European Journal of Computer Science and Information Technology, 12 (2),75-92, 2024
Print ISSN: 2054-0957 (Print),
Online ISSN: 2054-0965 (Online)
Website: https://www.eajournals.org/
Publication of the European Centre for Research Training and Development -UK

A Smart Contract-based Blockchain Solution in IoT Networks

Ofuchi Ngozi Rich Olieh

University of Nigeria, Department of Computer Science

doi: https://doi.org/10.37745/ejcsit.2013/vol12n27592 Published April 27, 2024

Citation: Olieh O.N.R. (2024) A Smart Contract-based Blockchain Solution in IoT Networks, European Journal of Computer Science and Information Technology, 12 (2),75-92

ABSTRACT: *The emergence and growing use of advanced technologies has opened up new possibilities for addressing the security challenges of resource-constrained IoT net- works. As IoT devices exchange sensitive data, secure key management is essential for IoT network security, particularly during the key revocation phase. However, current IoT key management solutions require improvements due to the resource limitations of IoT devices. Despite these limitations, existing key revocation solutions still have several areas for improvement, including high communication overheads. Therefore, a decentralized and efficient solution is necessary to address these issues in IoT networks, with a focus on security. This paper proposes a new solution for key revocation based on Blockchain technology using smart contracts to minimize communication overhead and energy consumption in IoT networks. The paper presents a security and performance analysis to assess its correctness. The results indicate that our proposal outperforms other solutions by having a reduced communication overhead of 93.55%, 91.87%, and 99.75% compared to other solutions during the compromising, leaving, and draining cases, respectively. This demonstrates that our solution is efficient and suitable for IoT networks.*

KEYWORDS: internet of things (IoT), blockchain, security, wireless sensor networks (WSNs), key revocation

INTRODUCTION

The emergence of the Internet of Things (IoT) has brought about a significant techno- logical shift, enabling us to interact with smart homes, wearable devices, and industrial automation in ways we never thought possible. However, security and privacy concerns have become significant challenges as the number of connected devices increases. One of the most critical security issues in IoT networks is key management in devices with limited resources.

Robust key management solutions are very important, especially in dynamic and interconnected networks such as Internet of Things (IoT) networks. Robust key management serves as a crucial pillar for ensuring the integrity and confidentiality of data exchanged within IoT networks. In this context, effective key management involves a meticulous process of generating, distributing, managing, and revoking cryptographic keys. These keys play a pivotal role in encrypting and decrypting communication, ensuring that sensitive information remains secure from unauthorized access. Beyond its cryptographic function, key management also contributes to secure device authentication and facilitates the revocation and update of keys as needed.

Key revocation is an essential phase of key management schemes. However, traditional systems require a central authority to manage and distribute keys, creating single points of failure vulnerabilities and significant communication overhead, leading to potential security breaches and high energy consumption. To address these challenges, blockchain technology has emerged as a potential solution. Blockchain's properties, such as decentralization, immutability, and security, make it ideal for various applications, including key management and revocation in IoT networks. This paper proposes a new solution combining blockchain for key revocation using smart contracts in IoT networks. This solution uses blockchain to decentralize the key revocation process and enhance security in IoT networks. It demonstrates the feasibility and effectiveness of using blockchain-based smart contracts for key revocation in devices with limited resources.

Furthermore, the paper aims to analyze the potential benefits of blockchain-based key revocation in IoT networks, including improved security, reduced communication overhead, and energy consumption.

The proposed solution has the following features:

• Decentralization: The proposed solution is decentralized, meaning there is no central authority to manage the key revocation process. Instead, the blockchain network participants are responsible for the key revocation process.

• Transparency: The proposed solution is transparent, meaning that all the

participants can see each device's key revocation process and revocation status.

• Reduced communication overhead and energy consumption: The proposed solution reduces the communication overhead by using blockchain-based smart contracts to manage the key revocation process.

• Security: The proposed solution is secure because it uses blockchain technol-

ogy to manage the key revocation process, making it difficult for an attacker to compromise the system.

By incorporating blockchain into our key revocation solution, this proactive approach minimizes the window of opportunity for potential security breaches, ensur- ing enhanced security and system transparency. Furthermore, the proposed approach significantly reduces communication overhead and energy consumption, making it suitable for resource-constrained IoT devices.

The paper's organization is as follows: Section 2 provides a brief background on Blockchain, Section 3 comprehensively reviews related work in key management and revocation in IoT networks, Section 4 describes the proposed blockchain-based key revocation mechanism, including the smart contract design and implementation, Section 5 evaluates the performance and security of the proposed mechanism and compares it with existing approaches, and finally, Section 6 concludes and highlights future research directions in this field.

Preliminaries

Blockchain

Decentralized ledger technology, known as Blockchain, is a chain of blocks first intro- duced in 2008 by Nakamoto in a paper on decentralized peer-to-peer cash transactions [2]. The primary goal of Blockchain is to create a shared ledger distributed across all parties in the network, which contains all transactions made in the network and is synchronized using a consensus algorithm. Blockchain offers several benefits: distri- bution, security, traceability, and immutability [3, 4]. Different consensus algorithms have been proposed to ensure a common consensus among participants, such as Proof of Work [2], Proof of Stake [5], Delegated Proof of Stake [6], and Practical Byzan- tine Fault Tolerance [7]. Proof of Work is the first algorithm introduced in Blockchain as a consensus, which is used in Bitcoin [2]. It is a computationally intensive algo- rithm that requires much computing power to solve a cryptographic puzzle. Proof of Stake is another consensus algorithm that does not require much computing power. Instead, participants must stake a certain amount of virtual currency to participate in the consensus algorithm. Delegated Proof of Stake is a variation of Proof of Stake that requires participants to vote for a certain number of delegates to participate in the consensus algorithm. Practical Byzantine Fault Tolerance is a consensus algorithm that requires a certain number of participants to agree on a transaction before it is added to the Blockchain. Blockchain technology comes in various types, each designed to serve specific purposes and address different use cases.

Types of Blockchain technology

Blockchain can be classified into three types: public, private, and consortium. A public blockchain lets anyone join the network and participate in the consensus algorithm. In contrast, only a restricted number of participants from the same organization can participate in private Blockchain. In the Blockchain consortium, participants from various organizations can participate in the consensus algorithm.

In 1994, Nick Szabo first introduced Smart contracts when he proposed a digital contract [8]. They are self-executing programs on Blockchain, are utilized to auto- mate contract execution between parties, and reduce time and cost without needing a trusted third party. They are deployed on the Blockchain and are executed when a specific condition is met. They are immutable and cannot be modified once deployed on the Blockchain.. Blockchain has numerous applications in various fields, including key management and data integrity. Several studies have utilized Blockchain technology, such as those presented in [3, 9–13].

Smart contract

A smart contract is a self-executing contract with the terms of the agreement directly written into code. It is a piece of code that runs on a blockchain and is designed to automatically enforce, execute, or facilitate the negotiation of a contract when predefined conditions are met. Smart contracts operate on the principles of decen- tralization, transparency, and automation.

Smart contracts complete blockchain with several advantages:

• Consensus: Upon information transmission to the blockchain by a node, all nodes record the information, ensuring its effectiveness through consensus.

• Anti-tampering: With all nodes storing the uploaded information, tampering

becomes challenging, fostering the integrity of the data.

• Certainty: Once information is uploaded, it becomes irreversible. If an error occurs in the uploaded information, a new upload is required. Simultaneously, both sets of information remain queryable at any node within the blockchain.

RELATED WORK

Multiple research papers introduced different approaches and solutions to the key management problem with its different phases: key generation, distribution, pairwise key establishment, group-wise key establishment, new node addition, key refresh, and revocation. In this section, we discuss the most relevant works in key revocation.

According to [14], the authors proposed a matrix-based key management scheme, on which the key revocation is one of its phases. They distinguish between two scenar- ios: the first one is when a node gets compromised and detected by the gateway, and the second one is when a node needs to leave the network voluntarily. In the first sce- nario, upon detecting malicious activity from a node by a gateway, the gateway starts revocation by deleting the node’s keys from its memory, its secrets from its matrix, and its identifier from its neighboring vector. It then broadcasts a revocation message to all nodes in the network. Upon receiving the revocation message, each node, as the gateway already did, deletes the compromised node’s keys from its memory, its secrets from its matrix, and the malicious node identifier from the neighbors’ vector. It then replies to the gateway with an acknowledgment message confirming that it has suc- cessfully revoked the compromised node. The gateway then starts the rekeying process to establish new keys with the remaining nodes in the network.

The remaining nodes also start rekeying to establish new keys with their neighbors. In the second scenario, when a node needs to leave the network voluntarily, it sends a leave message to its neighbors. Upon receiving the leave message, each node deletes the leaving node’s keys from its memory, its secrets from its matrix, and the leaving node identifier from the neighbors’ vector. It then replies to the leaving node with an acknowledgment message confirming that it has successfully revoked the leaving node. The leaving node leaves the network after receiving the acknowledgment message from all its neighbors and erasing its memory. The remaining nodes start rekeying to establish new keys with their neighbors.

In the scheme, the authors introduced a node deletion phase that involves neighboring nodes of a compromised or leaving node deleting it from their neighboring tables and adjusting their polynomials by reducing them by one term. Additionally, they remove the shared pairwise keys from their memory. Nonetheless, in their approach, the authors did not provide details regarding detecting a revoked node and the communications necessary for handling such an event.

As stated by the authors in, neighboring nodes are responsible for the key revocation process of a node. No messages are sent to neighbor nodes, yet each node invalidates the key of any node that fails to send a message within a predetermined time interval. Additionally, nodes perform a key generation phase every Tr, and the server transmits a set of new one-one functions to all nodes every (2∗Tr). In the event of attack detection, the server sends a refresh message to all nodes via the gateway node.

This message includes the number of attacks to refresh the value of Tr. Receiving and responding to the server's refresh message ensures that the communication overhead of the solution is similar to that of. In their study, Mesmoudi et al. delineate two primary scenarios for key revoca- tion within their scheme as their network model includes two distinct types of nodes: the cluster head (CH) and the cluster member (CM). The first scenario occurs when the base station (BS) identifies a compromised CH. At this point, it revokes the com- promised CH's key and associated CMs, initiating a key refresh for the remaining nodes. The second scenario arises when a CH detects a compromised CM, prompting it to revoke the key of the affected CM and notify the other CMs within the cluster while initiating the key renewal process. Notably, the scheme does not account for key revocation when a node leaves the network or experiences battery depletion.

The authors of proposed a decentralized blockchain-based scheme that featured two layers - one for key management and the other for nodes' key management. To revoke the key of a leaving node, they sent a refresh message to the node set of the leaving node containing the identifier and a randomly generated Kr that was used to refresh the keys of the remaining nodes. They also broadcast a refresh message to other sets of nodes containing the identifier of the leaving node set and Kr. When the second layer of this scheme, composed of Blockchain participants (BPs), received information about the leaving node, it created a new transaction containing the identifier and the other information about the leaving node. Finally, all the BPs verify this transaction by checking the signature, hashes, and validity. This scheme decentralizes the key revocation process and resolves the single point of failure problem faced by centralized schemes. However, the constrained nodes' participation in the key revocation process produces a high communication overhead.

Recently in a blockchain-based key management solution is presented for dis- tributed IoT architectures. The network is modeled as three layers, i. e., an equipment layer, an intermediate layer, and a cloud layer. The authors use hash chains and a key distribution scheme based on network security coding to secure communications. The one-way hash chains offer the identity authentication process between the intermediate layer and the equipment layer, as well as the mutual identity authentication process during device communication. The latter employs secure network coding principles to fortify the security of transmitting sensitive information. Sensors and IoT devices are in the equipment layer and the blockchain structure is stored in the intermediate layer to avoid heavy computational to the resource-limited devices. The intermediate layer is connected to the cloud layer which manages each blockchain network. Adding an intermediate layer in the network architecture to reduce the computing time and overhead of IoT devices makes this solution salable. However, the IoT devices have to run heavy operations within an asymmetric cryptosystem. In addition, key revocation phase is not presented in the solution.

Proposed solution

Network model

Our network model comprises two fundamental components:

- IoT devices: These are the primary components of the network that collect and transmit data from the environment to the gateway nodes. Ranging from small sensor nodes to large industrial sensor-equipped machines and appliances, these devices are often constrained in terms of resources such as memory, processing, power, and communication.

- Gateway nodes: Acting as intermediaries between the IoT devices and the

remote server, these resource-rich devices collect data from the IoT devices and forward it to the remote server. Each gateway node is a participant in the blockchain network.

- Remote server: This component is responsible for receiving data from the gate-

way nodes. It processes the data further, such as storing it in a database, analyzing it, visualizing it, and making it available via an API.

On another side, the network model is structured into two layers. The first layer comprises the IoT devices, which are responsible for key management and data col- lection. The second layer consists of the gateway nodes, which integrate blockchain technology and manage data transmission from and to the remote server.

In this setup, each gateway is linked to the remote server and has one or more IoT devices connected to it. At the same time, each IoT device is linked to at least one gateway node, which is responsible for managing the device’s keys.

The network model is illustrated in Figure 1.

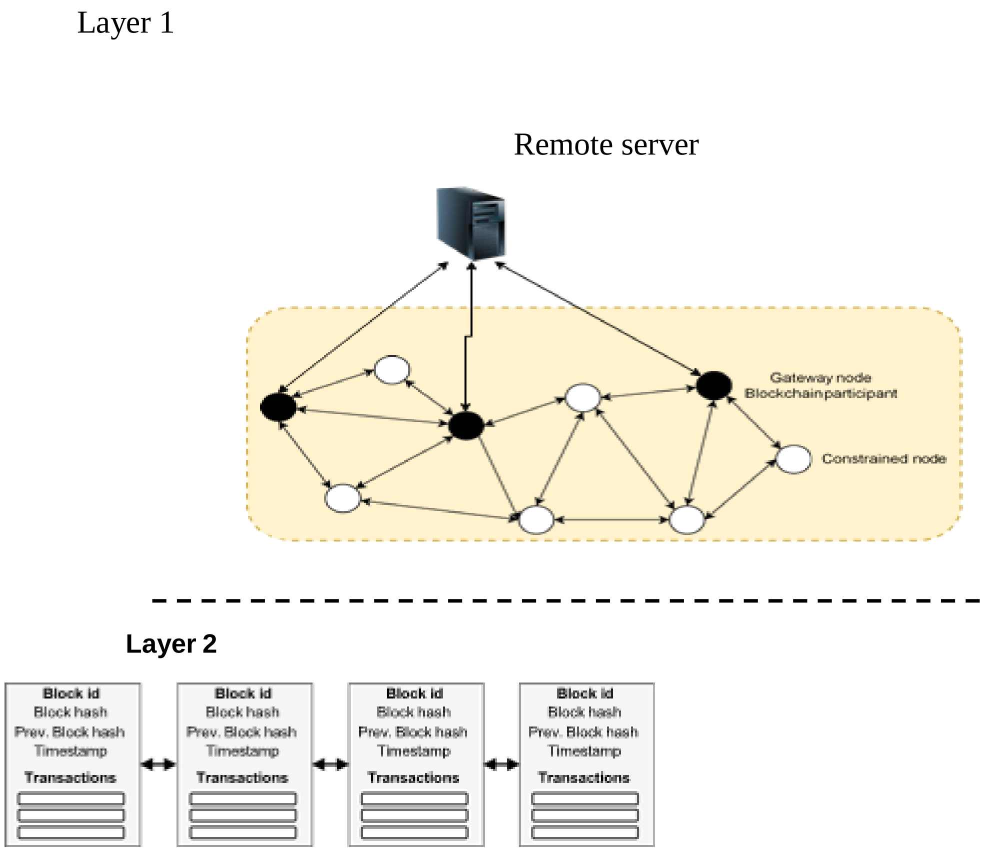

Fig. 1 Network model.

Notations

The used notations in this paper are shown in Table 1.

Notation	Meaning
lms	The sent message size in bytes
lmr	The received message size in bytes
d	The total number of a node's neighbors
n	The total number of nodes in the network
ns	The number of a set member nodes
ngm	The number of member nodes that are linked to the gateway *g*
nc	The number of cluster members
EPSB	The consumed energy per sent byte
EPRB	The consumed energy per received byte

Table 1 Notations

Assumptions

It is assumed that:

- The proposed scheme is built based on a key management scheme that leverages blockchain technology to at least administer its key distribution phase.
- The utilized consensus algorithm is of the Proof-of-Stake (PoS) variety, owing to its heightened energy efficiency when compared to the Proof-of-Work (PoW) consensus [18].
- The used blockchain type is private, as it is deemed more efficient in energy consumption and transaction processing time when compared to the public blockchain.
- The used signature scheme is the Elliptic Curve Digital Signature Algorithm (ECDSA) [19], owing to its widespread adoption and its use in the Bitcoin blockchain [20].

Solution steps

A node can be revoked in the proposed solution under three distinct scenarios: firstly, as a compromised node. Secondly, when the node voluntarily exits the network, and thirdly, when the node's battery power is completely drained.

Compromised node

A compromised node is one that an adversary has compromised. In such cases, the node becomes a security threat to the network, necessitating the revocation of all its sensitive data, including cryptographic secrets and keys. The proposed solution enables the revocation of a compromised node through the blockchain and key management layers. The process is initiated by a Gateway, a blockchain participant, who analyzes the node's behavior on the blockchain ledger to detect malicious activity. It creates a revocation transaction containing the node's information, including its identifier and revocation time, and the reason for revocation. This transaction is then signed and broadcast to the blockchain network. Each blockchain participant that receives the transaction validates it by executing the appropriate smart contract, checking if the node already belongs to the network. If the node is not found, the transaction is rejected. If malicious activity is detected, the transaction is accepted and added to the blockchain ledger. Once added, each blockchain participant checks for any nodes linked to the compromised node and broadcasts a revocation message to them, instructing them to erase any cryptographic keys of the revoked node. Linked nodes reply with an acknowledgment message upon completion of this task. If the node is not found to be malicious, the transaction is rejected and not added to the blockchain ledger.The compromised node revocation process is illustrated in Figure 2.

Leaving node

A leaving node is one that voluntarily exits the network, perhaps to conserve battery power. Although not a security threat, the node's cryptographic keys and sensitive data must still be revoked to prevent future attacks. The leaving node initiates this process by sending a revocation message to its linked gateway. The gateway then cre- ates, signs, and broadcasts a revocation transaction containing the node's information to the blockchain network. Each blockchain participant validates this transaction by executing the associated smart contract. If the node is not found in the network, the transaction is rejected. If found, the transaction is added to the blockchain ledger, and Fig. 2 Flowchart of compromised node revocation process.each blockchain participant checks for any nodes linked to the leaving node, instruct-ing them to erase its cryptographic keys. Acknowledgment messages are exchanged upon completion of these steps.The leaving node revocation process is illustrated in Figure 3.

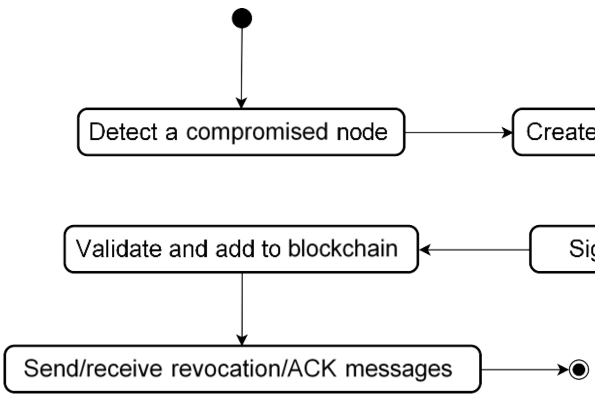
Detect a compromised node
Create a revoke transaction
Sign and broadcast
Validate and add to blockchain
Send/receive revocation/ACK messages

Fig. 3 Flowchart of leaving node revocation process

Drained node

A drained node is one whose battery power is completely depleted. Similar to a leaving node, it is not considered a security threat, but its cryptographic keys and sensitive data must still be revoked. The revocation process is initiated by a blockchain par- ticipant who detects that the node's battery is drained by analyzing its last activity. If the node is confirmed as drained, the blockchain participant creates a revoca- tion transaction containing the node's information and broadcasts it to the network. Each blockchain participant validates the transaction by checking specific conditions through a smart contract. If any condition is not met, the transaction is rejected. Oth- erwise, it is added to the blockchain ledger, and each blockchain participant checks for any nodes linked to the drained node, instructing them to erase its cryptographic keys. Acknowledgment messages are then exchanged to confirm the completion of these steps.The drained node revocation process is illustrated in Figure 4.

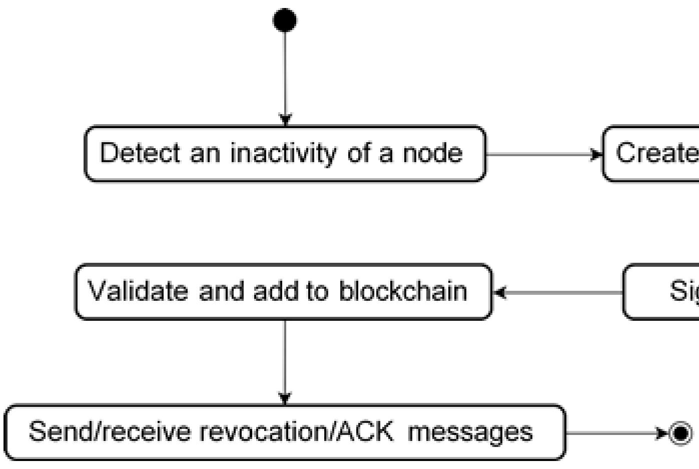
Detect an inactivity of a node
Create a revoke transaction
Sign and broadcast
Validate and add to blockchain
Send/receive revocation/ACK messages

Fig. 4 Flowchart of drained node revocation process.

Evaluation

Security analysis

In this section, we analyze the security of our proposed solution. We first discuss the security of the exchanged messages between the different constrained devices, and then we discuss the security of the data stored on the blockchain ledger.

Security of the exchanged messages

Our proposed solution is assumed to be built on a key management scheme with at least a blockchain-based key agreement phase. It uses its different cryptographic primitives to ensure the security of the exchanged messages between the different constrained devices.

Security of the data stored on the blockchain ledger

Audit and automation are ensured by transparency, immutability, and smart contracts deployed on the blockchain ledger. Furthermore, the blockchain ledger does not store sensitive materials like cryptographic secrets and keys. It only stores the different transactions signed by the different blockchain participants.

Performance analysis

In this section, we present the performance evaluation of our proposed solution. We first present a comparative study of different key revocation schemes. Then, we present the performance evaluation of our proposed solution through experi- ments and simulations regarding communication, computation overheads, and energy consumption.

Comparative study of different key revocation schemes

Tables 2, 3, and 4 present a comparative study regarding the communication over- head and energy consumption of our proposed solution with some of the existing ones, namely, [17], [14, 16], and [10]. The tables present the communication overhead for each node type as the size of the sent and the received bytes during the key revocation process. The energy consumption is calculated as the number of bytes multiplied by each node's energy consumption per byte. Moreover finally, the total energy consump- tion is calculated as the energy consumption of each node multiplied by the number of nodes of that type.

In Table 2, we compare our solution with [17], [14, 16] in the case of a compromised node. As we can see, in the case of compromising a CH in [17], it needs to send one and receive nc − 1 messages by gateway nodes and needs to send one and receive one message for the constrained node. While in [14, 16], the gateway node needs to send one and receive n messages by gateway nodes and needs to send one and receive one

message as [17]. In contrast, in our solution, the gateway node needs to send one and receive ngm messages by the gateway nodes, and send one and receive one message as in [17] and [14, 16]. The number of the involved gateway nodes in the revocation process is the same for all the solutions and higher in the case of [14, 16] in the case of constrained nodes; This makes our solution outperforms [14, 16] and has the same performance as [17] in the case of a compromised node for the gateway nodes.

In Table 3, we compare our solution with [14, 16] and [10] in the case of a leaving node. As we can see, in the case of a leaving node in [14, 16], it needs to send one and receive d messages by gateway nodes and needs to send one and receive one message for the constrained node. While in [10], it needs to send (ns − 1) and receive (n − 1) messages by the leaving node and needs to send one and receive one message for the

other nodes' types. In contrast, the gateway node must send one in our solution and receive (d+1) messages. However, the number of the involved constrained nodes in therevocation process is the smallest in our solution; This makes our solution outperforms [14, 16] and [10] in the case of a leaving node for the constrained nodes

Solution	**Node type**	**Communicatio**	**Energy consumption**	**nodes count**	**Total energy consumption**
[17]	CH	$l_{ms} + (n_c - 1) * lmr$	$EPSB * l_{ms} + EPRB * (n_c - 1) l_{mr}$	1	$EPSB * l_{ms} + EPRB * (n_c - 1) l_{mr}$
	CM	$lms + lmr$	$EPSB * l_{ms} + EPRB * l_{mr}$	n_c	$n_c * (EPSB * l_{ms} + EPRB * l_{mr})$
[14, 16]	Gateway	$l_{ms} + n * l_{mr}$	$EPSB * l_{ms} + n * EPRB * l_{mr}$	1	$EPSB * l_{ms} + n * EPRB * l_{mr}$
	Constrained	$lms + lmr$	$EPSB * l_{ms} + EPRB * l_{mr}$	n	$n * (EPSB * l_{ms} + EPRB * l_{mr})$
Our solution	Gateway	$l_{ms} + n * l_{mr}$	$EPSB * l_{ms} + n * EPRB * l_{mr}$	1	$EPSB * l_{ms} + n * EPRB * l_{mr}$
	Constrained	$lms + lmr$	$EPSB * l_{ms} + EPRB * l_{mr}$	ngm	$n_{gm} * (EPSB * l_{ms} + EPRB * l_{mr})$

Table 2 Comparison of our solution with some existing solutions for compromised node detection.

Solution	Node type	Communication	Energy consumption	nodes count	Total energy consumption
[14, 16]	Leaving	$l_{ms} + d * l_{mr}$	$EPSB * l_{ms} + EPRB * d * l_{mr}$	1	$EPSB * l_{ms} + EPRB * d * l_{mr}$
	Others	$l_{ms} + l_{mr}$	$EPSB * l_{ms} + EPRB * l_{mr}$	d	$d * (EPSB * l_{ms} + EPRB * l_{mr})$
[10]	Leaving	$(n_s - 1)l_{ms} + (n - 1)l_{mr}$	$(n_s - 1)l_{ms} * EPSB + (n - 1)l_{mr} * EPRB$	1	$(n_s - 1)l_{ms} * EPSB + (n - 1)l_{mr} * EPRB$
	Others	$l_{ms} + l_{mr}$	$EPSB * l_{ms} + EPRB * l_{mr}$	n	$n * (EPSB * l_{ms} + EPRB * l_{mr})$
Our solution	Gateway	$l_{ms} + (d + 1)l_{mr}$	$EPSB * l_{ms} + EPRB *(d+1)l_{mr}$	1	$EPSB * l_{ms} + EPRB *(d+1)l_{mr}$
	Others	$l_{ms} + l_{mr}$	$EPSB * l_{ms} + EPRB * l_{mr}$	ngm	$n_{gm} * (EPSB * l_{ms} + EPRB * l_{mr})$

Table 3 Comparison of our solution with some of the existing solutions in the case of a leaving node.

In Table 4, we compare our solution with [14, 16] in the case of a drained node. As we can see, in the case of a drained node in [14, 16], it needs to send one and receive (n − 1) messages by gateway nodes and needs to send one and receive n − 1 messages for the constrained node. While in our solution, the gateway node needs to send one and receive (ngm) messages, and the constrained node needs to send one and receive one message. Furthermore, the number of the involved constrained nodes in the revocation process is the smallest in our solution and the same for the gateway nodes; This makes our solution outperforms [14, 16] in the case of a drained node for the constrained and the gateway nodes.

Solution	Node type	Communication	Energy consumption	nodes count	Total energy consumption
[14, 16]	Gateway	$l_{ms} + (n - 1)l_{mr}$	$EPSB * l_{ms} + EPRB *(n-1)* l_{mr}$	1	$EPSB * l_{ms} + EPRB * (n - 1) * l_{mr}$
	Constrained	$l_{ms} + (n - 1)l_{mr}$	$EPSB * l_{ms} + EPRB * l_{mr}$	n	$n * (EPSB * l_{ms} + EPRB * l_{mr})$
Our solution	Gateway	$l_{ms} + (ngm)l_{mr}$	$EPSB * l_{ms} + EPRB * (n_{gm})l_{mr}$	1	$EPSB * l_{ms} + EPRB * (n_{gm})l_{mr}$
	Constrained	$l_{ms} + l_{mr}$	$EPSB * l_{ms} + EPRB * l_{mr}$	ngm	$n_{gm} * (EPSB * l_{ms} + EPRB * l_{mr})$

Table 4 Comparison of our solution with some of the existing solutions in the case of a drained node.

Experimental results (Simulation)

This section showcases the outcomes of our proposed solution through simulation. We utilized Rust programming language to implement the simulation for our solution and the two other ones [14] and [16]. We considered the network nodes' total com- munication overhead and energy consumption. The simulation was conducted with 100 randomly deployed nodes. The rate of the gateway nodes is 10% of the total net- work nodes number. The number of neighbors of each node varies between 10 to 15. We ran the simulation 1000 times for the three scenarios: compromised, leaving, and drained nodes. The energy consumption model used in this simulation is based on [21], which considers the EPSB and EPRB factors as follows: EPSB = 59.2µJ and EPRB = 28.6µJ for the transmission and reception of one byte, respectively.

Communication Overhead

Figure 5, Figure 6, and Figure 7 demonstrate the communication overhead required by all the limited-resource nodes in the network in the three cases of compromised, left, and drained nodes, respectively.

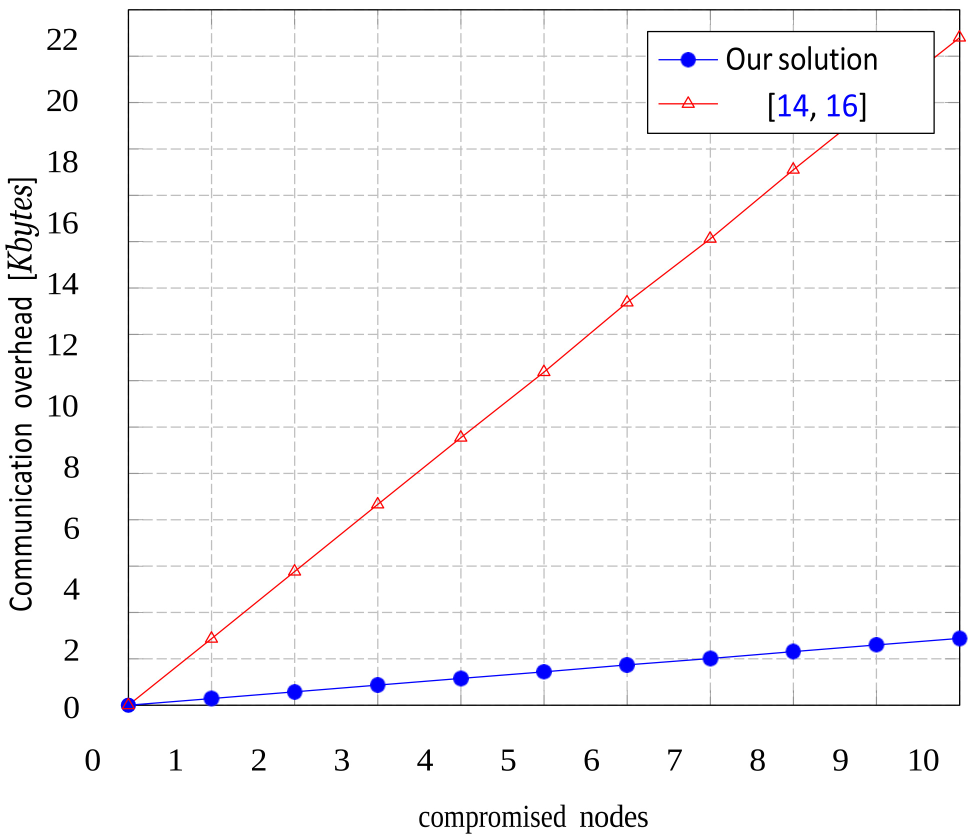

Fig. 5 Communication overhead required by all the constrained nodes in the network in the case of compromised nodes.

The efficiency of our solution surpasses that of current solutions in terms of com- munication overhead, as evidenced by the results; this is due to the decentralization of Blockchain and the delegation of most communication tasks to the gateway nodes. Each gateway is responsible for communicating only with its attached constrained nodes, unlike previous studies such as [14] and [16]. Furthermore, most communica- tion is carried out by the constrained nodes, with the gateway communicating with all network nodes, thereby increasing the communication overhead of the constrained devices. In our proposal, in the event of a leaving node, the node communicates only with its gateway, which communicates with the other nodes to notify them, result- ing in the sending and receiving of only one message by each of them. In contrast, in [14] and [16], the leaving node must send and receive d ACK messages, increasing communication overhead.

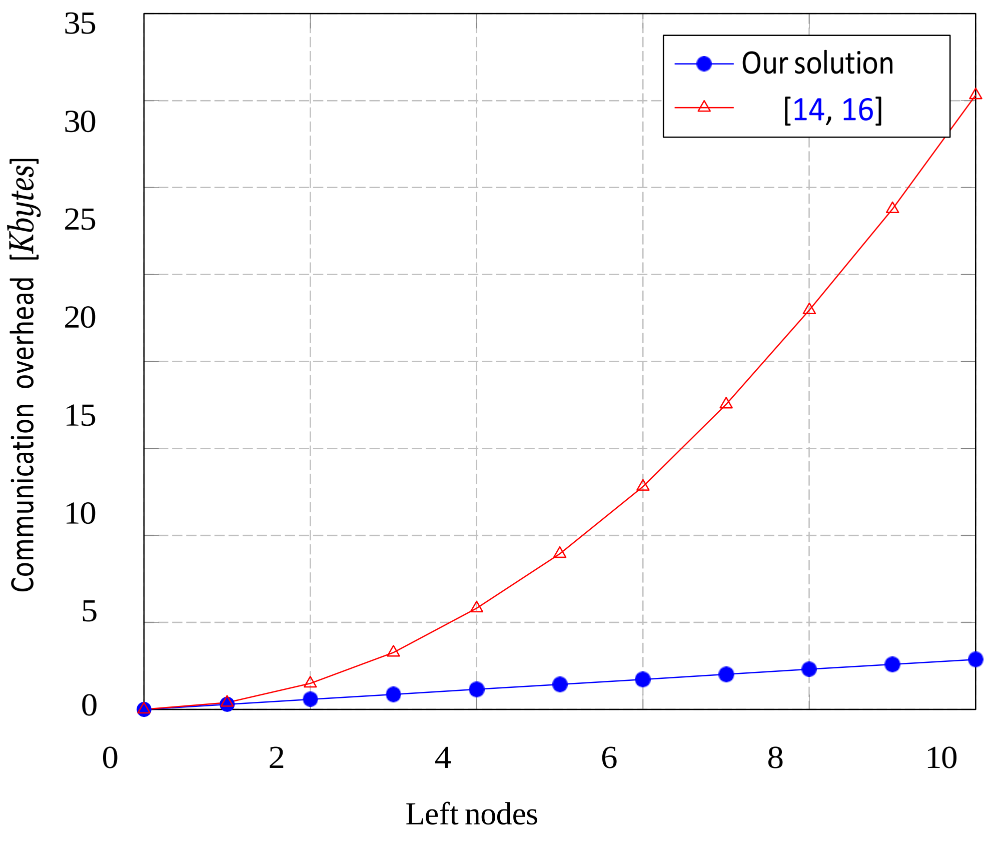

Fig. 7 Communication overhead required by all the constrained nodes in the network in the case of drained nodes.

Energy Consumption

The evaluation of energy consumption in IoT networks stands as a important aspect in the effectiveness and the feasibility of security solutions. Moreover, IoT networks often comprise resource-constrained devices with limited processing power and battery life. Optimizing energy consumption can pave the way for more efficient and durable IoT Networks.

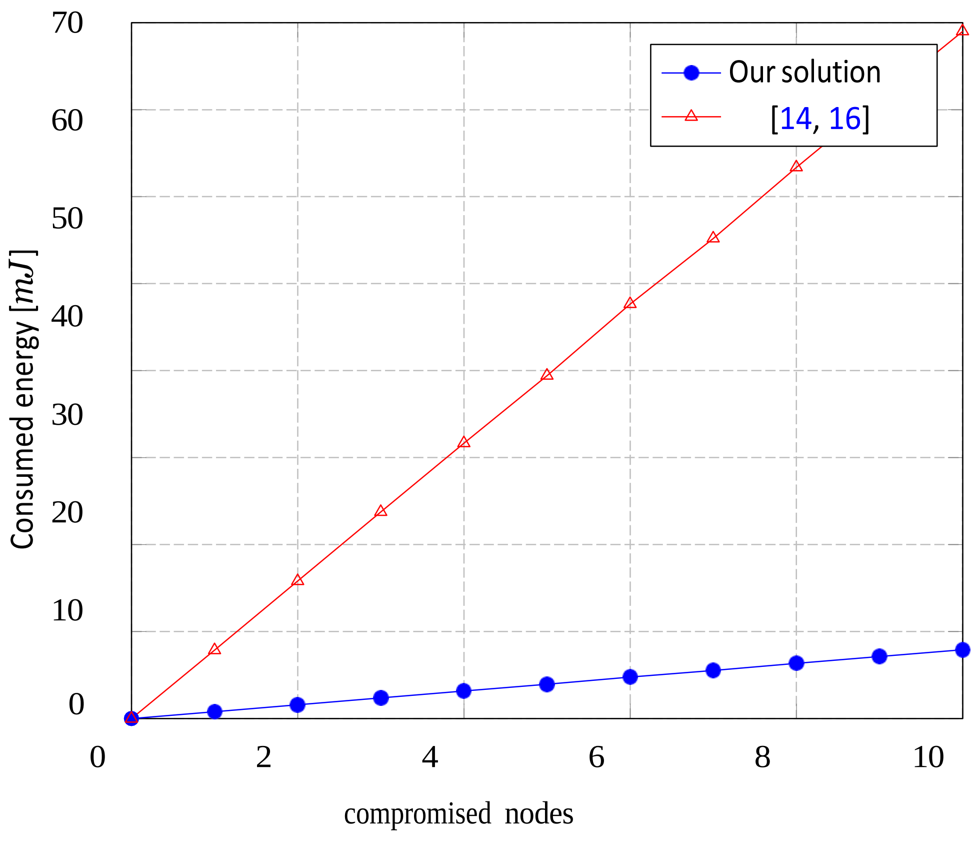

Fig. 8 Energy consumption of communication between all the constrained nodes in the network in the case of compromised nodes.

The results of the simulation on the consumed energy by the constrained nodes in the three different scenarios are presented in Figure 8, Figure 9, and Figure 10. The number of affected nodes ranges from 1 to 10. As we see clearly in the figures, our approach is more efficient than existing solutions regarding consumed energy by the constrained nodes in all scenarios; This is due to the involvement of fewer constrained nodes in communication and delegating communication to gateway nodes. It is worth noting that Figure 8, Figure 9, and Figure 10 share similarities with Figure 5, Figure 6, and Figure 7, as the communication energy consumption is directly related to the number and the size of exchanged messages.

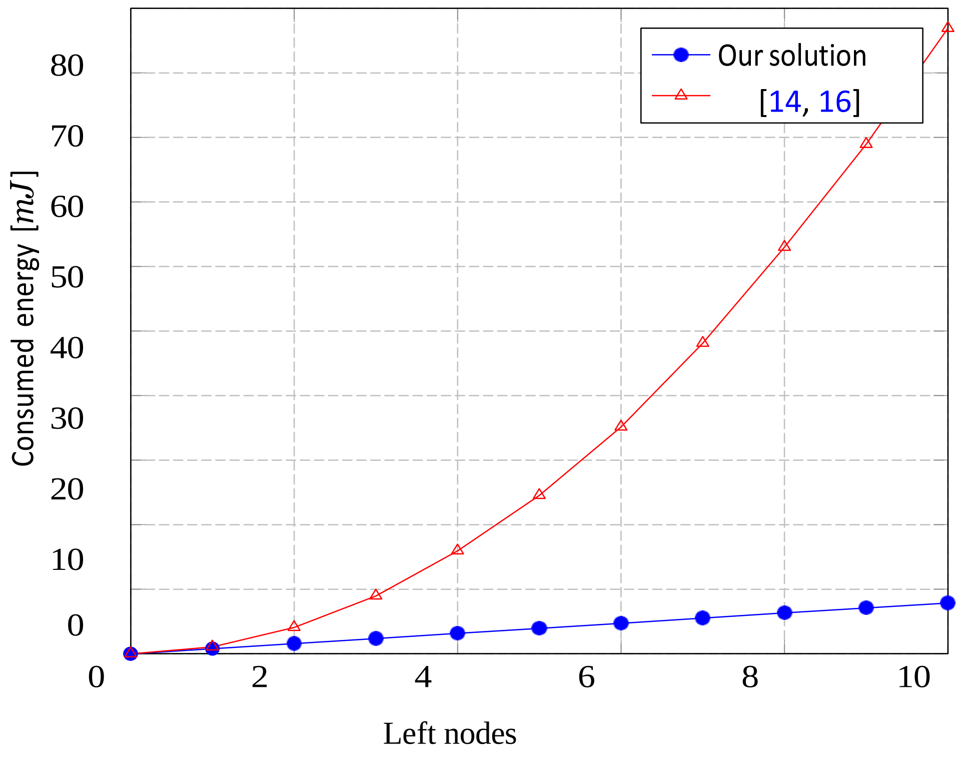

Fig. 9 Energy consumption of communication between all the constrained nodes in the network in the case of left nodes.

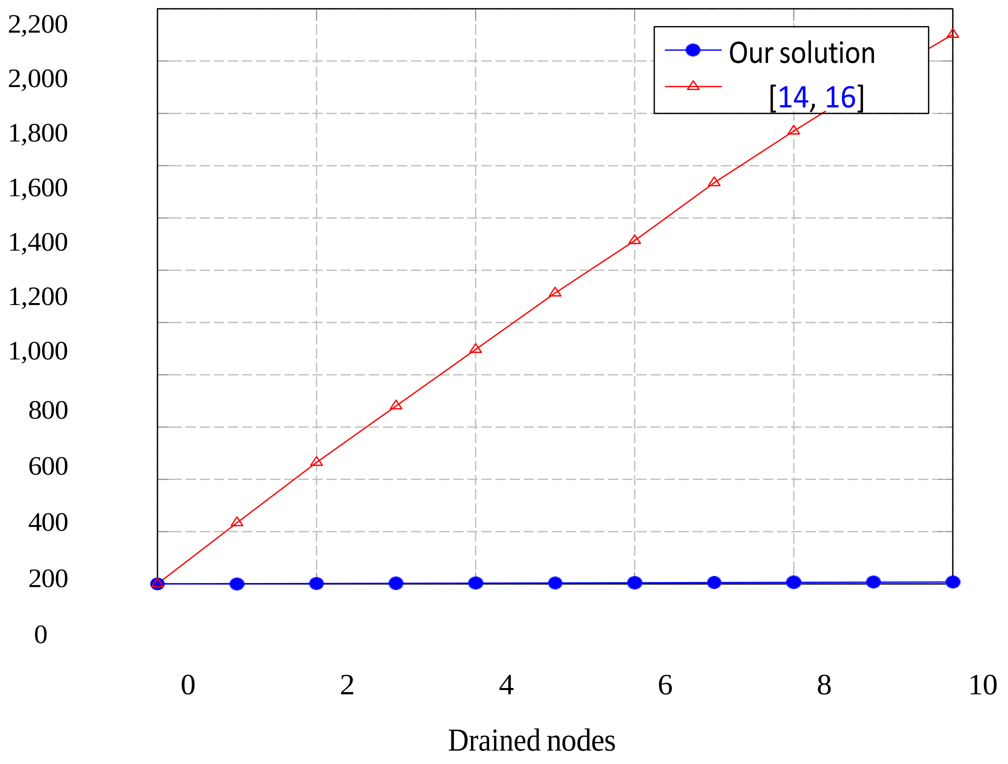

Fig. 10 Energy consumption of communication between all the constrained nodes in the network in the case of drained nodes.These simulation results substantiate the robustness of our proposed solution and present its potential to significantly enhance energy efficiency in real-world deployment scenarios.

CONCLUSION

The issue of key revocation is crucial in any key management solution. This work pro- poses a blockchain and smart contract-based approach to tackle the key revocation problem in IoT networks. The proposed solution benefits from blockchain technology: transparency and automation features, which increases the security reduces limitations like the compromission of a blockchain participant. Moreover, it decentralizes and reduces communication overhead and energy consumption. Additionally, the security analysis and performance evaluation demonstrate that the security of the proposed solution is proportional to that of the underlying scheme. Moreover, the solution does not store any key materials in the blockchain. Simulation results indicate that our proposal outperforms existing ones regarding efficiency. However, it is essential to acknowledge that the security of the proposed solution is directly related to the secu- rity of the base scheme. We plan to extend the proposed solution in future work to include other phases of key management, such as key distribution and node addition, and tackle their challenges.

REFERENCES

[1] Messai, M.-L., Seba, H.: A survey of key management schemes in multi-phase wireless sensor networks. Computer Networks 105, 60–74 (2016)
[2] Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. Decentralized business review, 21260 (2008)
[3] Ma, M., Shi, G., Li, F.: Privacy-oriented blockchain-based distributed key man- agement architecture for hierarchical access control in the iot scenario. IEEE access 7, 34045–34059 (2019)
[4] Viriyasitavat, W., Hoonsopon, D.: Blockchain characteristics and consensus in modern business processes. Journal of Industrial Information Integration 13, 32– 39 (2019)
[5] Vasin, P.: Blackcoin's proof-of-stake protocol v2. URL: https://blackcoin.org/blackcoin-pos-protocol-v2-whitepaper.pdf 71 (2014)
[6] Larimer, D.: Delegated proof-of-stake (dpos). Bitshare whitepaper 81, 85 (2014)
[7] Castro, M., Liskov, B., et al.: Practical byzantine fault tolerance. In: OsDI, vol. 99, pp. 173–186 (1999)
[8] Szabo, N.: Formalizing and securing relationships on public networks. First monday (1997)
[9] Lei, A., Cruickshank, H., Cao, Y., Asuquo, P., Ogah, C.P.A., Sun, Z.: Blockchain-based dynamic key management for heterogeneous intelligent transportation systems. IEEE Internet of Things Journal 4(6), 1832–1843 (2017)
[10] Kandi, M.A., Kouicem, D.E., Doudou, M., Lakhlef, H., Bouabdallah, A., Challal, Y.: A decentralized blockchain-based key management protocol for heterogeneous and dynamic iot devices. Computer Communications 191, 11–25 (2022)

[11] Hameedi, S.S., Bayat, O.: Improving iot data security and integrity using lightweight blockchain dynamic table. Applied Sciences 12(18), 9377 (2022)

[12] Liu, Q., Luo, L., Wang, J., Li, W., Liu, R., Yu, M.: Key management scheme of distributed iot devices based on blockchains. IET Communications(2023