service attack, the BehaviorControl smart contract hosted on the blockchain detects smart devices misbehavior and blocks their blockchain addresses.

7 EXPERIMENTS AND RESULTS

This section provides experiment details to demonstrate the application of the proposed framework for privacy-preserving device management in the IoT domain. We first introduce the software and hardware used in the study, then define a use case for IoT device management, and evaluate the performance of the proposed blockchain-based solution. Finally, we compare our proposal with the existing ones to evaluate its efficiency.

7.1 Framework Configuration

Ethereum is currently the most common blockchain platform for the development of smart contracts [4]. Hence, we implemented our proposed smart contracts using the Solidity language [19] and deployed it to the Ethereum test network. To deploy a lightweight blockchain, we used Ganache [9], which is a personal blockchain for Ethereum development. Therefore, we created a test system using Truffle development framework [20], which is the most popular development framework for Ethereum, which, among others, generates JavaScript bindings for the smart contract, enables automated smart contract testing, and includes libraries such as web3.js [21] that facilitates the communication between the smart contract and the Ethereum clients. In our experiments, we used the contract events to automate the actions taken by the different nodes. Then, we implemented event callbacks in our testing framework using the web3.js library [21]. All the experiments were conducted on computers with Intel Core i5 CPU (2.30 GHz and 8 GB RAM).

Moreover, we implemented a test system that consists of several nodes, namely 1 data owner, 1 miner node, and 50 smart devices. We associated the smart devices with an Ethereum account to be represented in the network. In the Ethereum account, each node is identified by a blockchain address, which can deploy a smart contract in the blockchain, and invoke a smart contract function by sending a transaction.

7.2 IoT Device Management Use Case

Let Emma be a data owner that had a set of smart devices that help her to follow a healthcare protocol, which consists in practicing some sport activities and eating healthy meals. Then, let the smart devices be a wearable sensor, a smart treadmill, and a smart phone that Emma owns. These smart devices collect her heartbeat, steps, and training duration. Let a tablet be a computer that hosts Emma's Ethereum account as well as the miner node as one personal computer that has a high memory and storage capabilities.

7.2.1 Smart Device Registration. Using her computer tablet, Emma hosted a PrivacyPermissionSetting smart contract then, an Ownership smart contract that includes her own smart device blockchain addresses in the private BC. After that, she created a new transaction that invokes the addNewIoTDevice function defined in the Ownership smart contract by indicating a smart device address, a smart device output, and the address of the associated PrivacyPermissionSetting smart contract. The computer tablet Ethereum account signs this transaction and propagates it to the network to be mined by the miner nodes. In fact, before adding a new smart device, a set of conditions needed to be satisfied. First, only the first sender of the Ownership smart contract constructor (i.e., the smart contract owner) can add a new smart device. Second, the smart device address cannot be added if it already exists. In this case, the modifyIoTDevice function can be used to update the smart device permissions. Third, the PrivacyPermissionSetting smart contract needs to be already published in the blockchain. Then, when all the conditions are satisfied, a new smart device is added to the Ownership smart contract and the transaction is added to the private BC. After that, the new smart device can communicate with the rest of the network using its published PrivacyPermissionSetting smart contract.

Table 1. Privacy Permission Setting Verification Conformance Checking Results

Request Type	Request Number	Correctness
Conforming	1	100%
Not Conforming	29	100%

7.2.2 Privacy Permission Setting Definition. To manage the own smart devices, Emma first deployed the BehaviorControl smart contract using her tablet computer. Second, she defined the privacy permission setting of each smart device using the privacySettingAdd function. For instance, Emma allowed (i) the wearable device to locally store her heartbeat on her personal computer, (ii) the smart treadmill to collect her steps and monitor her heartbeat, and (iii) the personal computer to externalize the collected data to the hospital server. Indeed, during the training, the wearable device collected Emma's vital parameters and sent them to her personal computer that when it received Emma's sensitive data could send them to the hospital to be stored on Emma's medical base, which is regularly checked by her doctor. Moreover, these stored data are analyzed to propose personalized recommendations for data owners. Hence, a need for a break or water notifications can be sent to Emma when necessary.

7.2.3 Privacy Permission Setting Verification. Once each smart device receives the blockchain address of its PrivacyPermissionSetting smart contract, it invokes the appropriate function to be authorized to execute the needed operation. For instance, the wearable device invokes the LocalStore function defined on the PrivacyPermissionSetting smart contract to be able to store the produced heartbeat data on Emma's personal computer. Indeed, to enforce the privacy permission setting of the wearable device, the LocalStore function calls the verifyPermission function defined on the BehaviorControl smart contract. Thus, the verifyPermission function first verifies the smart device authorizations, then analyzes the smart device's behavior, and finally emits the ReturnRequestResult event with the appropriate decision.

To check the conformance of the privacy permission setting verification step, we conducted an experiment, which consists in adding a permission only for the first smart device and sending the same request by 30 smart device blockchain addresses. Then, we randomly generated conforming and not conforming requests by invoking the LocalStore function by several smart devices blockchain addresses (30 in our case). Figure 2 depicts the results of this experiment during the privacy permission setting verification. As expected, all the requests are correctly executed but only the first smart device is authorized to execute the requested action. The rest of the smart devices received *"Wrong subject specified"* as message from the ReturnRequestResult event.

We summarized the conformance checking results in Table 1. For both request types (i.e., conforming and not conforming), the obtained correctness is one hundred percent. Indeed, the proposed smart contracts ensure the conformance of the defined privacy permission settings.

7.2.4 Privacy Permission Setting Violation Attempt Detection. Let a Denial of Service be an attack in which an attacker sends a lot of transactions to the same target in a very short time. In this sense, we conducted two experiments to simulate this kind of attack. The first experiment consisted in sending many transactions to the same target using one blockchain address. The second one consists in sending a great number of transactions to the same target using several blockchain addresses.

Figure 3 shows the result of the first experiment during the privacy permission setting violation attempts. Let a wearable sensor that sends several access requests to the heartbeat resource using its blockchain address. Then, the BehaviorControl smart contract first, authorizes the action then,

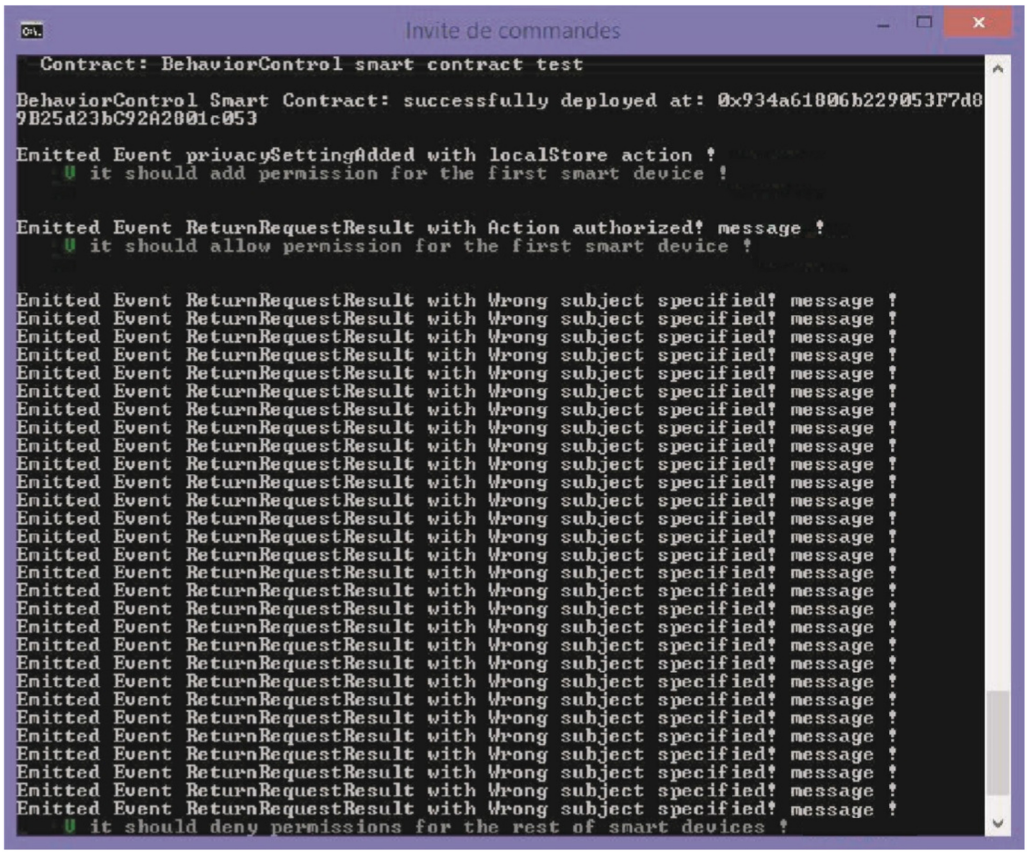


Fig. 2. Blockchain-based smart home test system screen-shot in case of privacy permission setting verification.

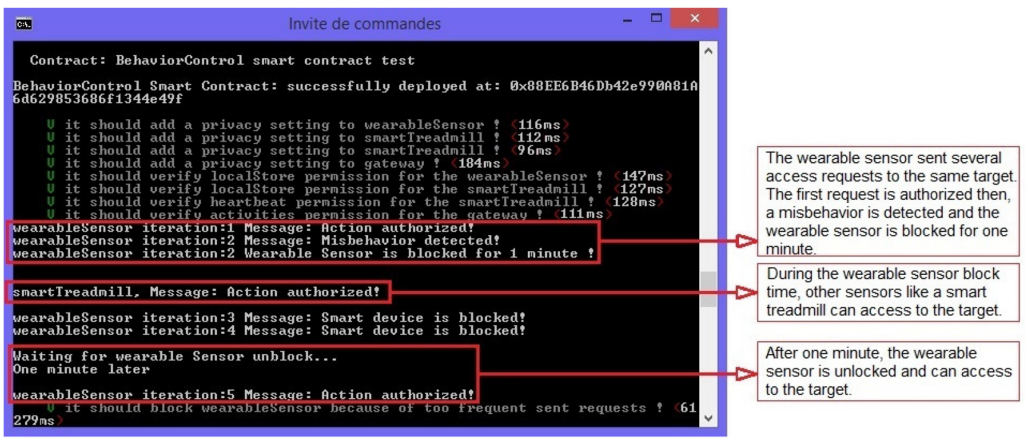


Fig. 3. Blockchain-based smart home test system screen-shot: The case of a denial of service from one blockchain address.

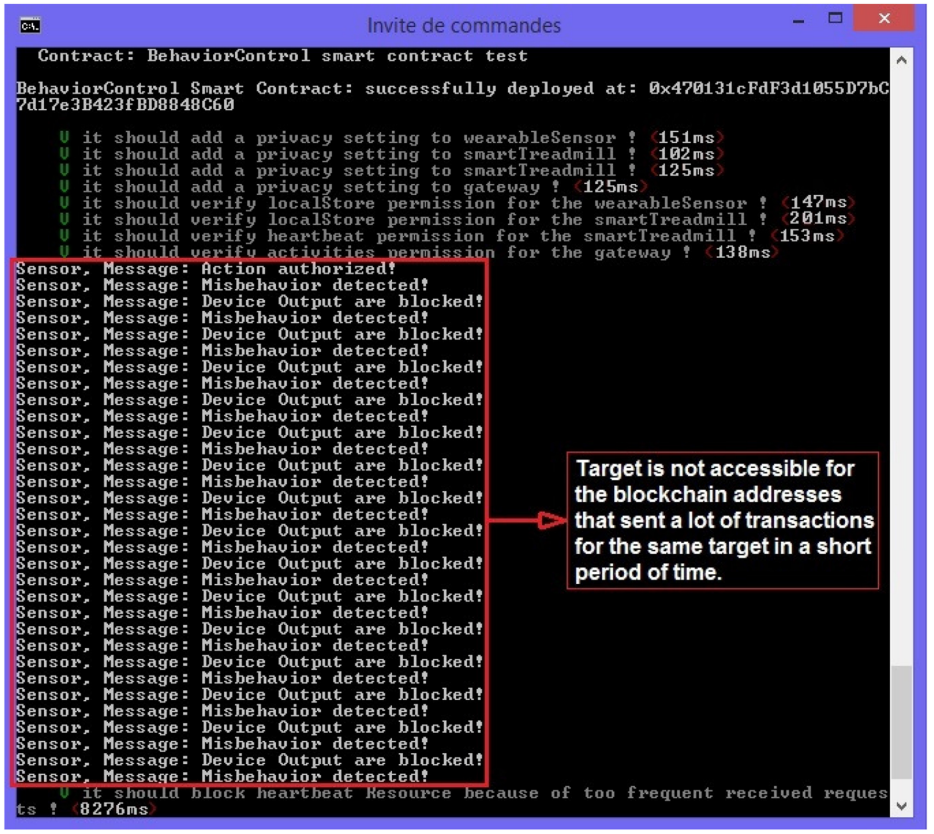


Fig. 4. Blockchain-based smart home test system screen-shot: The case of denial of service from several blockchain addresses.

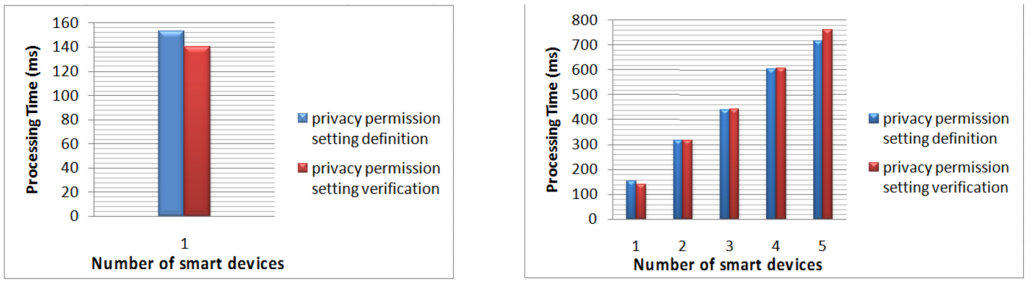
it detects the misbehavior, and blocks the address for a few minutes. After that, the penalty (i.e., the block duration) is computed according to the detected misbehavior number in the past. During the blocking time, the wearable sensor cannot access the heartbeat resource, whereas other sensor, like the smart treadmill, can access to it.

Figure 4 presents the result of the second experiment. Let several blockchain addresses send several access requests to one target, such that heartbeat resource in our example. The BehaviorControl smart contract detects this misbehavior and blocks the access to that target for a few minutes to protect it.

7.3 Performance Evaluation

In this section, the proposed system performance is evaluated in terms of computation time cost and scalability overhead.

7.3.1 Computation Time Cost. To evaluate the performance of our solution, we conducted an experiment to compute the processing time needed by one miner node to validate a privacy permission setting definition transaction that invokes the privacySettingAdd function and a privacy permission setting verification transaction that invokes the verifyPermission function defined on the BehaviorControl smart contract. First, we conducted an experiment to measure the processing time of invoking both privacySettingAdd and verifyPermission functions. Figure 5(a) depicts the



(a) Computational cost for one smart device (b) Average computational cost for five smart devices

Fig. 5. Average computational cost of privacy permission setting definition and verification of smart devices.

computational cost of the two functions for one smart device. Only 150 ms are needed to add a new privacy permission setting or verify the smart device behavior for one smart device. We can also observe that the processing time of invoking the privacySettingAdd function is higher than the processing time of invoking the verifyPermission function. This can be explained by the necessity to initialize a new state on the smart contract when invoking the privacySettingAdd function with a lot of information, such as the data output, the action to be handled on the data, its permission, and its allowed frequency threshold.

After that, we conducted the same experiment while increasing the number of smart devices managed by one miner node. Figure 5(b) depicts the computational cost of the two functions while increasing the smart device number from 1 to 5. The processing time varies from 150 to 750 ms. We observe that the processing time is equal to the processing time for one smart device multiply by the smart device number. Thus, the more the smart device number increases, the more the miner's computing capabilities are required to reduce the processing time.

7.3.2 Scalability Overhead. To evaluate the scalability of both the privacySettingAdd and verifyPermission functions, we made several tests while increasing the number of the managed smart devices by the miners from 1 to 50. Moreover, we ran the simulation for 60 seconds during which a total of 554 transactions are created. Figure 6 shows the average of 10 runs of the simulation. We also observed that the processing time increases with the number of smart devices, which ranges from 100 to 8,000 ms. Therefore, one miner node can manage 50 smart devices in about 8 seconds, which is a short delay time while improving the data owner's control over the own smart devices. It is worth noting that in the case of increasing the number of smart devices in the system, it is recommended to increase the number of miners to reduce the processing time. Moreover, by considering off-chain data storage mechanisms, the IoT produced data are stored in off-chain databases using storage nodes. This reduces the transaction data size and increases the number of transactions that can be accommodated within the block. Hence, the throughput and the scalability of the overall system are enhanced.

7.4 Comparative Study Analysis

In this section, we introduce a comparative study analysis by comparing our proposed system to the existing privacy-preserving approaches in the IoT domain in Table 2. Four axes are simultaneously used to qualify the state-of-the-art, namely the smart contract, IoT data privacy, permission updating, and misbehavior judging.

Being different from the above proposals, the proposed model used smart contracts for self-execution policies. Moreover, to increase the privacy of the data owners, our model stored the

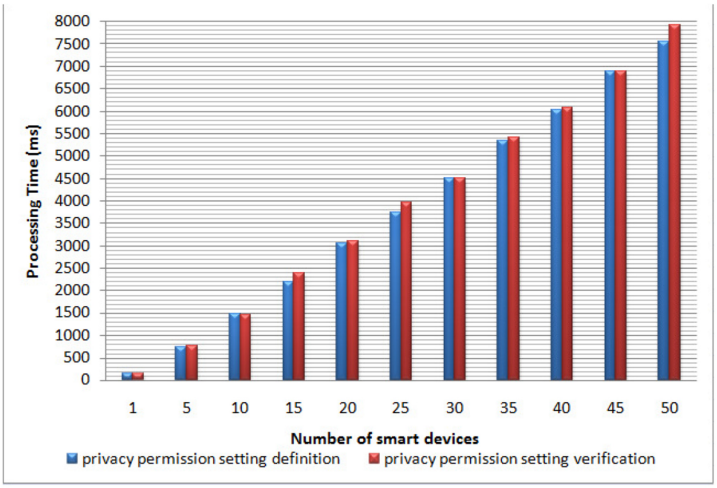


Fig. 6. Average computational cost of privacy permission setting definition and verification for fifty smart devices.

Table 2. A Comparison between Proposed Architecture and Existing Models

	Centralized-based management [2]	Distributed-based management [12, 13]	Blockchain-based management [8, 16]	Proposed model
Smart contract	No	No	No	Three types of smart contracts
IoT data privacy	The IoT data are stored in the centralized cloud server	The IoT data are stored in the storage server	The IoT data are stored in the blockchain database	The IoT data are stored in the storage devices that increases the privacy of the data owner
Permission Updating	Permissions are updated only if they are authorized by the centralized cloud server	Permissions need to be updated at each constrained IoT device	Permissions need to be updated at each place where ever they are used	Permissions are updated through the privacySettingUpdate function
Misbehavior Judging	No	No	No	Judging the misbehavior of the smart device and determines the corresponding penalty

data collected by the smart devices in storage devices. Besides, permissions are updated through the privacySettingUpdate function defined in the BehaviorControl smart contract.

Table 2 also shows that the biggest difference in the proposed model is the use of a smart contract to judge the misbehavior of the IoT devices and determine the corresponding penalty, which is not used in the existing models. Moreover, the behavior monitoring ensures that the data owner's privacy preferences be enforced in an untrustworthy IoT network.

It is therefore worth noting that the misbehavior-judging method was used before by Zhang et al. [22]. Thus, to evaluate our proposal efficiency, we compare it with the access control system proposed in [22], which is chosen because it is one of the latest approaches offering privacy-preserving access guarantees to the data owner; besides, it is the closest to our proposal. In Reference [22], each couple (subject, object) shares an AccessControlMethod smart contract. The authors defined sending access requests by one subject to the same object too frequently in a short period of time as misbehavior. Thus, three parameters are defined to characterize this misbehavior,

namely *minInterval*, which is the minimum time interval between two successive requests, *NoFR*, which is the number of frequent requests, and *threshold*, which is the maximum frequent request number in a minimum time interval. In case of any misbehavior detection, the subject is blocked for the duration of time, called *penalty*, which is computed by the Judge smart contract.

In our case, each object that can be the gateway owned by the data owner defined a BehaviorControl smart contract to manage several subjects (i.e., smart devices). Moreover, we defined three misbehavior types, such as (i) sending requests to invoke unauthorized action on one target (e.g., device output), (ii) sending requests during the penalty duration time, and (iii) sending multiple requests in a short period of time. Another difference between the proposed system and [22] is that our BehaviorControl smart contract maintained a history of the previous queries of each target thus, it can detect the misbehavior of receiving multiple requests from multiple subjects to the same target in a short period of time. In this case, the target can be momentarily blocked to be protected from this attack. In case of any misbehavior detection, the subject (or the target) is blocked for the computed duration of time, called *penalty*, using the following function:

$$penalty = length/frequencyThreshold \tag{1}$$

where *length* is the number of misbehavior that the subject had exhibited and *frequencyThreshold* is the maximum allowed request number in a short time period.

Unlike Zhang et al. [22], who addressed the access control issue, we focused on the IoT device management by offering new misbehavior types.

8 CONCLUSION

In recent years, several researchers have agreed that the combination of blockchain and IoT generates a peer-to-peer system, in which peers interact in an untrusthless and auditable manner. However, a few proposed solutions have dealt with the advantage of this technology to preserve the individuals' privacy by controlling their own smart devices. For this reason, we have proposed a privacy-preserving IoT device management framework based on the blockchain technology. In fact, the smart devices are controlled by several smart contracts that validate connection rights according to the privacy permission settings predefined by the data owners and the stored record array of detected misbehavior of each smart device. Moreover, we carried out several experiments to demonstrate the efficiency of the proposed solution. Then, both computation time cost and scalability overhead are analyzed. Finally, we compared our proposal with the existing approaches.

Moreover, it is worth noting that the blockchain use leads to a storage overhead cost. In future work, we plan to store only the newer blocks to overcome this issue. Indeed, the miners do not require storing all the blockchain for the long term. Thus, they can only save the hash of the previous blocks and not the entire blocks to keep the blockchain immutable.

REFERENCES

[1] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, et al. 2018. Hyperledger fabric: A distributed operating system for permissioned blockchains. In *Proceedings of the 13th EuroSys Conference*. ACM, 30.

[2] Ahmed Banafa. 2017. IoT and blockchain convergence: Benefits and challenges. *IEEE Internet of Things* (2017).

[3] Jorge Bernal Bernabe, Jose Luis Hernandez Ramos, and Antonio F. Skarmeta Gomez. 2016. TACIoT: Multidimensional trust-aware access control system for the Internet of Things. *Soft Comput.* 20, 5 (2016), 1763–1779.

[4] Vitalik Buterin et al. 2014. A next-generation smart contract and decentralized application platform. *White Paper* (2014).

[5] Jiachi Chen, Xin Xia, David Lo, John Grundy, and Xiaohu Yang. 2020. Maintaining smart contracts on Ethereum: Issues, techniques, and future challenges. *arXiv:2007.00286*. Retrieved from https://arxiv.org/abs/2007.00286.

[6] Cisco. 2016. Internet of Things At a Glance. Retrieved June 30, 2020 from https://www.cisco.com/c/en/us/products/collateral/se/internet-of-things/at-a-glance-c45-731471.pdf?dtid=osscdc000283.

[7] Roger Clarke. 2006. What's privacy. In *Proceedings of the Australian Law Reform Commission Workshop*, Vol. 28.
[8] Ali Dorri, Salil S. Kanhere, Raja Jurdak, and Praveen Gauravaram. 2017. Blockchain for IoT security and privacy: The case study of a smart home. In *Proceedings of the 2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops'17)*. IEEE, 618–623.
[9] Ganache. 2016. Ganache: Personal blockchain for Ethereum development. Retrieved June 30, 2020 from https://www.trufflesuite.com/ganache.
[10] Seonghyeon Gong, Erzhena Tcydenova, Jeonghoon Jo, Younghun Lee, and Jong Hyuk Park. 2019. Blockchain-based secure device management framework for an internet of things network in a smart city. *Sustainability* 11, 14 (2019), 3889.
[11] José L. Hernández-Ramos, Antonio J. Jara, Leandro Marín, and Antonio F. Skarmeta Gómez. 2016. DCapBAC: Embedding authorization logic into smart things through ECC optimizations. *Int. J. Comput. Math.* 93, 2 (2016), 345–366.
[12] Jose L. Hernandez-Ramos, Marcin Piotr Pawlowski, Antonio J. Jara, Antonio F. Skarmeta, and Latif Ladid. 2015. Toward a lightweight authentication and authorization framework for smart objects. *IEEE J. Select. Areas Commun.* 33, 4 (2015), 690–702.
[13] Dina Hussein, Emmanuel Bertin, and Vincent Frey. 2017. A community-driven access control approach in distributed IoT environments. *IEEE Commun. Mag.* 55, 3 (2017), 146–153.
[14] Kristián Košťál, Pavol Helebrandt, Matej Belluš, Michal Ries, and Ivan Kotuliak. 2019. Management and monitoring of IoT devices using blockchain. *Sensors* 19, 4 (2019), 856.
[15] Antony Lewis. 2016. A Gentle Introduction to Smart Contracts. Retrieved July 7, 2020 from https://bitsonblocks.net/2016/02/01/gentle-introduction-smart-contracts/.
[16] Damiano Di Francesco Maesa, Paolo Mori, and Laura Ricci. 2017. Blockchain based access control. In *Proceedings of the IFIP International Conference on Distributed Applications and Interoperable Systems*. Springer, 206–220.
[17] Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system. Retrieved on December 23, 2020 from https://bitcoin.org/bitcoin.pdf.
[18] Nxt community. 2016. Nxt Whitepaper. Retrieved July 7, 2020 from https://nxtdocs.jelurida.com/Nxt_Whitepaper.
[19] Solidity. 2014. Solidity Language. Retrieved June 30, 2020 from https://solidity.readthedocs.io/en/v0.7.1/introduction-to-smart-contracts.html.
[20] Truffle. 2016. Truffle: Ethereum Development Framework. Retrieved June 30, 2020 from https://github.com/trufflesuite/truffle.
[21] Web3. 2017. web3.js—Ethereum JavaScript API. Retrieved June 30, 2020 from https://github.com/ethereum/web3.js/.
[22] Yuanyu Zhang, Shoji Kasahara, Yulong Shen, Xiaohong Jiang, and Jianxiong Wan. 2018. Smart contract-based access control for the internet of things. *IEEE IoT J.* 6, 2 (2018), 1594–1605.

Received June 2020; revised September 2020; accepted November 2020