The Journal of Supercomputing
https://doi.org/10.1007/s11227-020-03605-3

Blockchain-based federation of wireless sensor nodes

F. J. Haro-Olmo[1] · J. A. Alvarez-Bermejo[1] · A. J. Varela-Vaca[2] · J. A. López-Ramos[1]

Accepted: 27 December 2020

Abstract
Wireless sensor networks (WSNs), as an integral part of most Internet of Things (IoT) devices, are currently proliferating providing a new paradigm of emerging technologies. It is estimated that the number of globally connected products will increase exponentially in the next decade. Therefore, it is not surprising finding applications in different areas such as smart homes, smart cities, industry, e-health, defense, security, vehicle networks, agriculture, and logistics among others. WSNs transmit the information gathered by the existing sensors in the IoT devices to header nodes acting as gateways to reach cloud computing nodes that will process this information. The processing of the data gathered through the sensors has made it possible to provide intelligence in strategical environments such as in the agrifood sector. Securing how this information is transmitted and assuring the integrity of the information is preserved. The use of blockchain technology has proven to be effective for securing the integrity of data transactions among entities. In this paper, we seize the advantage of this circumstance to design a robust mechanism based on smart contracts and blockchain technology that allow the reliable processing of data.

Keywords Secure communication protocol · IoT device · BaaS · Blockchain

✉ J. A. Alvarez-Bermejo
jaberme@ual.es

F. J. Haro-Olmo
fdo730@inlumine.ual.es

A. J. Varela-Vaca
ajvarela@us.es

J. A. López-Ramos
jlopez@ual.es

[1] Universidad de Almería, Almería, Spain

[2] Universidad de Sevilla, Sevilla, Spain

Published online: 12 January 2021

1 Introduction

Wireless sensor networks, as an integral part of most Internet of Things (IoT) devices, are currently proliferating providing a new paradigm of emerging technology, finding applications in different areas. Since the IoT concept appeared in 1999 [1], the number of globally connected products has exceeded 20 billion and it is exponentially increasing [2]. The data collected through the existing sensors in the IoT devices have made it possible to provide intelligence in many different ways. In general, the data are processed and transformed into information. For instance, in private environments as an intelligent home and in corporations where these devices can provide considerable help in the development of the production processes [3]. This reality leads us to focus on the information security requirements [4] that must be preserved, thus, the confidentiality, integrity, authentication, and availability, so that such communication takes place securely. Maintaining security on an IoT system is therefore crucial and of special relevance [5, 6]. Most of the risks assumed to come from the very definition of IoT [7], as an interrelated system of devices with the ability to transfer data over networks without human interaction. According to the Zscaler company report [8] IoT communications over the Zscaler cloud has increased, detecting some key points for security: (1) unauthorized IoT devices are increasing, most IoT communications are not secure and the only 17% of them use secure channels (SSL); (2) there is an exponential increase of IoT malware; and (3) new exploits are emerging to attack this type of devices. Thereby, it is necessary to find a way to globally protect the system, both the sensor network and components that provide services as a whole.

In this scenario, it is of special relevance to the advantages that blockchain technology brings in terms of security and privacy on the information [9]. The use of cryptography and consensus mechanisms [10, 11] to acknowledge (trust) each transaction reinforces the idea of security and integrity. The evolution experienced by the blockchain technology [12] provides the concept of smart contracts. Smart contracts are piece of code that will automatically execute when the pre-established conditions are met [13, 14]. Thus, smart contracts provide a new functionality to the transactions controlled by the blockchain. Once the information is received, it must be submitted to screening processes to determine its validity or rule out that no node of the sensor network is feeding the system with invalid either malicious data. In this paper, we propose a blockchain-based federated network model of wireless sensor nodes that ensures, on the one hand, secure communication within the sensor network by means of elliptic curve cryptography (ECC) [15, 16] and key exchange, on the other hand, security and integrity in the permanently stored data. Therefore, we ensure in our model the three fundamental aspects of information security: confidentiality, integrity, and availability features provided by the blockchain technology [10].

The rest of the paper is structured as follows: Sect. 2 is devoted to the related work regarding blockchain, smart contracts and its application as a mean to protect the information that IoT devices produce. In Sect. 3, the architecture is

proposed. Section 4 exposes the method to secure group communications of a set of IoT sensor nodes. In Sect. 5, it is shown how data can be screened by means of smart contracts before being stored in the blockchain. Section 6 shows the performance tests and results, and Sect. 7 conclusions are drawn.

2 Related works

Private blockchains require all members to be authenticated before being part of the group. This raises two main concerns: (1) how members (or nodes) can be securely added and/or removed by re-configuring the group; and, (2) how to double-check data, collected by sensors, before sending the transaction to the blockchain. In what refers to the integration of WSNs and blockchains, [17] exposes open challenges. The scenario is complex as the sensor nodes are restricted in terms of computational resources and much of this computing power is devoted to collecting and sending data through a not fully secure wireless link. Our proposal is based on a private blockchain design in which to apply cryptographic key exchange protocols to secure the communications between the sensor nodes and their gateway node as well as smart contracts to avoid data manipulation, damage or falsification of data as a way to prevent potential threats, attacks as well as misuse [20–22]. Cryptography is widely used to provide stronger security to the blockchain as in [14, 18, 19], elliptic curve cryptography is also used in Elliptic Curve Digital Signature Algorithm (ECDSA) to provide the addresses to be included in the blockchain.

Our proposed framework does not involve miners, just like in [23], with the associated savings in computing capacity. Regarding the power consumption, in [13] authors explore how the IoT and its integration with blockchain technology can be optimized. Using smart contracts is not new, in [24] authors propose a reliable blockchain-based framework which smart contracts can step in to provide solutions to data storage and access. It establishes a role-based access control (RBAC), and for the storage of the data, it mentions the use of Inter-Planetary File System (IPFS). IPFS is also used in [25] to the purpose of increasing security in the data storage when being distributed. In [26], a viable solution when the data is large. This is so-called an off-chain solution, where only part of the information is on the blockchain, especially everything related to integrity and verification. This was an option we explored and discarded to keep the blockchain storage resources to the minimum. Most IoT devices cannot afford to run a blockchain node nor an IPFS. Therefore, two types of nodes are defined in our network: data hub node (gateway for us) and IoT device node. While the IoT device nodes are limited to capturing data and sending it to the data hub node which in turn has greater computing capacity and is responsible for analyzing the received data and including it on the blockchain. The application of the blockchain to the IoT, and WSNs, has brought interesting benefits, also in the industrial environment as in [27], it has empowered the IoT devices providing them with greater operating autonomy within a decentralized network, where each device assumes its functions. Our proposal can be understood as a hybrid integration where part of the interaction and data exchanged between IoT devices, belonging to the same group, is added to the blockchain. And where the cryptographic protocol is

executed off-chain to ensure that all members of the group are legitimate. A slightly similar architecture is proposed in [28] creating an infrastructure of IoT devices integrated with the blockchain through IoT servers dedicated for the strongest computing tasks, so IoT devices will only have to handle very specialized tasks.

3 Proposed architecture

This section exposes the architectural proposal, see Fig. 1. It is based on an Ethereum [29] network configured in Amazon Web Services Elastic Cloud 2 (AWS EC2) [30]. AWS EC2 allows developers to provision on-demand compute resources, configure them to their needs, and scale resources up or down depending on application requirements. AWS EC2 has proven to be the most suitable option when compared with OpenStack [31] or HPC private supercomputing centers [32]. The setup was configured to be used with two, four, and eight nodes (virtual CPUs) running Ethereum Virtual Machines (EVM) [33], a bastion node, the balancers, and two Go-Ethereum (Geth) [34] nodes to monitor the architecture. This infrastructure serves as a Blockchain as a Service (BaaS) [35] to receive the information sent by the IoT devices that are collected as shown in Fig. 1. The IoT devices are organized in groups composed of sensor devices and a data hub node or bridge. This configuration is defined as a device farm.

Device farms are created by associating a set of sensor nodes by running the secure group communication protocol, see Sect. 4. Each node inside a device farm shares a cryptographic key used to sign messages sent to the bridge node, see Fig. 1. Each device farm is controlled, therefore, by a bridge node (i.e., light node) that serves as a point of communication between the IoT devices and the blockchain network. Each header node or bridge node that acts as the communication proxy between a certain group of IoT nodes or device farm and the blockchain has the necessary configuration to connect the bastion host (also known as Jump Host) in the

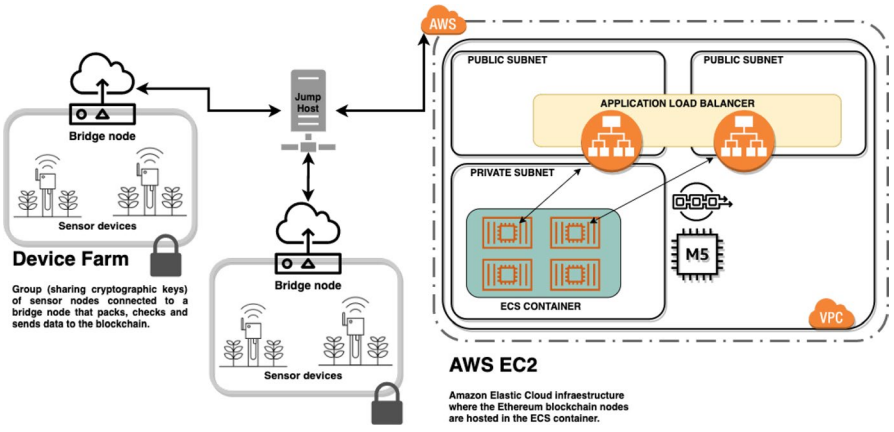

Fig. 1 Proposed AWS EC2 architecture

private Ethereum network through a SOCKS proxy [36], accounts connectivity is implemented through MetaMask and Web3. It allows to access the Ethereum blockchain from a web connection.

Bridge nodes have as their main functionality, in each IoT device farm, to act as a mean of connection with the Ethereum nodes in AWS. Bridge nodes are the ones who initiate the transactions to store, in the blockchain, the information collected by all the sensors or the actions required from the actuators installed in the IoT devices at every device farm. According to the protocol developed in Sect. 4, the bridge node must prove the validity of the messages received from each of the sensor nodes. This is done by checking the signatures and the identity of the sender. Attempts to inject messages that were produced by non-authorized IoT sensing nodes are easily rejected by the bridge node. Transactions created by bridge nodes are therefore containing data from verified nodes that, also, belongs to the group or device farm. Device farms are also a way of organizing nodes according to locality criteria that later, see Sect. 5, will help in deciding about the coherence of the data sent to the blockchain. Each bridge node is, therefore, in charge of keeping all the IoT devices within a certain device farm inside a secure group of authorized nodes so every node that belongs to the device farm is correctly authenticated and under complete control of its bridge node. Every communication among the IoT nodes and the Ethereum network is controlled by the bridge at the device farm. Thus, an additional security layer is included in each device in the device farm, preventing the information reaching the bridge from being malicious or manipulated by third parties. The secure layer is implemented using elliptic curves cryptography (ECC) [15, 16]. The curve and all the public information related to it is included in the first block of the blockchain; hence, all nodes have access to the curve details.

4 Secure group communication

Regarding the group communications, the preferred method is the multicast communication for a set of participating nodes. Therefore procedures for securely managing keys are needed due to communications characteristics, mainly in IoT environments where third parties can access easily to sensor nodes and try to intercept and reproduce conversations. Each node uses keys to encrypt and sign the data, this way the bridge node can certify the originality of the received information and can provide data integrity warranty when packing all the data to send it to the blockchain, as shown in Fig. 2. As the data in the blockchain are used to support application layers in business applications, the data integrity and validity are of interest. In [37], the authors developed methods to securely share keys in a group of communicating nodes.

In this scenario, we have proposed disjoint sets of IoT devices named IoT device farms. This is useful to parallelize smart contracts as we will prove later, see Sect. 5. Also it provides an interesting method to secure IoT devices with sensors and actuators. Therefore segregating the devices and distributing them in device farms are useful from a security point of view since we can create groups of IoT nodes that communicate their data to bridge nodes, but that is also subject

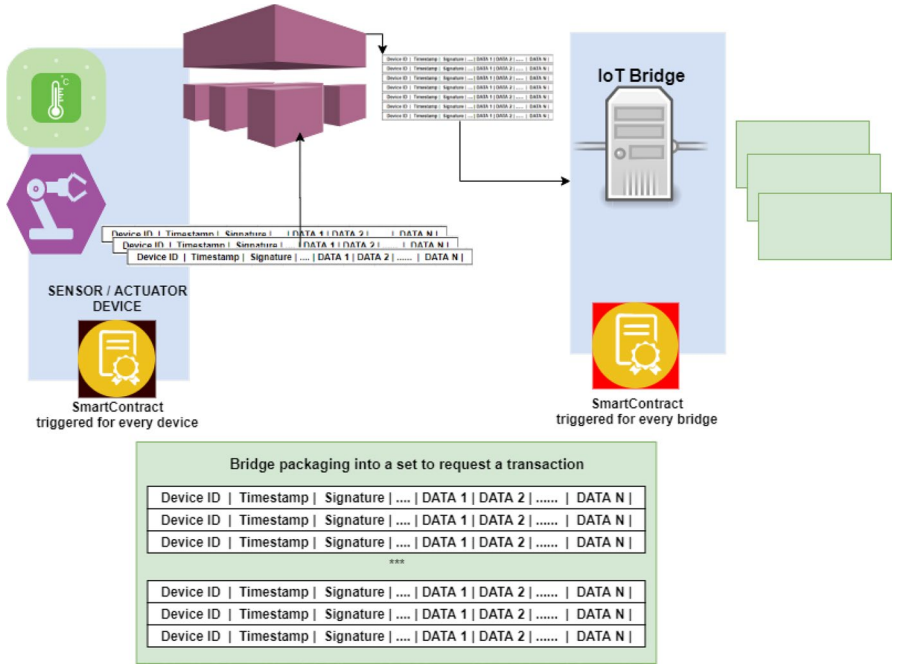

Fig. 2 Device and bridge operation and smart contracts

to continuous key refreshing. Since the bridge node executes the key exchange protocol either to reassure the environment, to add a new node or to remove a node that has stopped providing reliable information. Nodes providing unreliable information need to be removed from the device farm or ignored. To be able to know if the information provided by a sensor node is reliable or not we have to ensure if the node is legal or not (i.e., it is communicating with the bridge using correct keys) or if on the contrary, its sensors are misrepresenting the information (because they are out of calibration or because incorrect data are being injected on purpose). To identify illegal communications, the protocol in [37] has proven to be robust, valid, and effective. On the contrary, if the sensor node has experienced a miscalibration or the data being sent is corrupt on purpose, we will be able to identify the attack because when the information from the corrupt device reaches the bridge. It will be packaged and sent to the private network that hosts our Ethereum blockchain. At the reception, a smart contract will be activated, a *bridge smart contract* as it is defined in Sect. 5. This smart contract selects virtual CPUs on the Ethereum infrastructure (see Fig. 3) and invokes a *device smart contract*. This smart contract has the responsibility to query the measurement of a device and its neighbors to check if the values are consistent. If the *device smart contract* is not able to determine the existence of data coherence then the node will be marked as a suspicious node. The *bridge smart contract* will request its removal and a key refresh stage, according to the protocol, if the trend is confirmed. The data of the deleted node will not be stored in the processed block.

A device farm can be first created by setting all the nodes in the key exchange stage where a common and secret key is shared to cipher and sign the data that will be sent to the bridge node (see Table 1).

$\text{Device}_1, \ldots, \text{Device}_n$ retrieve information about the curve that is defined in the blockchain, E then selects P as a point in curve E. When a key exchange occurs, each device, Device_i, secretly and randomly selects a positive integer R_i. The key session is therefore the point defined as $K = (R_1 \ldots R_n)P$ related to the curve E.

To protect the IoT nodes operating under the control of a bridge node, all the nodes (bridge and devices) have already agreed on a certain E of prime order q that is stored in the blockchain, a pair of private integers need to be hold by each Device_i, let them be R_i and $S_i = R_i^{-1} \bmod q$.

In details, the protocol to share the secret key aforementioned follows the next steps:

1. Device_1 sends to Device_2 $R_1 P$.
2. For $i = 2, \ldots, n-2$, Device_i , receives from Device_{i-1}: $\prod_{r=1}^{i-1} R_r P$ and then calculates $R_i \prod_{r=1}^{i-1} R_r P$ and sends it to Device_{i+1}.
3. Device_{n-1} calculates $R_{n-1}(R_{n-2} \ldots R_1)P$ once this is done, this message is sent $\left(\prod_{j=1}^{n-1} R_j\right)P$ to Device_*.
4. Device_n can access to the shared key by computing $R_n\left(\prod_{j=1}^{n-1} R_j\right)P$.
5. Every device from Device_1 to Device_{n-1} computes $S_i\left(\prod_{j=1}^{n-1} R_j\right)P$ and sends it to Device_n.
6. Device_n computes $\{f_j\}_{j=1}^{n} = \left\{R_n\left[S_j\left(\prod_{i=1}^{n-1} R_j\right)P\right]\right\}_{j=1}^{n-1}$ and sends it to Device_i, $i = 1, \ldots, n-1$,
7. For $i = 1, \ldots, n-1$, Device_i gets the shared key with $S_i f_i$.

After running the protocol, all the devices $\text{Device}_1, \ldots, \text{Device}_n$ agree on the common key $\left(\prod_{r=1}^{n} R_r\right)P$. As $f_i = \left(\prod_{j=1;j\neq i}^{n-1} R_j\right)P$ since $S_i(R_i)P = P$.After the protocol, all the nodes are prepared to share the common key to send data to the bridge node.

Table 1 Protocol notation

Notation	Used to denote ...
K	Shared key by all the devices within the farm
n	IoT devices running the protocol (same farm)
i, j, k	Device index
E	Elliptic curve
P	Point in the curve E where q is its order
R_i	Secretly generated random number $\in \mathbb{Z}$ by Device_i;
Device_i	ith device in the farm
Device_*	All the devices in the group

5 Efficient smart contracts

A smart contract is triggered by a predefined event. Usually, they are activated to process transactions requested on the blockchain. Smart contracts can invoke, also, other smart contracts. To better understand how to design a smart contract model, it is necessary to know the parts that make it up. The smart contract is supported by a storage system called state database, in it is stored what is necessary for the execution of the contract, from variables to the same code that the EVM must execute to enforce the contract. The result of the execution of a contract will affect the content of the database and will also be stored in a block of the blockchain. The contracts are usually executed in sequence in the EC2-EVM nodes of our configuration arranged in the AWS. Until recently, it has been impossible to have more than one active thread successfully. But it is necessary to know that strategies can be adopted in which the same contract is applied to different data of the same transaction.

The dependencies generated between the smart contract code and its execution mean that the data sets provided in each transaction must be chosen very carefully so that there are no dependencies. The model can be approached to the unwinding of loops. If between iterations there are no dependencies, then each iteration can be executed in a different node and make the loop—whose nature is sequential—to be executed in parallel. Figure 3 shows how the nodes can be used to run contracts on different sets of data inside the same transaction. Two different types of smart contract were created, one to receive the transaction of each bridge node that collects all the data from the sensor nodes and creates data packets in which it identifies the sensor node, the data, the signature, timestamp, etc. The bridge node checks that it is correctly signed, decrypts it, packs it, and sends it to the jump host node. This task is performed by each bridge node for each set of IoT devices under its responsibility. To this node, it is associated the smart contract *bridge smart contract* that has the

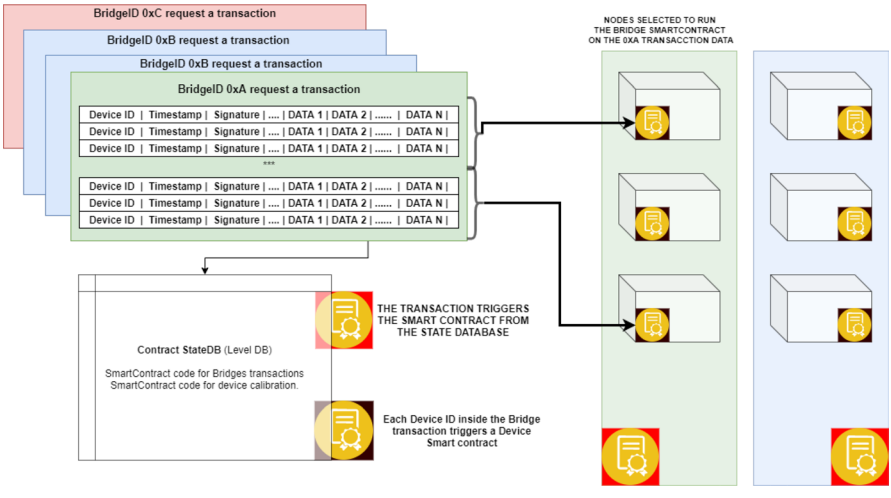

Fig. 3 Parallel smart contracts

Table 2 Time to insert and check data from a device (in milliseconds)

Number of operations	1000	2000	3000	4000	5000
Insertions	310	714	1191	1502	1925
Query operations	175	831	1636	7147	–

purpose of storing in the blockchain the data sent by each sensor (so that business logic applications, climate prediction, etc.) can be fed from the blockchain data with the assurance that they were validated and reliable.

The data package, sent by the bridge nodes, that will be processed by the *bridge smart contract* identifies each device along with its set of data collected. This is where the smart contract *device smart contract* is triggered. This smart contract is invoked from the *bridge smart contract*. When the *bridge smart contract*, which is executed over all the nodes, is started, it identifies how many different bridge transactions can be processed at the same time, hence if there are two bridges with available transactions, the EC2 nodes (which all execute the same contract) will read the bridge identifier. Therefore, half of the available nodes execute data sets coming from (see Fig. 3) 0xA and the other half of nodes coming from 0xB. Thus, half of all available EC2 nodes in the AWS will invoke *device smart contracts* from devices belonging to the 0xA bridge and the other half from 0xB. In this way, it is possible to perform a pseudo execution of smart contract device and bridge in parallel. Each *device smart contract* has the responsibility to retrieve the last readings from the device and check the consistency of the data sent. This is done by, see Fig. 4, (1) checking the data from the device by querying its history in the database associated to the smart contract, and (2) comparing the device data with the data just received from other devices of the same bridge. The *device smart contract* was designed using an index-based, multi-mapping approach. The query to check the validity, or coherence, of the data sent by each device is always under 2 s for under

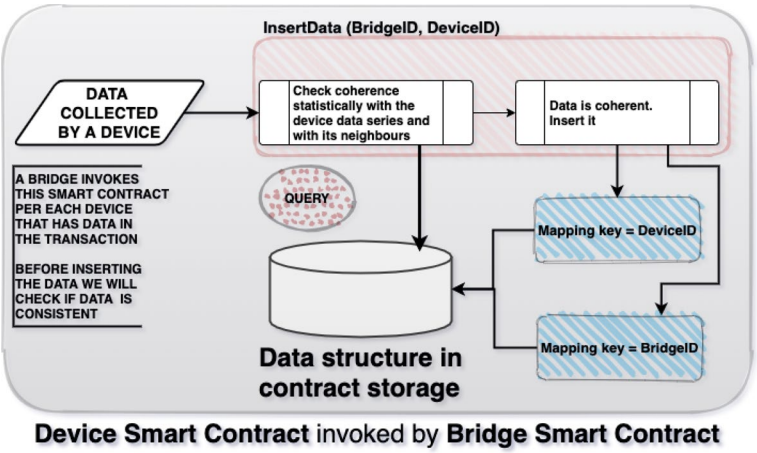

Fig. 4 Operations in a device smart contract

3000 queries. When this limit is overcome the performance is prohibitive as shown in Table 2.

Up to 5000 insertions were tested. Each block of data from each device is stored in a mapping searchable by the device internal ID (which is handled by the bridge contract). This ID is unique. This data are also stored by using an additional mapping which is the BridgeID (identifies the device farm), Table 2 shows the performance, in milliseconds, of the *device smart contract* when processing up to 5000 data transactions from a single device.

When the control is returned to the *bridge smart contract*, it will perform the check of each device. As shown in Fig. 3, each blockchain node is dedicated to performing parallel device checks.

6 Results

To implement the IoT devices rekeying protocol to secure group communications, Raspberry Pi 3 Model B with 12GB of memory and Ubuntu 18.04 LTS were used. Table 3 shows how fast the IoT group can be secured with new keys shared from the bridge node. Columns marked with asterisks mean that there were no such number of physical devices but were emulated using threads on a physical device (e.g., 16* means two threads in each one of the eight devices).

To create the blockchain environment, AWS EC2 (elastic computing cloud) services have been used with an M5 instance with Intel Xeon© Platinum 8175M processor technology 3.1 GHz, Intel (AVX-512). Storage of the instance offered via EBS or NVMe SSD physically connected to the hosting server to perform blockchain-related computing tasks. Figure 5 shows how the data were processed by varying the number of transactions per block and the number of available nodes (virtual CPUs) in the blockchain.

During the tests, 2, 4, and 8 CPUs have been used to distribute the execution of the smart contracts related to the devices. Smart contracts were executed sequentially and then separated by BridgeID of each Bridge node.

The sequential version implies that all the CPUs are running the smart contract on the same data. The optimized smart contract runs the smart contract on all nodes, but it creates subsets of nodes, each subset is devoted to a specific transaction data. As it can be seen in Fig. 5, there is a meaningful performance gain when adopting the strategy of segregating nodes to attend different sets of data. Adding more virtual CPUs to the computation was proven to be inefficient over $n = 4$. The configuration used was conditioned by using SSD storage.

Table 3 Time in milliseconds to refresh the keys for the group

8	16*	32*	64*	128*	256*	512*	1024*
0.000548	0.0011	0.001478	0.004942	0.013898	0.19402	0.05176	0.14292

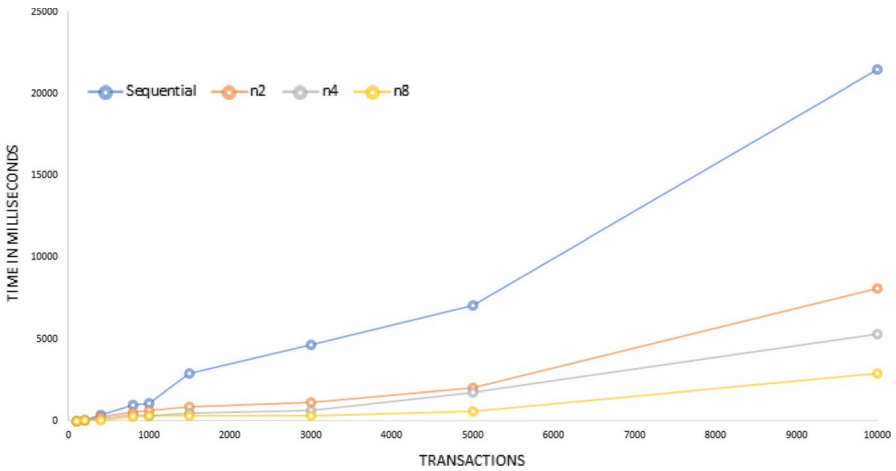

Fig. 5 Results of the performance tests

7 Conclusions

The use of blockchain proves to be effective for transactions between two entities. Taking advantage of this circumstance to try to have a robust mechanism that allows, as in our case, the reliable collection of validated data is more than justified. Once the transaction is incorporated into a block of the blockchain, the data will remain immutable and available to larger applications (or application layers) that may need to use this data for reasons related to their business logic.

The system of membership of a node to the network (private, as is the case here) also ensures that the nodes involved in the collection of information (which will then be incorporated into the blockchain) are nodes without the intention of corrupting the data voluntarily nor subtracting them.

However, the nature of the blockchain does not prevent that, even if the person who sends the data is a valid computer element, the data is incorrect due to, for example, sensor miscalibration, plate breakage due to exposure to high temperatures, etc. Fortunately, there is the ability to use smart contracts, which are nothing more than a mechanism that is activated every time there is an event between two parts that are related through the blockchain. These smart contracts allow certain controls or activities to be performed before the transaction is closed. During this work, we wanted to exploit this capacity so that before incorporating sensor data to a blockchain we can be able (taking advantage of the execution of the smart contract) to validate the consistency of the data received by contrasting it with the history (stored in the blockchain) of each sensor. And once the coherence is checked within a threshold of previous readings, enter a second phase where it would be compared with the coherence of the readings of the nodes that collect data in the same "farm" (context).

However, using the smart contracts for this work has the disadvantage of performance. Therefore, certain means have been provided for smart contracts to operate

on subsets of data incorporated in each transaction. The results show gains when applied. In addition, smart contracts have the ability to recommend the removal or addition of an IoT device in a given workgroup (or farm). Furthermore, the ability to strengthen security on each farm through the application of a secure key exchange protocol, supported by an elliptical curve.

It has been shown, therefore, how to protect group communications between sensor nodes, actuators, and the bridge that communicates them with the blockchain, as well as the optimization of the incorporation of these data to the blockchain.

Acknowledgements Ministry of Science and Technology of Spain, ECLIPSE (RTI2018-094283-B-C33); Junta de Andalucía, via COPERANICA and METAMORFOSIS project; European Regional Development Fund (FEDER).

Authors contributions All the authors are responsible for the concept of the paper, the results presented, and the writing. All the authors have approved the final content of the manuscript.

Compliance with ethical standards

Conflict of interest No potential conflict of interest was reported by the authors.

References

1. Li S, Xu LD, Zhao S (2015) The internet of things: a survey. Inf Syst Front 17:243–259. https://doi.org/10.1007/s10796-014-9492-7
2. Mähler V, Westergren UH (2019). Working with IoT—a case study detailing workplace digitalization through IoT system adoption. In: Strous L, Cerf V (eds) Internet of things. Information processing in an increasingly connected world. IFIPIoT 2018. IFIP advances in information and communication technology, p 548. https://doi.org/10.1007/978-3-030-15651-0_15
3. Deuker R, Meinhardt S (2019) IoT best practices. HMD 56:1091–1093. https://doi.org/10.1365/s40702-019-00574-x
4. Panagiotis I, Grammatikis R, Sarigiannidis PG, Moscholios ID (2019) Securing the Internet of Things: challenges, threats and solutions. Internet of Things 5:41–70. https://doi.org/10.1016/j.iot.2018.11.003
5. Kandasamy K, Srinivas S, Achuthan K (2020) IoT cyber risk: a holistic analysis of cyber risk assessment frameworks, risk vectors, and risk ranking process. EURASIP J Inf Secur 2020:8. https://doi.org/10.1186/s13635-020-00111-0
6. Boeckl K, Fagan M, Fisher W, Lefkovitz N, Megas K, Nadeau E, O'Rourke DG, Piccarreta B, Scarfone K (2019) Consideration for managing internet of things (IoT) cybersecurity and privacy risk. National Institute of Standards and Technology, Gaithersburg. https://doi.org/10.6028/NIST.IR.8228
7. Elijah O, Rahman TA, Orikumhi I, Leow CY, Hindia MN (2018) An overview of internet of things (IoT) and data analytics in agriculture: benefits and challenges. IEEE Internet of Things J 5(5):3758–3773. https://doi.org/10.1109/JIOT.2018.2844296
8. Zscaler cloug security (2020) IoT in the Enterprise 2020 report
9. Nakamoto S (2008) Bitcoin: a peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf
10. Gamage HTM, Weerasinghe HD, Dias NGJ (2020) A survey on blockchain technology concepts, applications, and issues. TSN Comput Sci 1:114. https://doi.org/10.1007/s42979-020-00123-0
11. Salimitari M, Chatterjee M, Yaser P, Fallah A (2020) A survey on consensus methods in blockchain for resource-constrained IoT networks. Internet of Things. https://doi.org/10.1016/j.iot.2020.100212
12. Szabo N (2018) Smart contracts: building blocks for digital markets
13. Reyna A, Martín C, Chen J, Soler E, Díaz M (2018) On blockchain and its integration with IoT. Challenges and opportunities. Future Gener Comput Sys 88:173–190. https://doi.org/10.1016/j.future.2018.05.046
14. Davila C, Tarnow J (2019) The blockchain in IoT. In: Rayes A, Salam S (eds) Internet of things from hype to reality. Springer, Cham, pp 269–296. https://doi.org/10.1007/978-3-319-99516-8_10
15. Caelli WJ, Dawson EP, Rea SA (1999) PKI, elliptic curve cryptography, and digital signatures. Comput Secur 18(1):47–66. https://doi.org/10.1016/S0167-4048(99)80008-X

16. Marzouqi H, Al-Qutayri M, Salah K (2015) Review of elliptic curve cryptography processor designs. Microprocess Microsyst 39(2):97–112. https://doi.org/10.1016/j.micpro.2015.02.003
17. Marchang J, Ibbotson G, Wheway P (2019) Become will blockchain technology, a reality in sensor networks?. Wireless Days (WD), Manchester, pp 1–4. https://doi.org/10.1109/WD.2019.8734268
18. Dasgupta D, Shrein JM, Gupta KD (2019) A survey of blockchain from security perspective. J Bank Financ Technol 3:1–17. https://doi.org/10.1007/s42786-018-00002-6
19. Sultan A, Mushtaq MA, Abubakar M (2019). IOT security issues via blockchain: a review paper. In: Proceedings of the 2019 International Conference on Blockchain Technology—ICBCT 2019, pp 60–65. https://doi.org/10.1145/3320154.3320163
20. Tapas N, Longo F, Merlino G, Puliafito A (2020) Experimenting with smart contracts for access control and delegation in IoT. Future Gener Comput Syst 111:324–338. https://doi.org/10.1016/j.future.2020.04.020
21. Ante L (2020) Smart contracts on the blockchain—a bibliometric analysis and review. Telemat Inf. https://doi.org/10.1016/j.tele.2020.101519
22. Zhang E, Li M, Yiu S-M, Du J, Zhu J-Z, Jin G-G (2021) Fair hierarchical secret sharing scheme based on smart contract. Inf Sci 546:166–176. https://doi.org/10.1016/j.ins.2020.07.032
23. Dorri A, Kanhere SS, Jurdak R (2017) Towards an optimized blockchain for IoT. In: Proceedings of the Second International Conference on Internet-of-Things Design and Implementation (IoTDI '17). Association for Computing Machinery, New York, pp 173–178. https://doi.org/10.1145/3054977.3055003
24. International Conference on Blockchain and Trustworthy Systems, Zheng Z, Dai H-N, Tang M, Chen X (2020) Proceedings of the Blockchain and trustworthy systems: first International Conference, BlockSys 2019, Guangzhou, China, December 7–8, 2019. https://doi.org/10.1007/978-981-15-2777-7_27
25. Javed MU, Rehman M, Javaid N, Aldegheishem A, Alrajeh N, Tahir M (2020) Blockchain-based secure data storage for distributed vehicular networks. Appl Sci (Switz) 10(6):10. https://doi.org/10.3390/app10062011
26. Gupta R, Tanwar S, Kumar N, Tyagi S (2020) Blockchain-based security attack resilience schemes for autonomous vehicles in industry 4.0: a systematic review. Comput Electr Eng 86:10. https://doi.org/10.1016/j.compeleceng.2020.106717
27. Pu S (2020) Industrial applications of blockchain to IoT data. In: Yano M, Dai C, Masuda K, Kishimoto Y (eds) Blockchain and crypto currency. Economics, law, and institutions in Asia Pacific. Springer, Singapore. https://doi.org/10.1007/978-981-15-3376-1_3
28. Hang L, Kim D-H (2019) Design and implementation of an integrated IoT blockchain platform for sensing data integrity. Sensors 19(10):2228. https://doi.org/10.3390/s19102228
29. Buterin V (2014) A next-generation smart contract and decentralized application platform. White paper
30. Marozzo F (2019) Infrastructures for high-performance computing: cloud infrastructures. In: Encyclopedia of bioinformatics and computational biology. Academic Press, pp 240-246, ISBN 9780128114322. https://doi.org/10.1016/B978-0-12-809633-8.20374-9
31. https://www.openstack.org/assets/survey/April-2016-User-Survey-Report.pdf
32. A Marathe, Harris R, Lowenthal DK, de Supinski BR, Rountree B, Schulz M et al (2013) A comparative study of high performance computing on the cloud. In: Proceedings of the 22nd International Symposium on High-Performance Parallel and Distributed Computing
33. Wood G (2014) Ethereum: a secure decentralised generalised transaction ledger. http://yellowpaper.io/
34. https://github.com/ethereumigo-ethereumi
35. Singh J, Michels JD (2018) Blockchain as a service (BaaS): providers and trust. In: IEEE European symposium on security and privacy workshops, London, pp 67–74
36. Fung KP, Chang RKC (2000) A transport-level proxy for secure multimedia streams. IEEE Internet Comput 4(6):57–67. https://doi.org/10.1109/4236.895017
37. Álvarez-Bermejo JA, Lodroman A, López-Ramos JA (2016) Distributed key agreement for group communications based on elliptic curves. An application to sensor networks. Math Methods Appl Sci 39:4797–4809. https://doi.org/10.1002/mma.3802

Terms and Conditions

Springer Nature journal content, brought to you courtesy of Springer Nature Customer Service Center GmbH ("Springer Nature").

Springer Nature supports a reasonable amount of sharing of research papers by authors, subscribers and authorised users ("Users"), for small-scale personal, non-commercial use provided that all copyright, trade and service marks and other proprietary notices are maintained. By accessing, sharing, receiving or otherwise using the Springer Nature journal content you agree to these terms of use ("Terms"). For these purposes, Springer Nature considers academic use (by researchers and students) to be non-commercial.

These Terms are supplementary and will apply in addition to any applicable website terms and conditions, a relevant site licence or a personal subscription. These Terms will prevail over any conflict or ambiguity with regards to the relevant terms, a site licence or a personal subscription (to the extent of the conflict or ambiguity only). For Creative Commons-licensed articles, the terms of the Creative Commons license used will apply.

We collect and use personal data to provide access to the Springer Nature journal content. We may also use these personal data internally within ResearchGate and Springer Nature and as agreed share it, in an anonymised way, for purposes of tracking, analysis and reporting. We will not otherwise disclose your personal data outside the ResearchGate or the Springer Nature group of companies unless we have your permission as detailed in the Privacy Policy.

While Users may use the Springer Nature journal content for small scale, personal non-commercial use, it is important to note that Users may not:

1. use such content for the purpose of providing other users with access on a regular or large scale basis or as a means to circumvent access control;
2. use such content where to do so would be considered a criminal or statutory offence in any jurisdiction, or gives rise to civil liability, or is otherwise unlawful;
3. falsely or misleadingly imply or suggest endorsement, approval , sponsorship, or association unless explicitly agreed to by Springer Nature in writing;
4. use bots or other automated methods to access the content or redirect messages
5. override any security feature or exclusionary protocol; or
6. share the content in order to create substitute for Springer Nature products or services or a systematic database of Springer Nature journal content.

In line with the restriction against commercial use, Springer Nature does not permit the creation of a product or service that creates revenue, royalties, rent or income from our content or its inclusion as part of a paid for service or for other commercial gain. Springer Nature journal content cannot be used for inter-library loans and librarians may not upload Springer Nature journal content on a large scale into their, or any other, institutional repository.

These terms of use are reviewed regularly and may be amended at any time. Springer Nature is not obligated to publish any information or content on this website and may remove it or features or functionality at our sole discretion, at any time with or without notice. Springer Nature may revoke this licence to you at any time and remove access to any copies of the Springer Nature journal content which have been saved.

To the fullest extent permitted by law, Springer Nature makes no warranties, representations or guarantees to Users, either express or implied with respect to the Springer nature journal content and all parties disclaim and waive any implied warranties or warranties imposed by law, including merchantability or fitness for any particular purpose.

Please note that these rights do not automatically extend to content, data or other material published by Springer Nature that may be licensed from third parties.

If you would like to use or distribute our Springer Nature journal content to a wider audience or on a regular basis or in any other manner not expressly permitted by these Terms, please contact Springer Nature at

onlineservice@springernature.com