Secure SLA Management Using Smart Contracts for SDN-Enabled WSN

Emre Karakoç[1*], and Celal Çeken[2]
[1] Faculty of Computer and Information Sciences, Computer Engineering Department, Sakarya University, Sakarya, Postal Code: 54187, Turkey
[e-mail: emre.karakoc3@ogr.sakarya.edu.tr]
[2] Artificial Intelligence Systems Research and Application Centre, Sakarya University
Sakarya, Postal Code: 54187, Turkey
[e-mail: celalceken@sakarya.edu.tr]
[*]Corresponding author: Emre Karakoç

Received June 6, 2023; revised August 25, 2023; accepted October 20, 2023; published November 30, 2023

Abstract

The rapid evolution of the IoT has paved the way for new opportunities in smart city domains, including e-health, smart homes, and precision agriculture. However, this proliferation of services demands effective SLAs between customers and service providers, especially for critical services. Difficulties arise in maintaining the integrity of such agreements, especially in vulnerable wireless environments. This study proposes a novel SLA management model that uses an SDN-Enabled WSN consisting of wireless nodes to interact with smart contracts in a straightforward manner. The proposed model ensures the persistence of network metrics and SLA provisions through smart contracts, eliminating the need for intermediaries to audit payment and compensation procedures. The reliability and verifiability of the data prevents doubts from the contracting parties. To meet the high-performance requirements of the blockchain in the proposed model, low-cost algorithms have been developed for implementing blockchain technology in wireless sensor networks with low-energy and low-capacity nodes. Furthermore, a cryptographic signature control code is generated by wireless nodes using the in-memory private key and the dynamic random key from the smart contract at runtime to prevent tampering with data transmitted over the network. This control code enables the verification of end-to-end data signatures. The efficient generation of dynamic keys at runtime is ensured by the flexible and high-performance infrastructure of the SDN architecture.

Keywords: Blockchain, IoT, SDN, Smart Contract, SLA, WSN.

1. Introduction

Internet of Things (IoT) offers new opportunities for businesses and individuals in smart cities, enabling technological advancements in areas such as building energy management systems, smart factories, precision agriculture, e-health systems, and smart homes [1]. Some of these services are highly sensitive and critical to the customers' benefits. Potential problems that may arise during the service period can be predicted in advance, and appropriate actions can be taken to address them and inform the customer accordingly. These processes are regulated through contracts known as Service Level Agreements (SLAs) in the field of informatics. An SLA is an official agreement established through mutual understanding between the customer and the service provider. It can be independently defined and implemented in any customer case by utilizing IT network infrastructure. The SLA outlines the customer's expectations and their obligations towards the service provider. It also plays a crucial role in identifying key performance criteria for the service, such as network availability, jitter, bandwidth, latency, error rate, packet loss, etc. [2]. Additionally, the SLA can establish the necessary procedures for monitoring and reporting issues, specify time limits, and define appropriate penalties in case of violations [3]. It is of utmost importance that these matters are documented in SLAs so that, in the event of an SLA breach (e.g., failure to meet performance criteria), the customer can claim damages from the service provider as compensation. However, even if the terms of compensation are clearly stated in the SLA, the process itself can be uncertain due to potential dishonest actions by the involved parties [4]. When a customer submits a complaint regarding an SLA breach, they may need to provide supporting evidence (data) to validate their claim. Conversely, the service provider may be required to furnish evidence demonstrating their compliance with the SLA. Moreover, in case of disagreement, the customer may choose not to pay the previously agreed-upon amount for the purchased service, or the service provider may decide not to provide the defined compensation.

The wireless network environment, on the other hand, can be more vulnerable to tampering and more challenging to manage, especially when it comes to sensitive data transmission over the air compared to the wired network environment. For example, in certain smart city applications that utilize a Wireless Sensor Network (WSN) with a dynamic infrastructure, real-time data transfer may be requested by the customer. However, there can be unexpected issues such as energy depletion during runtime or network manipulation by attackers, given that the wireless network nodes are low-energy devices. As a result, users of the wireless network environment may require a network that meets their own standards or take responsibility for their own security measures. To meet these standards, the service provider managing the network infrastructure may need to regularly monitor network changes, handle all network layers simultaneously, and ensure traffic load balancing when new clients connect to the nodes [5]. However, customers may also request proof of the service reliability from the service provider.

Considering the potential risks in the wireless environment, it appears that there may be more conflicts when using an SLA in the wireless network compared to the wired environment. As conflicts increase, both the service provider and the customer may have to undergo a more costly and bureaucratic process. They may need to rely on a Trusted Third Party (TTP) company, such as a bank or financial service, to facilitate the required payments. Therefore, there is a need for a cost-effective and straightforward solution that ensures accurate payment from both parties. We believe that blockchain technology, based on the principles of immutability and smart contracts (SC), can help address these issues by providing a reliable payment system and secure storage. In this regard, we propose a smart contract-Based Model

that safeguards the Wireless Network's Quality of Service (QoS) metrics used to express SLAs against tampering, utilizing a blockchain architecture like Ethereum, which offers a flexible development infrastructure. This eliminates the need for an intermediary institution to ensure service fees and compensation payments. Additionally, service providers will be able to demonstrate their reliability through the smart contract. We believe that this architecture holds promising potential.

In this study, we have resolved the disputes that may arise between the parties during the execution of the SLA specific to the WSN environment and the situations that may cause loss of trust with a new model based on smart contracts. In this context, we propose a model that enables customers to monitor the promised network metrics and facilitates the automatic execution of the SLA payment and compensation process between the customer and the service provider without the need for a third party. We have ensured the immutability of SLA agreement rules by storing them in the blockchain network via smart contract in the model we propose. We also evaluated the possibility of end-to-end tampering of data between WSN and smart contract. In this context, we created a hash value with signature quality of the data transmitted from WSN nodes to the smart contract by using the private key and dynamic key binary keys generated on the smart contract. Thus, we have eliminated possible malicious attacks by passing the data transmitted to the smart contract and the signature value of the data through a control function. As a result, all SLA outputs are considered reliable for both the customer and the service provider according to proposed model. The Ethereum infrastructure and Solidity language are utilized for implementing the smart contracts. Contributions of this study can be listed as follows:

- Although various models utilizing smart contracts have recently been proposed in the literature, to the best knowledge of the authors, the proposed model is the first study to demonstrate the feasibility of SLA management using smart contracts in WSN architecture. Blockchain is a hardware-intensive technology. Therefore, similar studies that employ smart contract-based SLA management in IoT have utilized single board computers or edge devices, as implementing this infrastructure on low-energy and low-performance wireless nodes can be challenging. In contrast to other studies, our proposed model developed a blockchain infrastructure specifically applicable to wireless nodes.
- We have developed a smart contract-based model that enables control over network metrics in low-energy and performance wireless nodes. This flexible network structure formed by wireless nodes is expected to find applications in various areas of smart city applications, offering a fresh perspective.
- While smart contracts provide immutability and impartiality to applications, concerns about third-party insecurity arise when data is sent to the smart contract. The proposed model overcomes this issue by implementing a dual-sided control mechanism, ensuring verifiability of the data and confirming the amount of data sent to the contract. This safeguards both the service provider and the customer against potential malicious incidents.
- In this study, we have proposed a global ecosystem that encompasses all stages, from production to sales and from sales to customers, including the supply chain of Wireless Sensor Nodes. By incorporating the supply chain into the model, we establish confidence in the authenticity of the devices throughout the entire lifecycle, from the production phase to customer usage.

The remainder of the paper is organized as follows: Section 2 presents related studies found in the literature. Section 3 explains the underlying technologies and subsystems of the proposed architecture. The overall properties and design stages of the proposed model, along with related algorithms, are provided in Section 4. In Section 5, an example scenario of a smart building incorporating the proposed model is implemented, followed by performance evaluation. With the case study, the aim of this article was to experimentally demonstrate the feasibility of the proposed model and to emphasize the solution we produced against customer requests in the face of a possible scenario in the real world. The paper concludes with the final section, which offers concluding remarks and suggestions for further studies.

2. Related Works

There are several studies in the literature suggesting SLA management using blockchain and smart contracts. In a couple of studies [3][6], a model powered by smart contracts was proposed to automate the SLA monitoring process on Pop-Es (Point of Presence-ES / RNP), which offers various network maintenance, management, planning, and development services. The smart contract they developed was hidden on the Ropsten network, a public Ethereum Test Network. Additionally, they stored files with high storage costs from the blockchain network in a decentralized file system (IPFS - Interplanetary File System) within their proposed system. Another study developed an Ethereum-based smart contract that automates the potential compensation process between the customer and the service provider [7]. Their proposed model required both the customer and the service provider to agree to the terms of the SLA included in the smart contract from the outset and deploy it onto the blockchain network. In this study, the authors simulated SLA conditions by using response times based on a php site hosted on a test web server.

A different framework for managing smart contracts and dynamic SLAs in a distributed manner involves the management of data collection and verification, as well as changes in network quality and the service payment system [8]. The authors utilized two different networks: off-chain and blockchain. The former was employed for computationally intensive tasks related to SLA management, while the latter was used for executing smart contracts. Building upon previous work [9], they introduced a decision-making structure by incorporating Oracle DB into the SLA application. In another study, the authors proposed using smart contracts instead of a secure intermediary platform for managing SLAs related to cloud computing communication standards [10]. They also addressed the challenge of verifying the correctness of data before recording potential violations in the blockchain. To address this, they proposed a witness model based on game theory, wherein witnesses were committed to receiving a specific fee as an incentive to ensure their credibility. The results of another study [11] indicated that centrally managed smart contracts were susceptible to manipulation, thereby compromising accessibility and data integrity. In response, the authors designed a decentralized approach where smart contracts could be grouped, and common variables or data could interact based on collectively determined shared values. They proposed a consensus mechanism utilizing asynchronous voting to achieve consensus among multiple members, and when a sufficient number of votes were received, consensus was reached. This approach aimed to ensure the reliability of data transmission between the client and the smart contract. Lastly, a similar study argued that managing the compensation process of Service Level Agreements (SLAs) is a complex and bureaucratic task [12]. They emphasized the expenditure of capital and effort in resolving breaches that occur within launched SLAs, with transactions typically conducted manually. As a solution, they employed smart contracts for

managing SLA violations due to their reliable structure that minimizes the need for third-party intervention.

Blockchain and smart contracts have been utilized in several studies for network slicing and channel allocation management. In [13], the authors assert that the inherent features of blockchain can support various processes in the top-tier level of 5G network slicing management. They explain how the system can serve as a manageable platform for virtualized network functions in 5G using blockchain and smart contracts. Similarly, another study proposed a novel network slicing technology called NSB chain, which leverages blockchain to meet the requirements of new business models without relying on traditional network sharing agreements [14]. In this model, smart contracts are employed to automate and scale the allocation of network resources among tenants. Addressing possible security issues in spectrum distribution, another study [15] emphasized the need for a reliable intermediary to safeguard the security and privacy of operators' spectrum sharing. To address this, they introduced a framework called Multi-OPs Spectrum Sharing (MOSS), which leverages smart contracts to provide an auction and marketplace infrastructure, enabling independent spectrum sharing among wireless networks.

In the literature, there are studies that employ smart contracts for authentication and data sharing management in the IoT network. One of these studies proposed the creation of a unique and global digital identity for IoT devices throughout their lifecycle, which is stored in the blockchain network [16]. Another study focused on using smart contracts to facilitate transaction coordination and automate monetary transactions between IoT devices [17]. Similarly, it was highlighted in a study that access management in IoT systems should be distributed efficiently when multiple IoT devices are connected, and a scalable architecture based on blockchain and smart contracts was proposed to address this challenge [18]. Another study adopted a similar approach, introducing a multi-tier management mechanism consisting of hub-based and pool-based layers to reduce smart contract processing costs in the management of numerous IoT devices [19].

A data lease system utilizing smart contracts was proposed in previous works [20]. The authors employed smart contracts and blockchain as an alternative to traditional models, aiming to enhance the data integrity and security requirements. Another study introduced a model based on smart contracts that incorporates Access Control Contracts (ACC), Judge Agreement (JC), and Registration Agreement (RC) to establish distributed and reliable access control for IoT systems [21]. ACC manages dynamic access permissions by employing authentication methods and defined rules. JC receives and evaluates misconduct information (e.g., too many erroneous entries) from ACCs and applies a behavioral assessment methodology, imposing penalties if necessary. This contributes to the functioning of ACCs. RC records the evaluation outputs of ACC in their respective smart contracts. In a different research study, a smart contract-based smart home system model was proposed where data from IoT sensors in emergency situations is transmitted to the Home Service Provider (HSP), which is also developed based on smart contracts [22]. The researchers utilized the meteor framework for communication between the HSP and the host, implementing One Time Password (OTP) as a measure to protect against DDoS attacks and other types of security threats. Another study addressing sensor data emphasized the need for a trusted third party to ensure the security and traceability of sensor data in IoT systems [23]. The proposed solution involved a smart contract and a DLT-based model to establish this trust. They developed a decentralized application (DApp) where only the checksum values of the data from the sensors could be verified. In the context of the sharing economy, a different approach to smart contract-based IoT has been proposed [24]. While many sharing economy platforms implement rating

systems to provide reliability, individual risks still exist even if users present a reliable profile. Therefore, the authors argue that utilizing the decentralized structure of smart contracts is a reasonable approach to provide reliable infrastructure for the sharing economy. They applied the SLA concept by developing an exemplary model for smart contracts in the sharing economy, automating the SLA specifications and the SLA life cycle, and ensuring transparency of all rules.

While some studies guarantee violations to be transferred to the smart contract without tampering, they are quite complex in terms of applicability. Some studies in this conjuncture have provided the opportunity to rent the recorded data later. In this case, it will not be possible for the consumer to meet the real-time data need. In addition, since blockchain technology includes high performance computations, it is very difficult to apply to low energy devices such as sensor nodes [1]. For this reason, a single board computer such as Raspberry Pi has been used in almost all IoT-based studies. Researchers can test their algorithmic methods on these small computers which are practical for fast demonstration that they expect. Unfortunately, although these approaches are theoretically possible, they cannot be applied in real-world systems such as mesh networks formed only by wireless sensors. Thus, our study, unlike other studies, has modeled an SDN-Enabled WSN consisting only of wireless sensors in order to show that the smart contract can be applied in a non-complex way on the WSN.

3. Background

3.1 SLA (Service Level Agreement)

A Service Level Agreement (SLA) is an agreement between the service provider and the customer that includes service qualifications. These qualities determine the quality standards of the service provided by the service provider to the customer. Criteria such as the nature and quantity of services provided under an SLA, backup, support, service delivery time and problem resolution times may be included to service qualifications. In the proposed model architecture, we deployed a WSN structure, therefore, some performance metrics related to WSN have been considered when building our model. Some of the WSN parameters that can be used for SLA contracts are as follows [25].

- Service availability
- Down-Time
- Network Failure Rate
- Measurement Period
- Latency
- Number of Nodes in Network
- Energy consumed

In the model we propose, we establish an infrastructure that enables the control of these parameters. If the customer desires, they can include the specified parameters in the SLA Contract. Furthermore, SLAs encompass various stages, forming the SLA life cycle. **Fig. 1** illustrates the life cycle of an SLA and indicates which stages are covered by our proposed model. Our model consists of 5 stages, including the following: SLA definition, acceptance of the SLA, violation monitoring, termination of the SLA, and enforcement of penalties. In our proposed model, it is the customer who determines the criteria, and if the service provider agrees to the conditions set by the customer, the SLA smart contract is created. The creation

of the smart contract is the responsibility of the service provider. Once the customer approves the contract, the process commences. At regular intervals during operational hours, the quality of data transfer in the wireless network is checked to ensure it surpasses the specified threshold values. These checks are recorded, and upon completion of the data transfer, the penalty fee for total violations is calculated and returned to the customer, thereby automating the SLA life cycle.

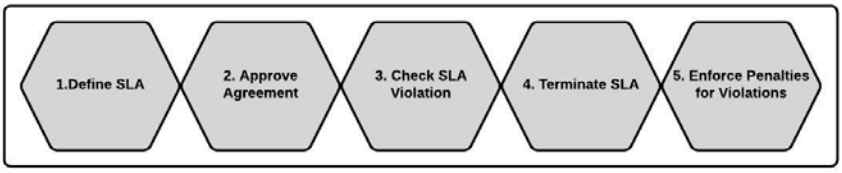

Fig. 1. SLA Life Cycle for The Proposed Model

3.2 SDN-Enabled WSN Architecture

In this study, we utilized a WSN with SDN capability, which was modeled using Riverbed Modeler Simulation Software. Further information about this architecture can be found in [26], [27], and [28]. The SDN-enabled WSN comprises low-energy nodes (ED) that can function as sources, destinations, or routers. Additionally, the network includes another node type called SDN Controller (SDNC), responsible for control and management operations, and equipped with a Dijkstra's-based path discovery mechanism. One of the primary tasks of the SDNC, which possesses all the network intelligence, is to determine the optimal path between the source and destination by considering the remaining energy of the nodes along the path and the Signal-to-Noise Ratio (SNR) of the neighboring nodes. The low-energy nodes are equipped with an application layer, enabling them to act as Sinks and forward incoming data to the internet environment. The WSAN Flow Protocol, extensively described in [27], facilitates control messaging between the SDNC and ED nodes.

3.3 Ethereum and smart contracts

The blockchain system, initially introduced by Satoshi Nakamoto in 2008 as the architecture for the Bitcoin cryptocurrency [29], is a cryptographic proof-based electronic payment system that enables direct transactions between two parties without the need for a trusted third party [29]. Blockchain can be conceptualized as a ledger where records are organized into time-stamped blocks, each having a unique hash code. Each block contains the hash code of the previous block, creating a chain of blocks [30]. When a new block is added, the node responsible for its completion notifies all other nodes, and the mapping is finalized. The distributed nature of the system ensures that attempts to attack the system in one or multiple locations do not compromise its reliability. Therefore, mutual trust between nodes in the blockchain is not required. Smart contracts, stored immutably on the blockchain network, are code snippets that execute within the network [31]. These code snippets are secured by the blockchain network and are resistant to external manipulation. Smart contracts can store data, execute decision-making processes, transfer money to other accounts, and interact with other contracts [18]. Smart contracts are permanently established on the blockchain by their owners.

Numerous cryptocurrency platforms utilize blockchain infrastructure, with Ethereum being one of the most prominent examples. Ethereum is an open-source cryptocurrency architecture based on blockchain that incorporates the smart contract protocol. It supports an updated version of the blockchain consensus protocol through transaction-based state transitions [32].

Smart contracts in Ethereum are executed on the Ethereum Virtual Machine (EVM), a Turing-complete software operating within the Ethereum network. EVM enables the execution of programs written in any programming language and provides sufficient time for their execution. Each transaction operation on Ethereum incurs a specific operation fee known as 'Gas'. Therefore, when designing an algorithmic method within a smart contract, it is crucial to optimize transactions and variables to minimize transaction fees. Moreover, Ethereum allows developers to create and deploy decentralized applications (DApps). DApps offer significant advantages by eliminating intermediaries in various industries, thanks to the Ethereum platform.

4. The Proposed SLA Management Platform and Algorithms

4.1 System Overview

The model we propose encompasses the manufacturer, service provider, and customer, covering the entire process from the production to the sale of all nodes that will constitute the Wireless Sensor Network (WSN). In general, the process can be summarized as follows: The manufacturer produces devices that incorporate private keys, and each device produced is transferred to a specific smart contract defined by the manufacturer, with the keys stored in the blockchain. The service provider purchases these devices and performs customer-specific installations using them. The service provider establishes specific Service Level Agreement (SLA) terms for each customer, and once both parties agree to these terms, they are incorporated into the smart contract and deployed onto the blockchain network, initiating the smart contract. Once the data reaches the End Device, which we refer to as the End Device, it is transmitted to the smart contract Helper (SCH) along with certain parameters.

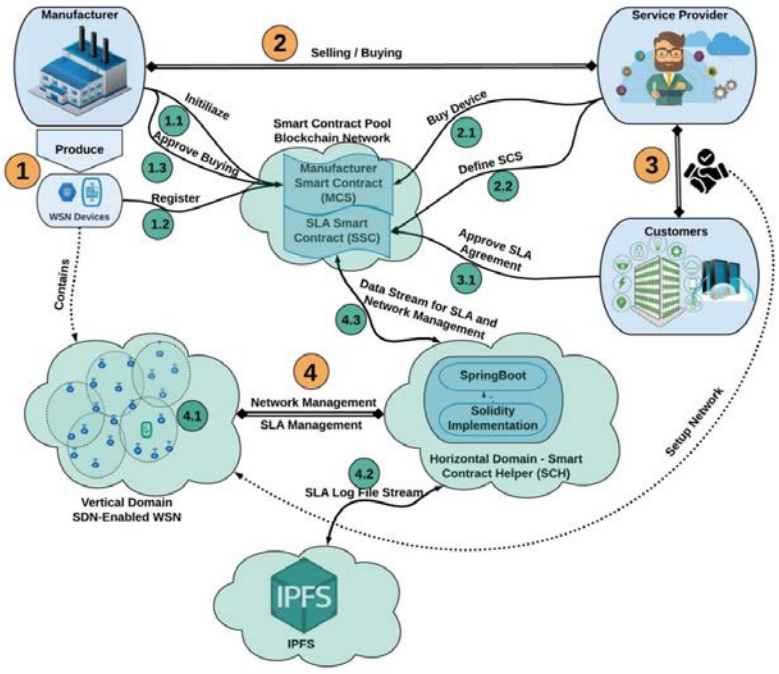

Fig. 2. Components of the proposed SLA Management Platform

SCH forwards the incoming data and parameters to SSC (SLA smart contract). SSC initially verifies the incoming data using MSC. MSC validates the verification process using the previously reported private keys from the manufacturer. If the verification process is successful, SSC checks whether the promised network metrics are being met. In case of a violation, the penalty amount specified in the SLA is deducted from the deposit. Additionally, if the service provider and customer agree, the data can be asynchronously transmitted through SCH to a data warehouse specified by the customer. The proposed model aims to detect and control the data being transmitted in accordance with the contract, while leaving the data storage methodology to the discretion of the developers. In the proposed model, the process is divided into four stages, as shown in **Fig. 2**.

4.2 Manufacturer Stage (1)

In this process, the manufacturer plays a crucial role in the proposed model. The manufacturer is responsible for ensuring the reliability of the devices, which are the sensing nodes in the WSN. To achieve this, the manufacturer generates a private key for each device and securely stores them in a tamper-proof memory section within the manufactured devices. The main purpose of these keys is to verify the accuracy of all the information coming from within the network. The manufacturer stores these device-specific keys in the blockchain network using a MSC (manufacturer smart contract) that is created and deployed exclusively for each manufacturer (**Fig. 2**, step 1.2). The initiation of these smart contracts is solely performed by the manufacturer. The MSC includes the production date of the device, the global key (device serial number), and the private key mentioned earlier. This enables anyone who purchases or rents the device to easily verify its authenticity through the MSC. Once the manufactured devices are registered with the MSC, they are ready for sale (**Fig. 2**, step 1.1).

While the private key produced by the manufacturer is stored in protected memory to prevent tampering, it is important to consider the potential disclosure of confidential information in the hardware due to advancing technology. To mitigate such security vulnerabilities, we adopt a dynamic key formation strategy leveraging the flexible structure of the SDN-Enabled WSN described in Section 3.2 The SDN Controller (SDNC) within the centralized device in the WSN possesses all the control functions and has knowledge of all the nodes in the network. In the SDN-Enabled WSN structure, when the SDNC receives a connection request from the source node, it determines the optimal path between the source and destination by considering the energy consumption ratio and RSSI metrics. The SDNC stores this path in its flow table, which consists of a group of flow entries that specify the actions to be taken by each node on the path for incoming packets. In our model, each flow entry also includes a field to store the corresponding dynamic key, as illustrated in **Fig. 3**. Subsequently, the SDNC sends each flow entry to the respective nodes on the path. This procedure is repeated whenever a new route is determined.

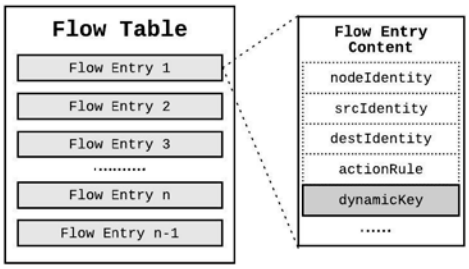

Fig. 3. Transmitting Dynamic Key to End Device via SDN Flow Table

In the model, SDNC requests key generation from MSC whenever a route change is required within the network. Route changes may occur due to attacks or unexpected failures within the network, which we consider as a critical situation requiring key change. When a route change occurs, SDNC receives a new random key from MSC and transmits it to the source node. The source node generates a new secret key (hs) by combining the incoming random key with the private key stored in its own memory. Using this secret key, the control code is created by hashing the data and other relevant information (1). The procedure for generating random key on the MSC is depicted in **Algorithm 1**. Symbols and their definitions are shown in **Table 1**.

Table 1. Definiton Table

Symbol	Definiton	Symbol	Definition
kh	keccak256 func	h_d	hash value of data
h_{dc}	hash value of data created date	h_{da}	hash value of data arrived date
h_{const}	hash value of constant value	h_s	hash value of secret key of node
c	control code	gk	global key of node
$WSNKey$	The parent object where keys such as hs, gk belonging to the relevant node are stored in the contract	$\oplus$	XOR operation
S	SLA information object of customer in the contract	S_{cc}	compensation data latency limit defined at current SLA

Algorithm 1. MSC – Key Generation Algorithm For Node

Input: gk
Output: $random\ key$
1: $Loop\ in\ WSNKeys\ as\ a\ WSNKey$
2: $If\ WSNKey_{gk}\ equals\ gk$
3: $random\ key = rand()$
4: $WSNKey_{hs} = kh(WSNKey_{hs} \oplus random\ key)$
5: $return\ random\ key$
6: $End\ If$
7: $End\ Loop$
8: $return\ empty\ value$

The rand() function is developed using the Solidity language and generates bytes32 values. This function is on the contract as a private view method and the return value is obtained through variable parameters of the Ethereum network such as block.timestamp, block.difficulty, msg.sender, now. Furthermore, all information exchange between SDNC and MSC takes place via SCH.

4.3 Service Provider Stage (2)

The service provider purchases the manufactured devices from the contracted manufacturer. The purchase process occurs exclusively through the previously established MSC by the manufacturer. The service provider specifies the desired brand and number of devices they want to purchase through the smart contract. The manufacturer approves the service provider's request (**Fig. 2**, step 1.3) and adds the private keys of the purchased devices by the service provider to a dedicated list on the MSC (**Fig. 2**, step 2.1). The main objective here is to ensure

the protection of the customer and service provider who will use the devices sold by the manufacturer.

In the proposed model, the SLA conditions requested by the customer are adapted and defined by the service provider (**Fig. 2**, step 2.2). When creating an SSC, the service provider must adhere to specific conditions imposed by the manufacturer. This means that each SSC needs to implement a smart contract interface specified by the manufacturer. In other words, the development of an SSC must be based on a template provided by the manufacturer. In the software realm, this approach aligns with object-oriented programming, where the template corresponds to an interface or an abstract class. This approach restricts the developer from deviating beyond certain structural requirements. In the proposed model, the manufacturer-imposed interface must satisfy the following conditions.

- It will have a method to verify data by connecting to MSC.
- Must take 6 parameters.
- Must have 1 byte return type.

The Solidity code created under these conditions is as **Fig. 4**.

```
pragma solidity ^0.5.1;
contract MSC_SSCI {
  function verifyControlCode(uint globalKey, bytes32 contentHash, bytes32 pckCreated, bytes32 pckArrived,
  bytes32 controlCode) public returns (bytes1);
}
contract SSC is MSC_SSCI {
   function verifyControlCode (uint globalKey, bytes32 contentHash, bytes32 pckCreated, bytes32 pckArrived,  bytes32
   controlCode) public returns (bytes1) {
      MSC msc = new MSC(mscAddress);
        ……...
   }
……..
}
```

Fig. 4. Solidity Code Interface For SSC

These conditions have been accepted as a standard specific to the model we propose. By accepting this standard, the smart contract is created and deployed together with the SLA terms. In addition, in our proposed model, the service provider must create and manage an intermediate service layer, which we call SCH, in order for the data coming over the Wireless Network to reach the smart contract. The service provider undertook the installation of the Wireless Network and the SCH infrastructure in the model we recommended.

4.4 Customer Stage (3)

The customer is the party which is requesting service; therefore, it usually contracts with the service provider that can best meet its needs. After the customer agrees with the service provider on the terms of the SLA, it approves the SSC created by the service provider (**Fig. 2**, step 3.1) and the deposit amount determined according to the terms of the SLA is transferred to the service provider's account. Thus, the service provider completes the setup.

4.5 Production Process Stage (4)

The details of the whole process are shown on the sequence diagram given in **Fig. 5**. In this stage, the data coming from the WSN is transferred to the SCH via TCP ports. Data is

transferred out of the wireless network over the sink End Device as stated earlier.

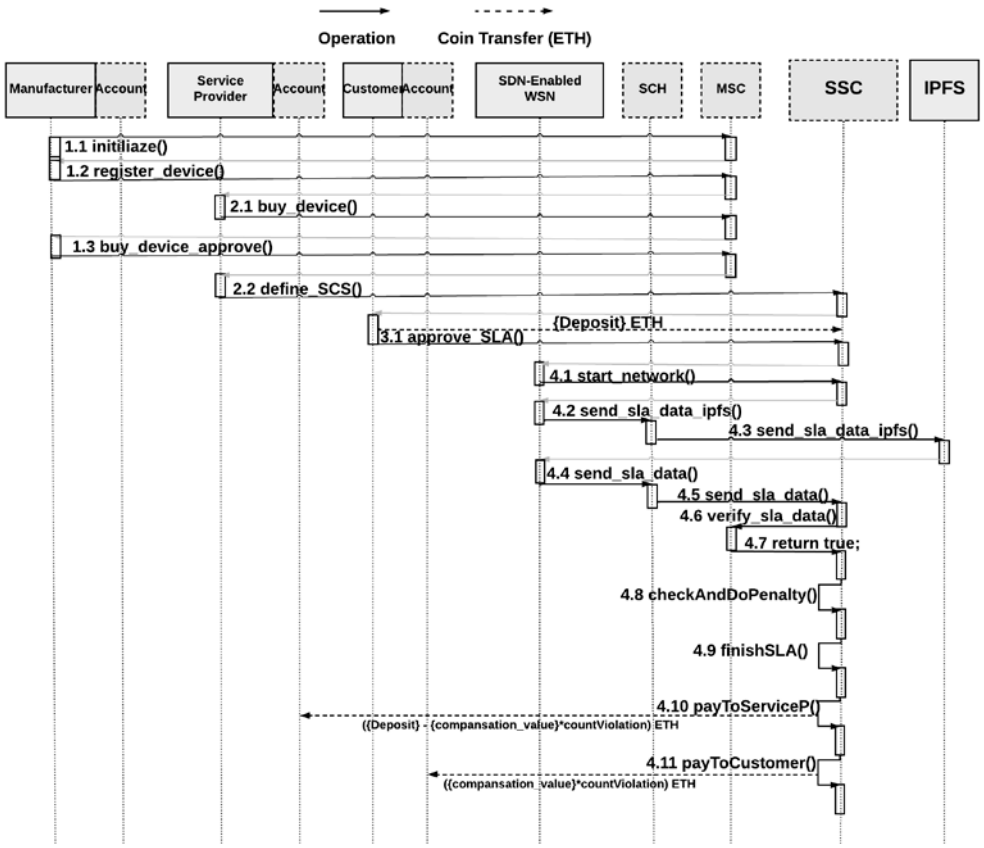

Fig. 5. Sequence Diagram of SDN-Enabled WSN SLA Ecosystem

In order to initiate data transfer over the Wireless Sensor Network, the SDNC must determine the optimal path between the source and destination. Once the data transmission commences (**Fig. 5**, step 4.1) and the data reaches the sink End Device, the sink appends certain parameters to the data to facilitate its delivery to the SCH. These parameters include the global key, network metrics, and control code specific to the End Device. Together with the data, these parameters form the frame depicted in **Fig. 6**. The network metrics can be adjusted to meet different requirements based on criteria such as downtime and energy consumption. In the case study presented in Section 5, we have selected the delay time as the criterion for mesh metrics. Thus, the parameters for this criterion consist of the data creation time and the arrival time at the End Device.

Fig. 6. SDN-Enabled WSN - SLA Data Frame

The control code is generated by applying specific processes to the data and network metrics using the private key stored in the secure memory of the End Device. In our proposed model, control code generation and verification occurs solely on the node and the MSC to ensure data transmission reliability. Furthermore, only the manufacturer is responsible for determining the

control code generation method and it remains undisclosed to others. We provide flexibility for manufacturers in this aspect. In our model, the parameters listed in **Table 1** are used for the generation of the control code. We perform an XOR operation on the parameters (h_x), followed by passing the obtained result through the keccak256 function to derive the control code (1).

$$kh(h_s \oplus h_d \oplus h_{dc} \oplus h_{da} \oplus h_{const}) = c \tag{1}$$

The SCH strictly records in IPFS the network measurements contained in every data it receives (**Fig. 5**, steps 4.2 and 4.3). SCH also transmits incoming data to SSC for violation control (**Fig. 5**, step 4.4 and 4.5). SSC sends the global key, network metrics and control code parameters in the incoming data to the MSC (**Fig. 5**, step 4.6), because it does not know the private keys of the devices. As we mentioned formerly the manufacturer registers all private keys to MSC during the production phase, with reference to their global keys. As a result, SSC definitely performs data verification over MSC. The MSC control code verification mechanism is shown in **Algorithm 2**. The control code verification function of the MSC is designated as a view function in Solidity, ensuring that it operates without gas consumption. Additionally, gas-free functions like keccak256 and abi.encodePacked are employed. In essence, no gas expenditure is required for these operations. In the model we recommend in this respect, the service provider or customer does not have to pay any fees for this transaction.

Algorithm 2. MSC Algorithm – Control Code Verification

Input: $gk, h_d, h_{dc}, h_{da}, c$
Output: $result\ of\ verification$
1: $Loop\ in\ WSNKeys\ as\ a\ WSNKey$
2: $\quad If\ WSNKey_{gk}\ equals\ gk$
3: $\quad\quad If\ kh(WSNKey_{hs} \oplus\ h_d \oplus\ h_{da} \oplus\ h_{dc} \oplus WSNKey_{hconst})\ equals\ c$
4: $\quad\quad\quad return\ valid$
5: $\quad\quad End\ If$
6: $\quad End\ If$
7: $End\ Loop$
8: $return\ invalid$

In the control code verification mechanism, the global key parameter is first searched in a Map located within the MSC. If a match is found, the private key associated with the corresponding list item is retrieved. Subsequently, this specific key and the incoming parameters undergo the keccak256 hash process, and the resulting code is compared with the control code. If the comparison yields a match, the SSC returns true; otherwise, it returns false (**Fig. 5**, step 4.7). The SSC also increments the verification or error counter based on the response. As a result, even if any malicious behavior occurs during the IPFS recording phase, the total number of data confirmations via the SSC can be compared with the IPFS file system during monthly reporting. Any discrepancies can lead to the cancellation of the SLA.

Algorithm 3. SSC Algorithm – SLA Check Data

Input: $slaId, gk, h_d, h_{dc}, h_{da}, c$
Output:
1: *Loop in SLA list as a S*
2: *If* S_{id} *equals slaId and c verification is successful in Algorithm* 2
3: *If* $S_{cc} < (h_{da} - h_{dc})$
4: *invoke calculatePenalties func for service provider*
5: *End If*
6: *End If*
7: *End Loop*

The SSC's data control mechanism is depicted in **Algorithm 3**. The algorithm functions as follows: the MSC data validation function is invoked with the incoming parameters. If the return value is true, the algorithm proceeds. Utilizing the slaId parameter, the current SLA object is retrieved from the SLA Map list. A predefined acceptable delay time is obtained from the corresponding SLA object. This time is then compared with the difference between the creation time and arrival time of the incoming data. If the difference exceeds a certain threshold, the calculatePenalties function is invoked (**Fig. 5**, step 4.8). Within the calculatePenalties function, if the breach period for the current compensation exceeds the acceptable time, penalty transactions are executed, resulting in the transfer of penalty fees from the SSC balance to the client. Upon the expiration of the SLA lifetime (**Fig. 5**, step 4.9), the remaining amount in the SSC balance, after the penalty transactions, is transferred to the service provider's account (**Fig. 5**, step 4.10). Finally, the total amount of penalty transactions is sent to the customer's account (**Fig. 5**, step 4.11), thereby completing the process.

We conducted a comparison of the most relevant models recently proposed and the model we proposed over some metrics and summarized this in **Table 2**.

Table 2. Comparison between proposed and most related recent models

Related model	Infrastructure	SLA data (network metric) generation place for smart contract	Blockchain network location	Dynamic SLA	Blockchain Connector
Battula et al. [33]	Fog computing	Fog Device	Ethereum	Yes	Smart Oracle
Alzubaidi et al. [34]	Edge Computing	IoT Service Provider (server)	Hyperledger Fabric	No	Smart Oracle
Hang et al. [24]	IoT	Third-party app	Hyperledger Fabric	No	Hyperledger Composer
Uriarte et al. [9]	Edge computing	Third-party app	Ethereum	Yes	Smart Oracle
Kochovski et al. [35]	Edge computing	Fog/Edge Devices	Chain Link-Ethereum	No	Smart Oracle
Zhou et al. [10]	Cloud	Cloud Service	Ethereum	No	Smart Oracle
Proposed model	SDN-enabled WSN	WSN Node	Ethereum	No	SCH

In most of the similar studies, they managed IoT devices through Edge, Fog or third party applications. Therefore, even if there is no decision maker for SLA violation detection, SLA data (network metrics such as latency) sent to the smart contract are generated from these middleware devices or third party applications. The installation and management of these devices is done by the service provider. In addition, the applications running on these devices are usually under the control of the service provider. There may be an opportunity for a service provider looking to turn SLA rules in their favour. Therefore, the fact that a party to the agreement manages these devices, which are critical for SLA, may leave a question mark for the customer.

The main difference between the proposed model and other studies is that security has a reliable method from wireless node to smart contract, and only strictly network metric data is generated and secured in the wireless node. Therefore, a control code signature that does not accept changes is transmitted directly to the smart contract. SCH acts only as a bridge. The service provider cannot interfere with the data or the device generating the data. The security analysis for the proposed model is detailed in Section 5.5.

5. Case Study

5.1 Technical Overview

We identified and applied a case study to demonstrate the feasibility of the proposed model and to obtain debatable results. We have specified all the components and applications required for simulation in section 5.2. In the context of the case study infrastructure, we utilized the wireless sensor architecture previously developed in our work [1] as the fundamental framework. In the case study, we examined a company that does genetic research as a customer, and we made a sample service provider the counterparty to make an SLA agreement with the customer. The customer has sensitive sensor data and there are some SLA quality service conditions determined by the customer. In this context, we identified the customer's two requests and examined the potential of our proposed model to fulfill these requests.

- SLA rules work without exception and leave no room for doubt.
- Ensure that data (network metric) is transmitted from the wireless sensor node to the SCH and from there to the SSC without any manipulation along the transmission line.

In the model we proposed for the first request of the customer, we ensured that the SLA rules are immutable by storing them on the smart contract. In addition, with the SSC we have developed, the relevant fines are transferred to the customer's account without exception at the time of working. Through the control code verification procedure we have devised, network metric values are safeguarded against any alteration. Thus, any external factor cannot violate the SLA rules by tampering with the network metric values along the transmission line and cause a monetary manipulation. In section 5.3, we concretely presented the SLA metrics and explained how we did the calculations. We showed the model on an example figure. In section 5.3, we discussed the applicability of the model with the experimental results. We have graphed the critical energy efficiency and time delay results for low-power wireless nodes. For the second request of the customer, we made a security analysis of the model in section 5.5, examined the possible threats on a scenario basis and expressed how we overcome these threats.

5.2 Platform Components

5.2.1 WSN Simulation Model

SDN-Enabled WSN structure in the platform is modeled and simulated using Riverbed Modeler Software. The WSN has 10 EDs and 1 SDNC device in the simulation scenario, as shown in **Fig. 7**.

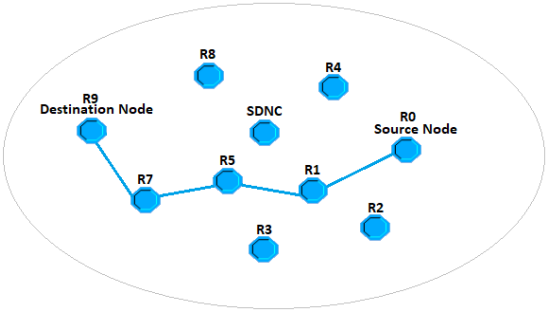

Fig. 7. Simulation model of SDN-Enabled WSN

MICAz node energy consumption parameters are used in the simulation for a more realistic performance evaluation. In addition, similar working conditions are chosen for all scenarios to obtain a consistent performance comparison. Relevant simulation parameters are tabulated in **Table 3**.

Table 3. Relevant simulation parameters

	Name	Value
Device settings for both the proposal and ZigBee-based WSAN	Data Rate	250 kbps
	ED status transmission period	25 s
	Initial energy	5 J
	Channel model	Free-space propagation model (LoS)
	Power threshold	−76 dBm (80 mW)
Battery parameters (Micaz mode) for both the proposal and ZigBee-based WSAN	Transmission mode (0 dBm)	17.4 mA
	Receive mode	27.7 mA
	Idle mode	35 µA
	Sleep mode	16µA

5.2.2 Smart Contract Tools

- The technologies we utilize to develop smart contract in the proposed model are listed below.
- Solidity (Ethereum VM Language): Solidity is an object oriented programming language used to develop smart contracts [36]. It is used on many blockchain platforms, especially Ethereum. Thus, the behavior of the accounts can be changed [37]. The developed code runs on EVM after it is compiled.
- Truffle (smart contract Test and Deploy Framework): Truffle is a developer environment that enables simple and easy testing of Ethereum smart contracts after they develop them [38].

- Ganache (smart contract Local VM): Ganache creates a virtual Ethereum blockchain for the smart contracts developed [39]. It also testing the code that is developed by opening new accounts on this virtual blockchain.
- Web3js (smart contract Java Class Helper): web3.js is a program library that allows you to communicate with a local or remote Ethereum network using HTTP, IPC or WebSocket [40].
- MetaMask (Blockchain Account Panel - Google Chrome Plugin): MetaMask is a cryptocurrency wallet developed as a browser extension to communicate with the Ethereum network [41]. Thanks to a compatible browser, users can manage their own accounts, and send or receive ETH with this add-on.

5.2.3 Smart Contract Helper (SCH)

smart contract Helper is a platform based on Spring Boot [42] that serves as a mediator between smart contracts and the SDN-Enabled WSN. It facilitates the communication between the SDN-Enabled WSN and the smart contracts deployed on the blockchain network. The responsibility for managing this platform lies with the service provider, who is tasked with ensuring its continuous operation. The helper leverages the Web3.js Java library to establish communication with the smart contracts. Additionally, several RESTful POST services have been developed to handle various tasks within the platform.

5.2.4 IPFS (The InterPlanetary File System)

IPFS can be defined as a distributed file system that generally enables data storage and sharing on a distributed peer- to-peer network [43]. In the IPFS system, a unique key code is given to each file and all of its blocks. The nodes in the network store the content being interested and some indexing information related to other nodes' content. Thus, a secure web environment against deletion is provided.

5.3 Experimental Model and Assumptions

Table 4 includes the operating parameters and assumptions of the experimental study employed to obtain the performance of the proposed SLA management platform. **Fig. 8** illustrates the SDN-Enabled WSN topology that the service provider establishes for XY company.

Table 4. Case Study for Proposed Model

Customer	XY Company Conducting Genetic Research
Network Infrastructure	SDN-Enabled WSN (Fig.7)
Sensor Measurement Period	2 sec
Compansation Period	300 sec (150*2)
Acceptable Average Transmission Time	0.194 sec
Targeted Transmission Delay Rate	< %5
Customer Deposit	30 ETH
Penalty Per Compensation Period	0.1 ETH
Simulation Time	30 min

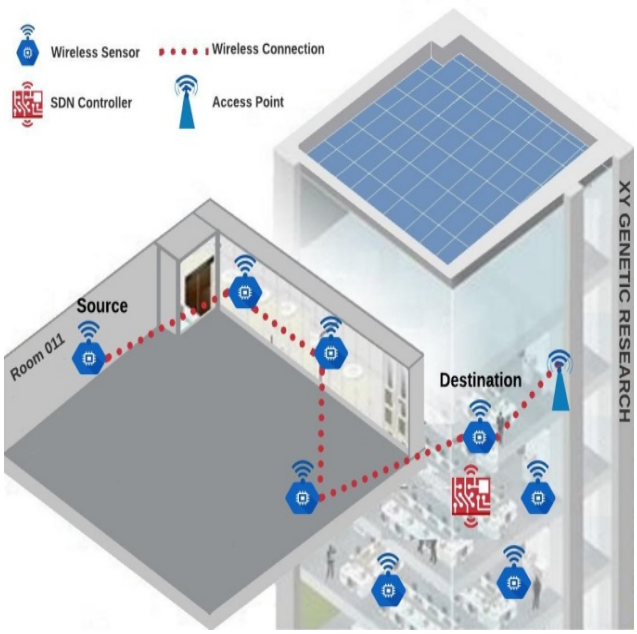

Fig. 8. SDN-Enabled WSN topology of the smart building in the case study

In the presented case study, XY is a customer company engaged in genetic research. Temperature measurement plays a crucial role in their research, particularly in room number 011. It is imperative to measure the temperature accurately and ensure uninterrupted data transfer to the central data servers. The primary criteria for this scenario are data continuity, immutability and speed. To meet these requirements, XY establishes an SSC based on blockchain assurance with a network infrastructure provider (Service Provider). The SSC outlines the terms and conditions for the data transmission process, specifically stating that sensor data must be transmitted to the central server before it drops below a predefined threshold. The client has the flexibility to define individual requirements for each research room. In the evaluation of performance, room number 011 is considered, taking into account the following conditions and assumptions;

- The temperature values measured in Room 011 must be transmitted to the central server every 2 seconds.
- If the customer accepts the contract, they are required to deposit a fee of up to 30 ETH.
- The total packet delay is calculated for every 150 sensor data packets.
- The maximum acceptable time for each data packet to reach the server is determined as 0.253 seconds. To determine this value, the average data transmission times on and off the network were measured in a pre-scenario test simulation. In this simulation, a total of 900 data packets were transmitted to the destination node. **Fig. 9** illustrates the time-dependent graph showing the duration of packet transmission from the origin node to the destination. The average transmission time was calculated as 0.194 seconds, and a 30% deviation time was added to the average time to determine the acceptable time.

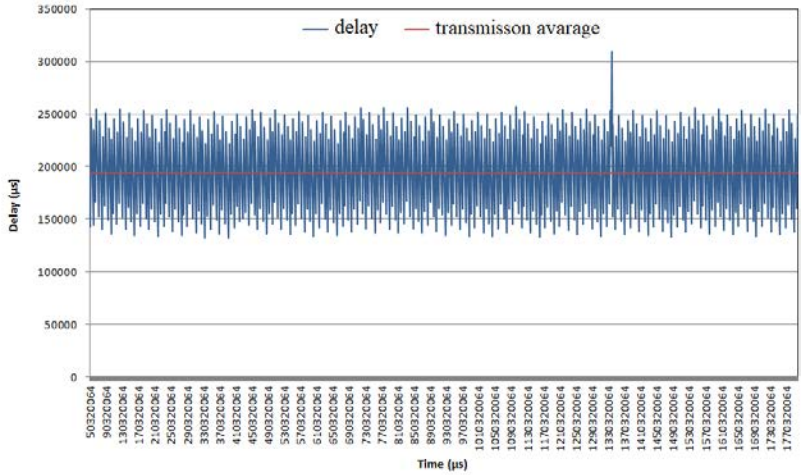

Fig. 9. Transmission Time

The latency limit was set by the customer as 5%. In order to calculate actual delay rate, the total delay time is calculated (2). Then, the delay rate is calculated by using this obtained delay time (3). Calculation details are shown on below.

$$\{TotalDelayForCompansation\} = \{TotalDataTransferredTime\} - (\{CompansationPeriod\} + \{AcceptableAverageTransmissionTime\} * 150)$$

$$\{TotalDelayForCompansation\} = \{TotalDataTransferredTime\} - (300 + 0{,}253 * 150) \quad (2)$$

$$\{TargetedTransmissionDelayRate\} = \left(\frac{\{TotalDelayForCompansation\}}{\{TotalDataTransferredTime\}}\right) * 100 \quad (3)$$

Total transmission success will be calculated; if $\{TargetedTransmissionDelayRate\}$ greater than five, penalty action $\{PenaltyPerCompensationPeriod\}$ will be applied to the service provider. Since data delay is the main criterion in this scenario, the network metrics in **Fig. 6** were determined as the in-network creation date of the data and the in-network arrival date of the data.

5.4 Experimental Results and Discussions

The simulation results of the WSN model in the proposed SLA management platform were obtained using a computer equipped with a 2.50 GHz i5 2450M processor and 4 GB of RAM. The simulation ran for 30 minutes, during which 900 temperature data points were transmitted from the source to the destination node as shown in **Fig. 9**. Upon receiving the data, the destination node adds certain parameters, including the global key, data creation date, data arrival date, and control code, before forwarding it to the smart contract Helper. The smart contract Helper then sends the incoming data to both IPFS and SSC. During each compensation period, which occurs every 5 minutes, the SSC calculates the delay. If the delay exceeds the predetermined value, penalty action is applied. The SSC verifies the correctness of the control code by transmitting it to the MSC. It is important to note that no faulty control codes were generated during the simulation. The initial balances in the customer, service provider, and SSC accounts are summarized in **Table 5**.

Table 5. Balance Initial Values of Customer, Service Provider and Contract Accounts

Account Name	Initial Value
Service Provider Account	5 ETH
Customer Account	40 ETH
SSC Account	0,5 ETH
Gas Unit Fee = 20 Gwei = $(\mathbf{20} * \mathbf{10^{-9}}\ \boldsymbol{ETH})$	

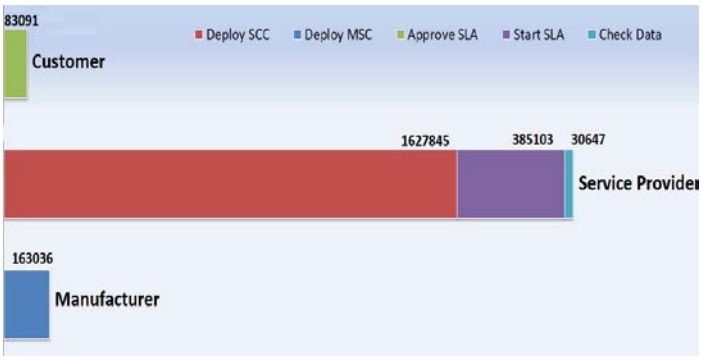

Fig. 10. Amounts of Gas Consumed for smart contract Processes

The initial balance of the service provider is set as 5 ETH, which covers the transaction fee required to initiate the SLA on the smart contract. Similarly, the initial balance of the SSC is determined as 0.5 ETH to handle penalty transactions. To provide more detailed simulation results, high values are chosen for deposit and penalty fees. **Fig. 10** illustrates the gas consumption of smart contract transactions received through Truffle. It is evident from the figure that the service provider consumes the highest amount of gas compared to other parties. This discrepancy arises from the fact that the service provider deploys the SSC. The gas consumption for SSC deployment is determined by variables defined in the contract for function content and storage of critical data. For instance, storing 32 bytes of data requires 20,000 gas consumption [44]. Therefore, the deployment cost of the SSC is relatively high. The total gas consumption for SSC and MSC deployment is calculated as 1,790,881 (0.0358 ETH). While the total gas consumption to approve the defined SLA in the customer SSC is 83,091, the gas amount spent by the service provider to initiate this SLA is 385,103. Additionally, during the Check Data operation in the SSC, 33,974 gas is consumed to record violations, and another 33,974 gas is consumed to pay penalties to the customer. Ethereum imposes gas consumption for money transfers, where higher gas amounts lead to faster transfer processes on the network. In our simulation, the gas consumption for a money transfer is determined as 8,281. **Fig. 11** illustrates the time-dependent graph of customer, service provider and SSC smart contract account balances during the SLA execution process. Upon SLA approval, the customer transfers 30 ETH to the SSC accounts. The total transmission time is monitored during six compensation intervals. The transmission times during compensation intervals 2 and 5 exceed the targeted time. As a result, in both compensation intervals, 0.15 ETH is transferred from the SSC account to the customer's account. When the term is completed, the SLA is terminated, and the SSC transfers the remaining balance to the service provider. As depicted in the graph, the proposed model ensures that penalties for SLA violations at time t are promptly transferred to the customer's account. Upon the SLA's expiration, any remaining balance in the contract account is automatically transferred to the service provider's account.

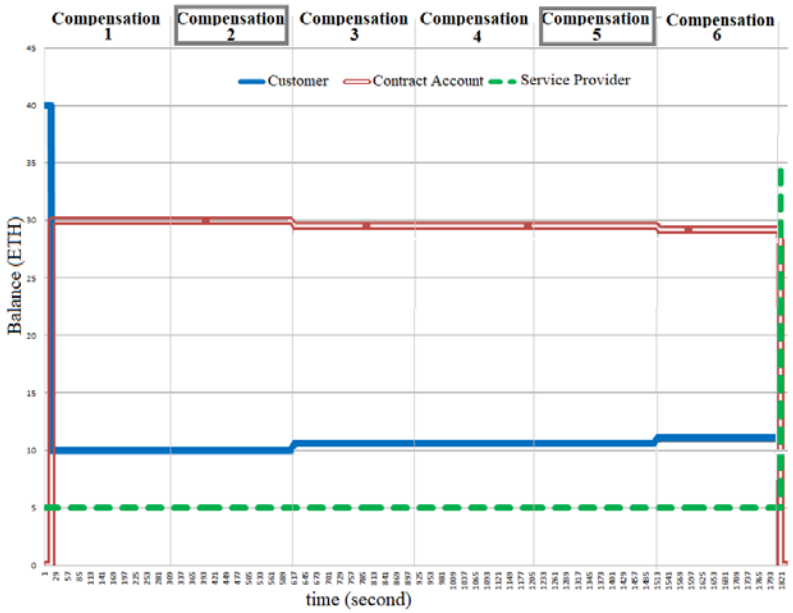

Fig. 11. Time Based Change of Balance Values of Customer, Service Provider and Contract Accounts in Data Transferring and Verifying Period

For the SDN-Enabled WSN in the case study, two different simulation scenarios are evaluated to investigate the effects of the developed models and algorithms. The first one, which is referred to as "Scenario 1", includes the sensor network without any SLA consideration. In contrast, the second one, which is referred to as "Scenario 2", has the proposed SLA-related modifications on the application layer of ED, which integrates the sensor network into the SLA management platform.

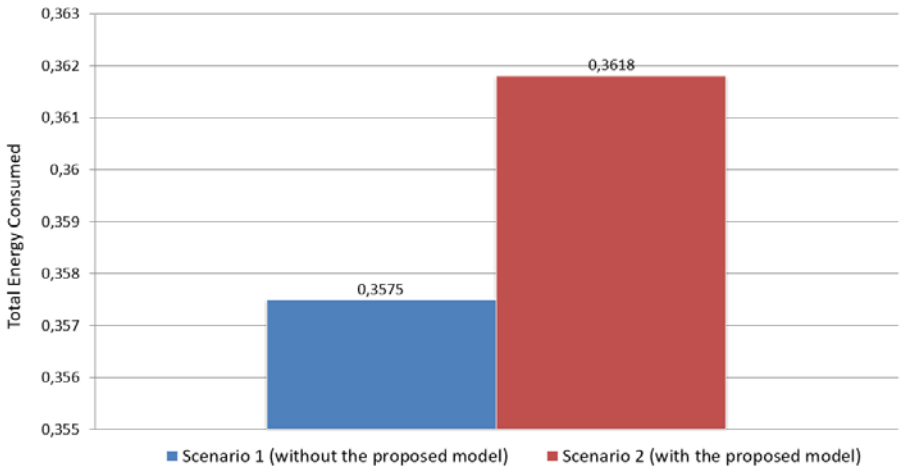

Fig. 12. Total Energy Consumed for Proposed Model Enabled/Disabled

Fig. 12 and **Fig. 13** present the results of both scenarios. **Fig. 14** displays the total energy consumption ratio of the SDN-Enabled WSN. It can be observed from the figure that the End Device (ED) in Scenario 2 consumes a total of 0.0043 Joules more energy compared to

Scenario 1. This difference is expected since the proposed SLA-related algorithms impose computational load on the system. Furthermore, when considering both results, there is no significant energy consumption observed on the last nodes. This is because the SDN controller (SDNC) performs the resource-intensive operations that require higher performance in the SDN-Enabled WSN architecture.

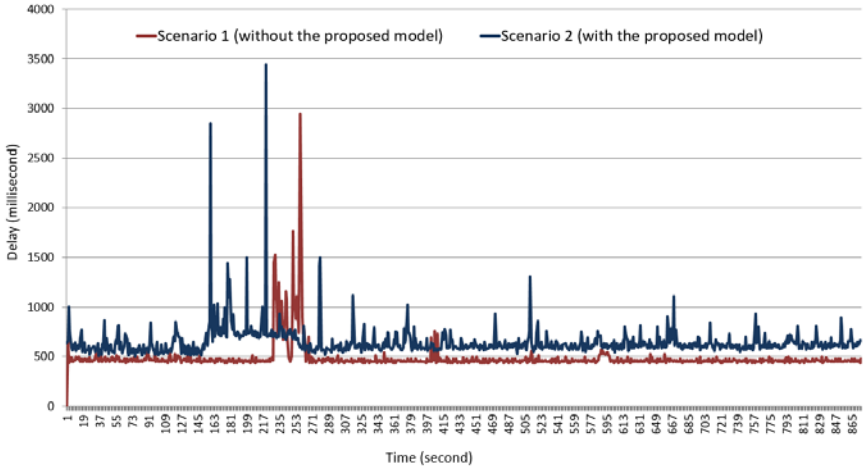

Fig. 13. Data Transmission Delay Between End Device and SCH for Proposed Model Enabled/Disabled

As a result, the EDs can conserve their energy, which is crucial for prolonging the network's lifetime [1] [26]. Another advantage of the SDN-Enabled WSN architecture for the proposed model is the quick transfer of statistics required for the SLA defined on the smart contract from the SDNC to the nodes, and subsequently transmitted to the smart contract Helper (SCH) through the last node. This enables fast and energy-efficient interaction with smart contracts. **Fig. 13** illustrates the time taken to transfer SLA data from the End Device to the SCH, considering both scenarios. The results indicate that Scenario 2, with the proposed SLA extension, exhibits a greater delay compared to Scenario 1 due to the additional processing cost required for generating the SLA-related control code in the End Device. Furthermore, in terms of the sustainability of the proposed model, smart contracts' processing speeds should be stable and fast. Therefore, the verification of the control code, which constitutes the main workload in the smart contract, should be swift.

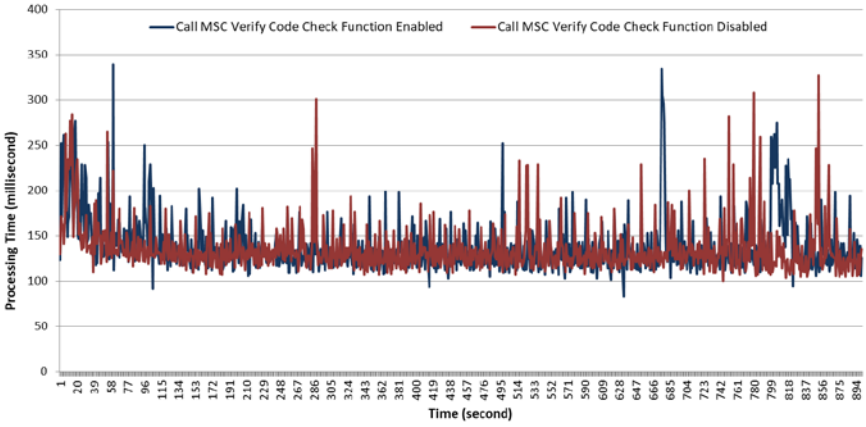

Fig. 14. Processing Time of Algorithm-3 According to Enabled-Disabled State of Line 4 in Algorithm-3

Fig. 14 demonstrates the processing time of the code (**Algorithm 3**) that includes the SSC data control mechanism. It was previously mentioned that the SSC calls the MSC function (line 2 of the code in **Algorithm 3**) to confirm the correctness of the received data. To assess the additional cost incurred by calling this action and determine if any unexpected delay would disrupt the system, we disabled this process and restarted the simulation. The results indicate that there was no significant change observed in the average time. This process took a negligible time of approximately 3,006 milliseconds. As a result, the data verification mechanism in the SSC introduces an extra delay of 0.75% during data transmission from the End Device to the SCH and increases the energy consumption ratio by 1.2% for the target node. Considering that these values are quite small, the additional cost of our security model can be neglected for many applications.

5.5 Security Analysis of Proposed Model

We have established the trust between the model intermediaries we recommend. However, to ensure this trust, it is possible to ensure that the SLA rules and network metrics are unchangeable on the blockchain and that the data is not manipulated throughout the transmission channel. We have demonstrated the feasibility of the proposed model through a comprehensive case study in previous chapters, complemented by a thorough security analysis. We tested the resilience of the model against threats over possible threat scenarios. Based on the case study shown in **Fig. 8**, we can examine the following threat scenarios.

5.5.1 Threat Scenario 1: Unauthorized Node Tampering and Data Interference

We assume that an unauthorized and malicious attacker in the building tampered with the nodes or tampered with the network, preventing data from reaching the End Device. In our previous study, we tested attack scenarios such as Wormhole attack, Sinkhole Attack and Black Hole Attack that cause data to move in different rotations or cause data to be destroyed, on the same infrastructure. We have used the model we developed in our previous study [1] in this working architecture as well. For such attacks, it has been simulated over OPNET on a scenario basis and it has been observed that the attacks made for the data transmitted from node to node in the WSN network are successfully prevented.

5.5.2 Threat Scenario 2: Impersonate The SCH (Copycat Attack)

The attacker could impersonate the SCH acting as the orchestrator between the WSN and the smart contract. For example, the attacker may want to transmit the temperature measurement values to the smart contract with different values through the copycat application.

- In the model we proposed, the service provider is responsible for creating a specific smart contract instance for each customer. Therefore, the attacker must first know the application-specific smart contract account address. In case of incompatibility between the attacker account address and SCH, he will not be able to forward the data as he wishes.
- Each contract has its own device and private keys and dynamic keys for each end-to-end data transmission at runtime. Therefore, if the application account address is captured, the copycat SCH can transmit its own generated data to the contract. However, the proposed model introduces **Algorithm 2** as a safeguard. If the data does not match the control code signature value, it receives a rejection from the contract. With a certain number of rejections, the service provider will be able to understand that there is an attack on the network. The service provider can determine the relevant metrics for network intrusion detection.

5.5.3 Threat Scenario 3: Abuse of Responsibility

In our proposed model, we stated that WSN architecture management and installation is a division of labor belonging to the provider. A vulnerability can occur if this responsibility is abused or installed in a way that is vulnerable to external threats. Especially in WSN, the change of the location of the nodes is an important criterion in terms of security. It is not expected that the position will change during operation. In our previous work, we have stored the route changes and node active/passive record keeping on the SDN Controller due to location change or other reasons. In addition, in this study, we ensured that the route changes are verifiable, and we made it mandatory to request a random key in the contract for situations that would require a route change. Network metrics and control packets are stored as read-only in the IPFS environment. Thus, the customer can examine such trust parameters or verify this data via MSC at the end of the day.

The security analysis performed in potential threat scenarios highlights the robustness of the proposed model in mitigating potential risks and protecting data and network integrity. Thanks to the integration of lightweight cryptographic methodology, the model stands as a solid solution for security challenges in Wireless Sensor Networks.

6. Conclusion

Blockchain technology is experiencing daily growth in its widespread use, and it is interacting with various technologies to create new architectures in numerous domains. With the flexible nature of wireless networks and the powerful capabilities of smart contracts, our proposed model offers extensive possibilities for future use in various fields. Although current EVM gas fees may not provide a comfortable experience for IoT, we believe that this issue will be resolved in the future with the emergence of new technologies that offer low transaction fees, presenting a novel approach to this problem. In our proposed model, we have ensured that gas usage is minimized in recurring operations. We have demonstrated that the data control code verification function, which is the cornerstone of our model, is cost-free, and the additional operations performed on the nodes do not impose a significant burden in terms of energy and performance. This makes the model promising for real-life applications.

Furthermore, we assert that the dynamic key generation mechanism in our proposed model effectively mitigates attack vectors such as imitation of the control code generation process, tampering with the node by malicious agents, or unauthorized installation of new programs. As IoT systems continue to advance rapidly, the usage of wireless networks expands accordingly. Therefore, the case study of our proposed model can be applied in various domains beyond network management operations. Some potential case studies include:

- Resolving disputes between parties in cases of customer-side damages (e.g., fire, system outages) resulting from weaknesses in the network.
- Network leasing and transfer.
- Incorporating additional sensors into the network (allowing users to contribute their own data using their own sensors by joining the wireless network through smart contracts with specific protocols).
- Determining network usage times based on SLA agreements.

References

[1] E.Karakoc, C. Ceken, "Black Hole Attack Prevent Scheme using Blockchain-Block Approach in SDN-Enabled WSN," *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 37, no.1, pp. 37-49, 2021. Article (CrossRef Link)

[2] D. C. Verma, "Service level agreements on ip networks," *Proceedings of the IEEE*, vol. 92, no. 9, pp. 1382–1388, 2004. Article (CrossRef Link)

[3] João Paulo de Brito Gonçalves, Roberta Lima Gomes, Rodolfo da Silva Villaca, Esteban Municio, Johann Marquez-Barja, "A Quality of Service Compliance System Empowered by Smart Contracts and Oracles," in *Proc. of 2020 IEEE International Conference on Blockchain (Blockchain)*, 2020. Article (CrossRef Link)

[4] O. F. Rana, M. Warnier, T. B. Quillinan, F. Brazier, and D. Cojocarasu, "Managing Violations in Service Level Agreements," in *Grid Middleware and Services*, Boston, MA: Springer US, 2008, pp. 349–358. Article (CrossRef Link)

[5] G. Gaillard, D. Barthel, F. Theoleyre and F. Valois, "Service Level Agreements for WSN: a WSN Operator's Point of View," in *Proc. of 2014 IEEE Network Operations and Management Symposium (NOMS)*, 2014. Article (CrossRef Link)

[6] João Paulo de Brito Gonçalves, Rodolfo da Silva Villaca, Esteban Municio, Johann Marquez-Barja, "A Service Level Agreement Verification System using Blockchains," in *Proc. of 2020 IEEE 11th International Conference on Software Engineering and Service Science (ICSESS)*, 2020. Article (CrossRef Link)

[7] E. J. Scheid, B. B. Rodrigues, L. Z. Granville, and B. Stiller, "Enabling dynamic sla compensation using Blockchain-based Smart Contracts," in *Proc. of 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, IEEE, pp. 53–61, 2019. Article (CrossRef Link)

[8] R. B. Uriarte, H. Zhou, K. Kritikos, Z. Shi, Z. Zhao, R. D. Nicola, "Distributed service-level agreement management with Smart Contracts and Blockchain," *Concurrency and Computation Practice and Experience*, vol. 33, no. 14, 2021. Article (CrossRef Link)

[9] R. B. Uriarte, R. de Nicola, and K. Kritikos, "Towards distributed sla management with Smart Contracts and Blockchain," in *Proc. of 2018 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pp. 266–271, 2018. Article (CrossRef Link)

[10] H. Zhou, X. Ouyang, Z. Ren, J. Su, C. de Laat, and Z. Zhao, "Enforcing trustworthy cloud SLA with witnesses: A game theory–based model using Smart Contracts" *Concurrency and Computation Practice and Experience*, vol. 33, no. 14, 2021. Article (CrossRef Link)

[11] Y. C. Hu, T. T. Lee, D. Chatzopoulos, P. Hui, "Analyzing Smart Contract interactions and contract level state consensus," *Special Issue: Special Issue on Cryptocurrencies and Blockchains for Distributed Systems*, Vol. 32, no. 12, 18 March 2019. Article (CrossRef Link)

[12] E. J. Scheid, B. Stiller, "Automatic SLA Compensation based on Smart Contracts," technical report – No. IFI-2018.02,University of Zurich Department of Informatics. Article (CrossRef Link)

[13] J. Backman, S. Yrjol¨ a, K. Valtanen, and O. M ammel a, "Blockchain network slice broker in 5g: Slice leasing in factory of the future use case," in *Proc. of 2017 Internet of Things Business Models, Users, and Networks*, IEEE, pp. 1–8, 2017. Article (CrossRef Link)

[14] L. Zanzi, A. Albanese, V. Sciancalepore, and X. Costa-Perez, "Ns-bchain: A secure Blockchain framework for network slicing brokerage," in *Proc. of ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020. Article (CrossRef Link)

[15] S. Zheng, T. Han , Y. Jiang, X. Ge, "Smart Contract-Based Spectrum Sharing Transactions for Multi-Operators Wireless Communication Networks," *IEEE Access*, Vol. 8, pp. 88547 – 88557, 2020. Article (CrossRef Link)

[16] A. S. Omar, O. Basir, "Identity Management in IoT Networks Using Blockchain and Smart Contracts," in *Proc. of 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, 2018. Article (CrossRef Link)

[17] M. Shurman, A.Al-Rahman Obeidat, S. Al-Deen Al-Shurman, "Blockchain and Smart Contract for IoT," in *Proc. of 2020 11th International Conference on Information and Communication Systems (ICICS)*, 2020. Article (CrossRef Link)
[18] D. R. Putra, B.Anggorojati, A. P. P. Hartono, "Blockchain and smart-contract for scalable access control in Internet of Things," in *Proc. of 2019 International Conference on ICT for Smart Society (ICISS)*, 2019. Article (CrossRef Link)
[19] M. A. B. Ahmadon, S. Yamaguchi , "IoT Device Multi-layer Connection Management Mechanism with Blockchain Smart Contracts," in *Proc. of 2020 Zooming Innovation in Consumer Technologies Conference (ZINC)*, 2020. Article (CrossRef Link)
[20] H.-A. Pham, T.-K. Le, T.-N.-M. Pham, H.-Q.-T. Nguyen, T.-V. Le, "Enhanced Security of IoT Data Sharing Management by Smart Contracts and Blockchain," in *Proc. of 2019 19th International Symposium on Communications and Information Technologies (ISCIT)*, 2019. Article (CrossRef Link)
[21] Y. Zhang, S. Kasahara, Y. Shen, X. Jiang, J. Wan,"Smart Contract-Based Access Control for the Internet of Things," *IEEE Internet of Things Journal*, Vol. 6, No. 2, pp. 1594-1605, April 2019. Article (CrossRef Link)
[22] Y. N. Aung, T. Tantidham, "Ethereum-based Emergency Service for Smart Home System: Smart Contract Implementation," in *Proc. of 147 International Conference on Advanced Communications Technology(ICACT)*, 2019. Article (CrossRef Link)
[23] C. Lehnert, G. Engel, T. Greiner, "Distributed Ledger and Smart Contract Based Approach for IoT Sensor Applications," in *Proc. of 2020 International Conference on Omni-layer Intelligent Systems (COINS)*, Barcelona, Spain, 2020. Article (CrossRef Link)
[24] L. Hang and D. Kim, "SLA-Based Sharing Economy Service with Smart Contract for Resource Integrity in the Internet of Things," *Applied Sciences*, 9(17), 2019. Article (CrossRef Link)
[25] G.IORDACHE, Ad. PASCHKE, M. MOCANU, C. NEGRU1, "Service Level Agreement Characteristics of Monitoring WSN for Water Resource Management (SLAs4Water)," *Studies in Informatics and Control*, vol. 26, no. 4, pp. 379-386, 2017. Article (CrossRef Link)
[26] Mohammed Al Hubaishi, Celal Çeken, Ali Al Shaikhli, "A novel energy-aware routing mechanism for SDN-enabled WSAN," *International Journal of Communication Systems*, 2018. Article (CrossRef Link)
[27] Ali Al-Shaikhli, Celal Çeken, Mohammed Al-Hubaishi, "WSANFlow: An Interface Protocol between SDN Controller and End Devices for SDN-Oriented WSAN," *Wireless Personal Communications*, 101.2, 755-773, 2018. Article (CrossRef Link)
[28] Internet of Things Research Laboratory Sakarya University. [Online]. Available: http://www.iotlab.sakarya.edu.tr/Projects/Project1.html (Accessed February 22, 2023).
[29] [Online]. Available: https://en.bitcoin.it/wiki/Block_hashing_algorithm. (Accessed January 4, 2023).
[30] G. Wood et al., "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014. Article (CrossRef Link)
[31] Buterin, Vitalik (August 7, 2015). "Ethereum - On Public and Public Blockchains,". [Online]. Available: https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains.
[32] https://tr.wikipedia.org/wiki/Ethereum. (Accessed February 1, 2023)
[33] S. K. Battula, S. Garg, R. Naha, M. B. Amin, B. Kang, E. Aghasian, "A blockchain-based framework for automatic SLA management in fog computing environments," *The Journal of Supercomputing*, vol. 78, pp. 16647-16677, 2022. Article (CrossRef Link)
[34] A. Alzubaidi, , K. Mitra and E. Solaiman "A blockchain-based SLA monitoring and compliance assessment for IoT ecosystems," *Journal of Cloud Computing: Advances, Systems and Applications*, vol. 78, 2023. Article (CrossRef Link)
[35] P. Kochovski, V. Stankovski, S. Gec, F. Faticanti, M.Savi, D. Siracusa, S. Kum, "Smart Contracts for Service-Level Agreements in Edge-to-Cloud Computing," *Journal of Grid Computing*, vol. 18, pp. 673-690, 2020. Article (CrossRef Link)
[36] https://en.wikipedia.org/wiki/Solidity (Accessed February 3, 2023)
[37] https://docs.soliditylang.org/en/v0.8.1/ (Accessed February 3, 2023)

[38] https://www.trufflesuite.com/docs/truffle/overview (Accessed February 3, 2023)
[39] https://www.trufflesuite.com/docs/ganache/overview (Accessed February 3, 2023)
[40] https://web3js.readthedocs.io/en/v1.3.0/ (Accessed February 3, 2023)
[41] https://en.wikipedia.org/wiki/MetaMask (Accessed February 3, 2023)
[42] https://spring.io/projects/spring-boot (Accessed February 3, 2023)
[43] https://ipfs.io/ (Accessed February 3, 2023)
[44] http://gavwood.com/paper.pdf (Accessed February 3, 2023)

Emre Karakoç graduated from Sakarya University Computer Engineering Department in 2012. At the same time, he completed her master's degree on Computer and Informatics Engineering at Sakarya University in 2016. He has been actively developing software since 2012 and took part in various IT sectors. His interests are IOT, Blockchain, Wireless Sensor Networks, Software Defined Networking and Blockchain technology. He completed his doctorate in 2021.

Celal Çeken received his MSc and PhD in Computer Science from University of Kocaeli, Turkey in 2001 and 2004, respectively. His current research interests include Wireless Sensor Networks, Internet of Things (IoT), IoT Data Analytics, Software-Defined Networking and Secure Software Development.