An Adaptive Security Governance Architecture based on Smart Contracts for Syntactically Interoperable Services in Smart Cities

Shahbaz Siddiqui [1], Sufian Hameed [1], Syed Attique Shah [1], and Dirk Draheim [1]

[1]Affiliation not available

October 31, 2023

Abstract

An Adaptive Security Governance Architecture based on Smart Contracts for Syntactically Interoperable Services in Smart Cities

Posted on 22 Jun 2020 — CC-BY 4.0 — https://doi.org/10.36227/techrxiv.23528370.v1 — e-Prints posted on TechRxiv are preliminary reports that are not peer reviewed. They should not b...

An Adaptive Security Governance Architecture based on Smart Contracts for Syntactically Interoperable Services in Smart Cities

Shahbaz Siddiqui, Sufian Hameed, Syed Attique Shah, *Senior Member, IEEE*, Dirk Draheim, *Member, IEEE*

***Abstract*—Smart cities have emerged as a promising paradigm for improving the quality of life for citizens through advanced infrastructure and technological innovations. Collaborative services play a pivotal role in fostering seamless cooperation and interaction among diverse entities within smart cities, including government agencies, businesses, and individuals, leading to improved community outcomes. However, the management of syntactically interoperable services for collaborative tasks poses administrative challenges, particularly in terms of data security and privacy when sharing sensitive information across heterogeneous IoT devices. Furthermore, authentication, authorization, and trust management across several different smart city services emerge as important research topics. This research proposes an innovative adaptive security governance framework designed specifically for smart cities to address these challenges. The framework leverages dynamic security policies implemented through smart contracts to ensure data security and privacy during the interoperation of smart services. Real-world collaborative smart city use cases are presented to validate the effectiveness of the framework, integrating multi-chain blockchain technology, smart services APIs, and Software-Defined Networking (SDN). The implementation demonstrates the framework's ability to enhance the security and efficiency of collaborative services in smart cities. This research contributes to advancing secure and efficient collaborative services in smart cities, with a focus on mitigating administrative challenges and prioritizing data security and privacy. By leveraging the proposed framework, smart cities can elevate the standard of living for their residents while addressing critical security concerns in a dynamic and evolving environment.**

***Index Terms*—Smart city, blockchain, Software-Defined Networking, Internet of Things, multi-chain blockchains, Multi-Chain, collaborative services, interoperability, security adaptation**

I. Introduction

SMART cities are rapidly emerging as a transformative model for urban living, capitalizing on cutting-edge technologies to improve the well-being of residents. At the core of these cities, smart services and applications are being developed with interoperability in mind, enabling seamless communication and collaboration among diverse systems and devices [1]. Deploying interoperability in smart cities has the potential to revolutionize various sectors and their services, including transportation, energy, healthcare, waste management, and public safety, by facilitating synchronized and efficient operations. For instance, the implementation of smart transportation systems allows for optimized traffic flow through the analysis of data from multiple sources, including real-time information from sensors, traffic cameras, and public transportation schedules [2]. Leveraging this data, traffic signal timings can be dynamically adjusted, vehicles can be redirected, and personalized travel advice can be provided to commuters, leading to a reduction in congestion and an improvement in overall mobility [3], [4]. Figure 1 illustrates a representative depiction of a smart city consisting of interoperable services.

Despite the numerous advantages offered by the interoperability of smart services, there are significant concerns that need to be addressed, with data security as the most crucial aspect. In a networked environment where systems and devices exchange data for collaborative tasks, ensuring the security, integrity, and accessibility of data becomes of paramount importance. The potential risks associated with data breaches or cyberattacks are a major challenge that can jeopardize private information, disrupt services, and compromise the security and privacy of residents [5]. Malicious entities, such as hackers, can exploit vulnerabilities within interconnected systems to gain unauthorized access or manipulate data, leading to severe consequences. Therefore, it is imperative to establish robust security measures to protect against such threats and safeguard critical information and infrastructure within smart cities.

In addition to data security, another obstacle to achieving effective interoperability is the need for data governance and standardization. With multiple systems and devices exchanging data, standardized data formats, protocols, and security measures are essential to ensure seamless communication and mitigate the complexities associated with data integration [6], [7]. The absence of standardization can result in challenges related to data integration, data quality, and potential security vulnerabilities [8]. To address these challenges, collaborative efforts among policymakers, researchers, and industry stakeholders are required. Comprehensive frameworks and guidelines need to be developed, encompassing robust security measures, standardized protocols, and data governance frameworks, to ensure secure and efficient data exchange between systems and devices within smart cities.

Blockchain technology has emerged as a promising solution for addressing the data security challenges associated with interoperable smart services in smart cities [9], [10]. By

Shahbaz Siddiqui and Sufian Hameed, are with the Department of Computer Science, National University of Computer and Emerging Sciences (NUCES-FAST), Karachi, Pakistan. (e-mail: shahbaz.siddiqui@nu.edu.pk, sufian.hameed@nu.edu.pk)

Corresponding author: Syed Attique Shah is with the School of Computing and Digital Technology, Birmingham City University, STEAMhouse, Belmont Row, B4 7RQ, Birmingham, United Kingdom. (email: syed.shah2@bcu.ac.uk)

Dirk Draheim is with the Information Systems Group, Tallinn University of Technology, 12618 Tallinn, Estonia. (e-mail: dirk.draheim@taltech.ee)

Fig. 1. Illustration of enabling interoperability between multiple smart services in a smart city network.

providing a distributed and decentralized ledger, a blockchain offers a secure and transparent method for recording and verifying transactions, making it an ideal solution for enhancing data security in interconnected systems. A key characteristic of blockchains is their use of cryptographic techniques to ensure data integrity and immutability, thereby preventing unauthorized access and manipulation [11]. This feature makes blockchains well-suited for securing sensitive data, including personal information, financial transactions, and critical infrastructure data. Through consensus among network participants, a blockchain ensures that data cannot be modified without detection, enhancing the overall security of the system. In the context of interoperable smart services, blockchain technology plays a crucial role in improving data governance and standardization. Smart contracts, which are self-executing contracts stored on a blockchain, can establish rules and protocols for data exchange. This enables the enforcement of uniform data formats, protocols, and security measures across multiple systems and devices [12]. By leveraging smart contracts, smart cities can promote seamless communication and interoperability while ensuring data integrity and security.

Furthermore, blockchain technology empowers residents in smart cities by enhancing data privacy and control. With a blockchain, individuals can have ownership and control over their data, determining how it is accessed and utilized. This capability enables residents to maintain their privacy and make informed decisions about data sharing and utilization within the smart city ecosystem [13]. Using blockchain technology to address data security challenges in interoperable smart services holds significant promises. Its inherent security, transparency, data governance capabilities, and potential for increased privacy and trust make it an appealing solution for ensuring secure and accountable data exchange within smart city environments [14]. However, the adoption of blockchain technology in smart cities requires careful consideration of various factors such as scalability, interoperability, and regulatory compliance [15]. Scaling blockchain networks to handle the large volume of transactions in smart cities is a challenge that needs to be addressed to ensure its effectiveness. Interoperability among different blockchain platforms and other existing systems is equally essential for seamless integration and communication. Additionally, adherence to relevant regulations and compliance frameworks is crucial to ensure the legally correct and ethical use of blockchain technology in smart city applications.

Software-defined networking (SDN) is a cutting-edge communication framework that utilizes a programmatic approach to enable the execution of communication mediums. SDN controllers consist of control and data planes, which offer high customizability and efficient packet-forwarding capabilities. In the deployment of network infrastructure for smart cities, SDN is a valuable tool. However, certain limitations of SDN pose significant challenges to its utilization in smart cities, with the risks introduced by single points of failure [16] as an important example. In contrast, blockchain technology, as a distributed ledger, offers a means to create a tamper-proof and secure network [17]. By integrating blockchain technology with SDN, network administration tasks can be decentralized across multiple nodes, increasing the system's resilience to external interference. With its decentralized and tamper-proof nature, blockchain technology provides an additional layer of security to the network [18]–[20].

The combination of blockchain technology and SDN permits the distribution of network control across multiple nodes, thereby enhancing the network's resistance to assaults and malfunctions. The deployment of communication infrastructure in smart cities can effectively resolve security concerns by utilizing the inherent security features of blockchain, such as immutability and consensus mechanisms [21]. This integration has the potential to yield benefits as it decentralizes network control and leverages blockchain's security features, resulting in a more resilient, transparent, and secure system. These innovations mitigate the risks associated with cyberattacks, unauthorized access, and data manipulation, which are crucial factors for the communication infrastructure of smart cities [22]. While this integration bears promise, additional research and development are required to address challenges such as scalability, performance, and interoperability to ensure practicable implementation in real-world smart city environments. Blockchain and SDN together show considerable potential for delivering adaptive security solutions in smart cities. Blockchain technology offers safe and transparent transactions, while SDN provides a dynamic and adaptable network architecture capable of adapting to changing security needs. Together, these technologies may improve the interoperability of smart services in smart cities while also ensuring system security [23], [24].

A. Motivation

The increasing number of smart city services/applications across different domains presents a challenge in maintaining consistent security measures and enforcing dynamic governance policies. To address these challenges and ensure data security and privacy during collaborative tasks in heterogeneous smart cities, this research aims to propose an adaptive

TABLE I
SUMMARY OF IDENTIFIED RESEARCH GAPS THROUGH AVAILABLE LITERATURE.

Research questions	Summary	Challenges
What are the essential security prerequisites to establish a secure communication mechanism for syntactically interoperable smart services?	A secure communication mechanism for smart service interoperability must possess certain important characteristics to ensure the confidentiality, integrity, and availability of sensitive information shared between different entities. Authentication and trust management are two fundamental characteristics that must be considered [25]–[28]	Ensuring trust is essential in implementing security requirements for the interoperability of smart services. However, achieving seamless interoperability and authentication becomes challenging when diverse services utilize authentication mechanisms that do not adhere to shared syntactic standards, encompassing data formats, message structures, and protocols [29], [30].
What is the feasibility of utilizing blockchain technology for achieving the syntactic interoperability of smart services?	In the context of interoperability, blockchain technology enables direct interaction and secure data exchange among heterogeneous systems without intermediaries or centralized authorities. Its key features, including data consistency, data integrity, and smart contract execution, contribute to a secure mechanism for syntactically interoperable smart services [31], [32].	Blockchain technology offers benefits for smart service interoperability, but it also presents challenges such as scalability, security, and regulatory compliance. As the blockchain grows, scalability issues arise, impacting transaction processing speed in interoperability. A blockchain-based regulatory mechanism is necessary to govern security rules in smart service interoperability. Previous studies have explored the evolution of blockchain and its potential in enhancing security during interoperation of smart services [33], [34]. Resolving these challenges is crucial for leveraging blockchain effectively in achieving secure and efficient smart service interoperability.
What are the characteristics and capabilities of SDN that make it a viable communication architecture for achieving syntactic interoperable services in a smart city?	SDN serves as a feasible communication architecture for achieving syntactically compatible services in smart cities. Through its centralized management and dynamic policy enforcement, SDN enables the enforcement of syntactic standards and uniform communication protocols. This simplifies tasks such as message translation, protocol conversion, and traffic segmentation, particularly for services with diverse syntactic forms. Previous studies have explored the integration of SDN in smart city contexts, highlighting its potential in enhancing interoperability and communication efficiency [35]–[38]. Leveraging SDN as a communication architecture contributes to the establishment of syntactically compatible services in smart cities.	The implementation of dynamic policy enforcement in SDN to achieve syntactic interoperability of smart services presents a complex challenge. It requires the development and configuration of policies that effectively capture the desired behavior of interoperable services. However, addressing this challenge in SDN is challenging, as it involves ensuring accurate policy definition and configuration. Previous research studies have explored various approaches to tackle this issue, such as the use of Learning-based Security Automation (LSA) and data-driven methods [39]–[41].
Is there an adaptive security solution, utilizing smart contracts, available for addressing the syntactic interoperability of smart services in a smart city?	The availability of an adaptive security solution utilizing smart contracts for addressing the syntactic interoperability of smart services in smart cities is yet to be determined and requires further investigation. Existing literature provides evidence of solutions for achieving interoperability among devices in IoT networks. For instance, Ali et al. [42] introduced xDBAuth, a decentralized access control framework based on Blockchain and smart contracts for cross-domain IoT devices. Wang et al. [43] discussed blockchain interoperability for data exchange, considering four variabilities: Atomicity, Consistency, Isolation, and Durability. Villarreal et al. [44], and Lin et al. [45] proposed frameworks specifically designed for the IoT environment and services.	Existing literature also highlights the absence of an adaptive security solution for achieving interoperability of smart services in smart cities [46], [47]. However, blockchain technology, with its smart contract capabilities, holds promise for providing such an adaptive security solution [27], [42], [48]. Further research is needed to explore the potential of blockchain-based adaptive security mechanisms in enabling secure and interoperable smart services in smart city environments.

security framework that leverages smart security contracts. To identify the research gap, four research questions related to collaborative service security were identified. The summarized version of the literature based on these research questions is presented in Table I. Upon examining the existing literature, it becomes evident that ensuring secure and reliable communication among diverse intelligent services during interoperation in collaborative tasks requires consideration of the following factors:

1) Scalable Communication Infrastructure: Smart cities' interoperable services require a communication infrastructure that is capable of dealing with diverse smart city applications to provide collaborative tasks efficiently. This infrastructure should be scalable to accommodate the growing number of services and devices.
2) Integration of Blockchain Technology: The adaptation of blockchain technology is essential in smart city solutions. The decentralized nature of blockchain allows for safe and transparent transactions among numerous parties in a smart city ecosystem. Moreover, the scalability and versatility of blockchain make it a suitable choice for various sectors of smart city automation, enabling customized solutions to meet specific use case requirements.
3) Comprehensive Security Solution: A complete security solution for smart cities' interoperable services should encompass several security components, including authentication, authorization, access control, encryption, and monitoring. Additionally, the security solution needs to be dynamic and adaptable, capable of quickly recognizing and addressing new security challenges that may arise in the evolving smart city landscape.

By addressing these factors, the proposed adaptive security framework utilizing smart security contracts aims to enhance the security and privacy of collaborative tasks in smart cities. The integration of scalable communication infrastructure, blockchain technology, and comprehensive security measures can contribute to the effective and secure functioning of smart city services across diverse domains [49]–[51]. Our research contributes to the ongoing efforts of advancing secure and efficient collaborative services in smart cities, aiming to elevate the standard of living for citizens while addressing critical security concerns in a dynamic and evolving environment. By mitigating administrative challenges and prioritizing data security and privacy, the proposed framework represents a significant step towards the realization of smart cities' full potential in improving the lives of their residents.

B. Contributions

The main contributions of this paper are outlined as follows:

1) We propose a novel decentralized adaptive security governance management mechanism for enforcing rules and regulations for smart services interoperability within smart cities. The mechanism ensures that security policies are dynamically adjusted to address evolving security challenges and to maintain the integrity of collaborative tasks.
2) We introduce an adaptive security policy engine based on smart contracts, which enables the secure execution of interoperable smart services in smart cities. The smart contracts ensure that security measures are consistently enforced and provide a transparent and auditable framework for verifying compliance with security policies.
3) We demonstrate the feasibility of the proposed solution by implementing a use-case scenario that showcases the interoperability between decentralized services within a smart city environment. This implementation serves as a practical demonstration of how the proposed framework can be applied to real-world scenarios.
4) We conduct an evaluation of the proposed framework using performance metrics such as throughput, access time delay, and running time complexity for each service security smart contract. Additionally, the evaluation includes an analysis of the impact of varying ECC (Elliptic Curve Cryptography) key lengths on the performance of the security framework.
5) We perform an adaptive security assessment to evaluate the adaptiveness of the proposed security framework. This assessment involved analyzing the framework's ability to detect and respond to security threats, adapt security policies based on changing conditions, and ensure the overall resilience and effectiveness of the system.

The rest of the paper is organized in the following manner. Section II examines the literature related to security concerns in smart city frameworks and explores the concept of adaptive security governance based on smart contracts in smart cities. Section III provides insights into the integration and operation of SDIoT, multi-chain blockchain technology, and smart contracts within the architecture. Section IV provides an overview of the security framework by describing the submodules of the overall architecture. Section V discusses the execution of smart contracts in the engines for enacting smart governance. In Section VI, we discuss the description of the use case. Section VII provides the implementation details about the test bed used. Section VIII presents the evaluation results of the comparative system performance of the proposed framework. Section IX presents the experiments conducted for adaptive security assessment and finally we finish the paper with a conclusion in Section. X.

II. Related Work

The existing literature has extensively examined the concept of enhancing the intelligence of smart cities by integrating collaborative services. Various domains such as "smart living," "smart environment," "smart people," "smart economy," "smart mobility," "smart tourism," and "smart governance" have been identified as key areas for collaborative interaction within smart cities [52]–[54]. These domains encompass a wide range of sectors and highlight the multidimensional nature of smart city development. Understanding the collaborative dynamics within these domains is crucial for the successful implementation and management of smart city initiatives.

Rathee et al. [55] propose a trust formation method aimed at establishing a secure and safe communication environment

in smart cities. Their approach involves the integration of multiple technologies, including IoT, AI, drones, and robots, with the utilization of a trust mechanism. By incorporating trust-based mechanisms, their method aims to enhance the reliability and security of communication networks within smart cities, enabling the seamless integration and collaboration of diverse technological components. This approach has the potential to contribute to the establishment of a robust and trustworthy communication infrastructure in smart city environments.

The authors in [56]–[58] extensively discuss the security and privacy challenges associated with achieving syntactic interoperability in smart cities. Maciel et al. [59] highlight the importance of interoperability in the context of regulatory policies, addressing the limited adoption of commercial smart city technologies. On the other hand, The authors in [60]–[65] present a comprehensive methodology for evaluating semantic interoperability solutions, examining their strengths, weaknesses, and potential future directions in the context of smart cities. Additionally, Msahli et al. [66] specifically focus on the privacy protection mechanisms for sensitive data during interoperation in a V2X environment, including the identification of fake identities or certificates. These studies contribute valuable insights into the various aspects of security, privacy, and regulatory considerations surrounding interoperability in smart city environments.

Rahman et al. [67] propose a hierarchical blockchain-based platform called Blockchain-of-Blockchains (BoBs) to address data management, integrity, traceability, and transparency challenges in IoT interoperability across smart city organizations. Karumba et al. [68] present the Blockchain Agnostic Interoperability Framework (BAILIFF), focusing on notary services and cross-chain attestation for verification. In [69], [70], the authors explore interoperable services for drone tracking in smart city networks. Basheer et al. [71] discuss the integration of Fog Computing, IoT, and MANETs to achieve interoperable systems in sustainable cities. Chen et al. [72] present Vehicle as a Service (VAAS) as an interoperable service for vehicles. Batayneh et al. [73] highlights the lack of proper security governance in smart city development and its impact on collaborative tasks. Dua et al. [74] propose a secure message transmission system using elliptic curve cryptography for interoperable cars in smart cities. Reegu et al. [75] emphasize the role of blockchain in healthcare and its challenges in achieving syntactic interoperability. The authors in [76]–[78] analyze security and privacy challenges in syntactically interoperable services using blockchain in smart cities. Bellavista et al. [79] discuss the role of blockchain technology in the fourth industrial revolution (Industry 4.0) and highlight issues with interoperation between blockchains. Villarreal et al. [44] focus on blockchain's potential for interoperability and security of healthcare information. These studies contribute valuable insights into various aspects of interoperability, security, and privacy in the context of smart cities and blockchain technology.

Flanagan et al. [80] propose a decentralized protocol and data exchange framework for human-driven and connected autonomous vehicles to achieve interoperability in intersections. Xu et al. [81] introduce BlendCAC, a decentralized capability-based access control mechanism for large-scale interoperable IoT systems using smart contracts. Gilani et al. [82] propose a vertical SDN-based framework to enhance reliability and stability in smart home interoperable services. Rana et al. [83] focus on ensuring compatible and effective communication in the IoT during interoperation utilizing SDN-based architectures. Shamsudheen et al. [84] discuss the challenges of SDIoT in emergency services for disaster management. Abid et al. [85] explore the potential of SDN and NFV solutions for implementing interoperable IoT services. Banerjee et al. [86] propose a secure and scalable scheme for data collection and access control in IIoT using blockchain technology. Latif et al. [23] emphasize the importance of interoperability of blockchains and SDN in addressing energy and security issues in IoT networks. Kozhevnikov et al. [87] present a multi-agent system prototype that enables adaptive planning for the interoperable services of gas, water, and energy resources in smart cities. They also highlight the need for future research in the area of adaptive security to enhance the proposed system. This work contributes to the development of intelligent and efficient resource management in smart cities, addressing the challenges of interoperability and security.

Recently [88], [89], we have suggested the architecture of the multi-chain technology Alphabill, a blockchain platform, which allows for universal asset tokenization, transfer and exchange as a global medium of exchange. Alphabill has been designed, genuinely, for the purpose of universal asset tokenization [88]. As such, it shares objectives with other multi-chain blockchain technologies such as Polkadot [90], [91]. In service of universal asset tokenization, the Alphabill platform [88] aims at offering 1) systematic support for joining a transaction system to the platform, 2) systematic features for the interaction of hosted tokens, and, last but not least, 3) uncapped scalability. The key difference between Alphabill and Polkadot is in their approach to decomposition: Polkadot is a federation of multiple blockchains, whereas Alphabill is a single-partitioned blockchain. The Alphabill platform is currently under development and will be published as open-source software [88]. However, up to today[1], no open implementation of Alphabill is available. Therefore, we have decided to utilize the multi-chain blockchain technology MultiChain [92], [93] as the technological basis for the security framework that we propose in this paper.

III. System Overview

Our research proposes an architecture that addresses the security requirements associated with collaborative tasks in smart cities and interoperable services. This architecture is built upon the integration of three key technologies: Software-Defined Internet of Things (SDIoT), multi-chain blockchain technology, and smart contracts. By combining these technologies, our approach aims to establish a highly secure and resilient system capable of effectively managing potential security threats that may arise during the interoperation of smart services in smart city networks, particularly for collaborative tasks. Authentication and access control represent the core

[1]June 2023

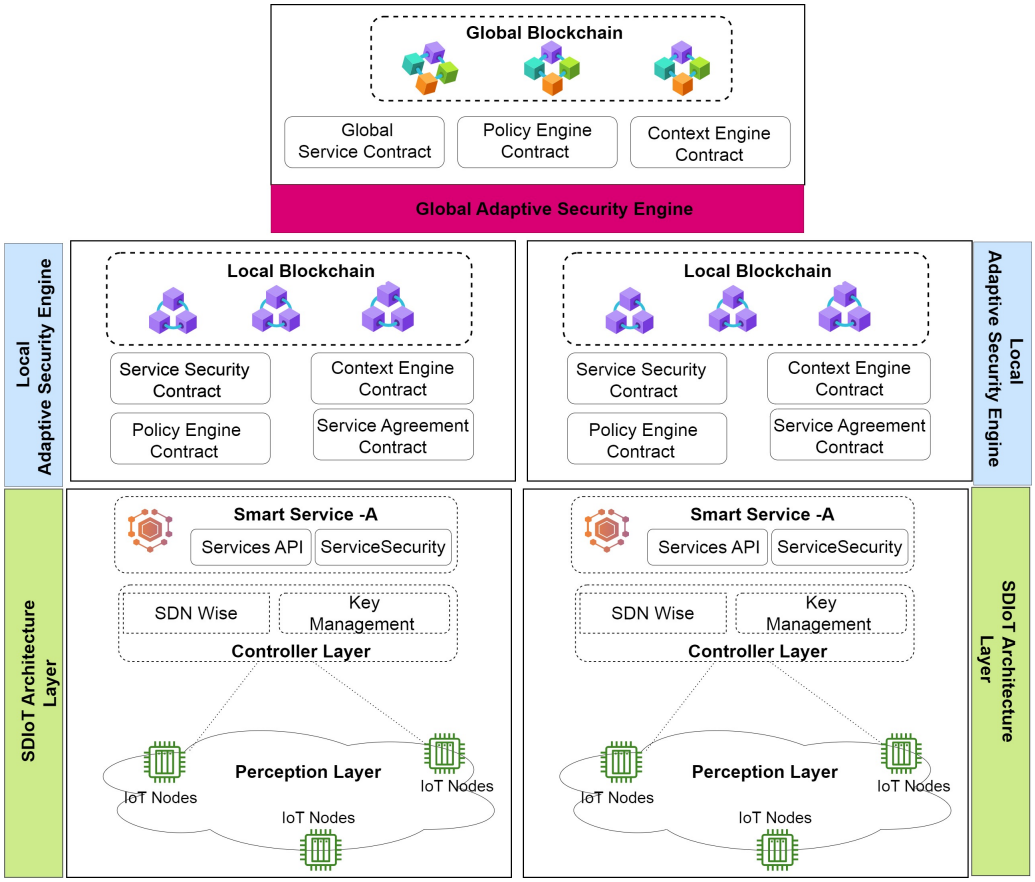

Fig. 2. Proposed security framework for collaborative services.

security components carefully incorporated into our proposed security framework. These components play a critical role in ensuring that only authorized entities are granted access to the system and its resources, thereby enhancing the overall security of the system.

In the following sections, we will provide a detailed discussion of the technologies and core components involved in our proposed security framework, as illustrated in Figure. 2. This discussion will offer insights into the integration and operation of SDIoT, multi-chain blockchain technology, and smart contracts within the architecture.

A. SDIoT (Software-Defined Internet of Things)

The Software-Defined Internet of Things (SDIoT) is a technology that makes it possible to build a dynamic and adaptive network of networked IoT devices, sensors, and other elements of the infrastructure for smart cities [94]. Smart city service nodes may be readily integrated and linked by SDIoT, enabling real-time data interchange, analytics, and decision-making. The architecture of SDIoT is based on the idea of SDN, which divides the control plane and the data plane of network devices to provide greater flexibility and programmability in controlling and running the network. We use SDIoT architecture [95] in our proposed framework in order to build a smart city network that enables interoperability between various smart services for collaborative tasks. The framework of SDIoT consists of three layers: the application layer, the controller layer, and the perception layer. In our proposed framework, Figure. 3 depicts the usual SDIoT architecture.

1) Application Layer: The application layer is a crucial component of the SDIoT architecture that enables the deployment of various smart city services [96]. By leveraging the application layer, cities can easily integrate and network smart services for seamless data exchange and real-time decision-making. This layer provides a flexible and scalable platform for developing and deploying innovative smart city services that can improve the quality of life for citizens. In our proposed framework, we take advantage of the application layer to integrate the open APIs of multiple smart city services, enabling us to implement a use case of interoperability between these services for collaborative tasks. By allowing different services to communicate with each other, our framework facilitates more efficient and effective delivery of smart city services.

2) Controller Layer: At the controller level, we have SDN-WISE controllers that are responsible for communication features and SDN programming benefits such as managing heterogeneity and scalability [16]. SDN-WISE is built on the IEEE 802.15.4 physical and MAC layers, and the Forwarding (FWD) layer processes incoming packets in accordance with the WISE Flow Table, which is changed by the Control Plane based on settings. The typical SDN-WISE architecture includes the default network module of SDN-WISE communication standards such as IEEE 802.15.4 for wireless nodes, topology discovery, packet processing, and the flow-wise [97].

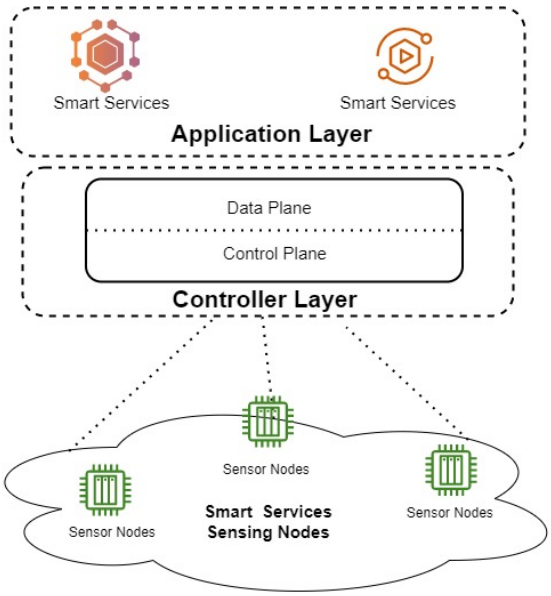

Fig. 3. Typical SDIoT architecture.

3) Perception Layer: The perception layer comprises the smart sensors and devices in our proposed framework that are working as sensing nodes for smart services. These sensing nodes are integrated into different smart city services and applications, enabling the city to collect real-time data and provide valuable insights for improving public safety and the quality of life of its citizens [98].

B. Multi-Chain Blockchain Technology

Multi chains such as Polkadot [90], [91], Alphabill [88], [89] and MultiChain [92], [93] are an advanced concept of blockchain technology that enables the creation of multiple interconnected and parallel blockchains within a single network. Multi chains are designed to address the limitations of traditional blockchain networks, such as scalability, interoperability, and privacy, by allowing multiple chains to operate independently and interact with each other. A multi-chain network typically consists of multiple interconnected chains, each with its own set of nodes, consensus mechanisms, and smart contracts. These chains can be customized to suit specific requirements, such as varying levels of trust, privacy, and performance, making them highly adaptable to different use cases. Multi chains are currently emerging with Web3 [99]–[103] (not to be confused with Web 3.0 [104], [105]), which takes blockchain to a next level by turning disintermediation ubiquitous – establishing disintermediation not only for basic payments, but also for a wide range of financial services, digital identities, data and business models [88], [102]. As such, the Web3 vision is about consolidating and integrating the fragmented landscape of specific blockchain visions expressed in the many initial coin offering (ICOs) that we have seen over the last decade; and multi chains are the natural fit to form the technological basis of Web3.

In our proposed framework, we integrate the open API [93] of the multi-chain technology, MultiChain [92] with the SDIoT architecture (see Sect. III-A) and a series of adaptive engines. We create a client-server relationship between the local chains and the global chain called a multi-chain client and server agent responsible for maintaining the validation attributes in the local blockchain and in the global blockchain. Following are the validation chains in client and server multi chain:

1) *Registration*: The registration validation chain is responsible to maintain the authenticity attributes of SDIoT architecture and local and global adaptive engines. It ensures secure and trustworthy interactions between them through the authenticity of their public keys, digital signatures, etc. during the interoperation of smart services in smart cities.
2) *Service Security Contract*: The service security contract validation chain is a critical component that ensures the integrity and security of the global and local service security contracts within the blockchain ecosystem. This validation chain is responsible for verifying and validating new security rules before they are implemented in the blockchain. It ensures that the security contracts adhere to the predefined criteria, policies, and regulations and that they are compatible with the overall security framework of the blockchain system.
3) *Global Service Agreement*: The agreement validation chain plays a pivotal role in the storage and management of agreements between diverse services within smart city ecosystems. It serves as a reliable repository for storing and safeguarding the local and global transaction agreements associated with service security smart contracts. These agreements are established between two smart services and serve as the foundation for collaborative tasks undertaken within smart cities. By diligently maintaining the local and global transaction agreements, the agreement validation chain ensures the coherence and consistency of service security smart contracts. This, in turn, facilitates seamless coordination and interoperability between smart services, thereby enhancing the efficiency and effectiveness of collaborative endeavors within smart cities.

C. Smart Contracts

A smart contract is a self-executing and autonomous agreement that is coded as a computer program and runs on a blockchain platform [106]. It defines and enforces the rules and conditions of an agreement between parties without the need for intermediaries. Once deployed on the blockchain, a smart contract automatically executes and enforces its predefined logic when certain conditions are met, without the need for human intervention [107]. In our proposed security framework we implement four types of security smart contracts in local and global adaptive engines. Multi-chain blockchains utilize these smart contracts to provide security automation during the interoperation of smart services in fulfillment of collaborative tasks.

1) Service Agreement Contract: is responsible for generating an agreement between service to the local adaptive engine and service to the global adaptive engine for

collaborative tasks between the interoperability of smart services in a smart city.

2) Service Security Contract: is a crucial component in ensuring the secure interoperability of smart services in collaborative tasks. It is responsible for creating local and global security contracts that outline the security requirements for the services involved. These security requirements typically encompass authentication, authorization, and access control mechanisms to protect the confidentiality, integrity, and availability of the services and their data.
3) Policy Engine Contract: is responsible for creating local and global policies for the interoperability of smart services based on local and global service security contracts, such as a policy to access collaborative messages between services.
4) Context Engine Contract: is responsible for executing the service security contract for local and global adaptive engines required during service interaction for collaborative tasks.

IV. Proposed Security Framework

The proposed security framework for collaborative tasks in smart cities is based on the integration of SDIoT architecture, a local adaptive engine, and a global adaptive engine. These components work together to ensure adaptive security during interoperation of smart services. The SDIoT architecture provides the foundation for dynamic and adaptable networking of IoT devices, sensors, and smart city infrastructure. The local adaptive engine and global adaptive engine, implemented as part of the blockchain, leverage smart contracts to automate security measures.

In the following subsections, we will provide an overview of the security framework by describing the submodules of the overall architecture. This includes the authentication and access control components that ensure the secure and resilient operation of the system. The local adaptive engine and global adaptive engine utilize the multi-chain blockchain and smart contracts to enforce security measures and ensure the integrity and confidentiality of data exchanged between different smart services in a collaborative task environment.

A. Key and Session Management

In our proposed security framework, we implement a key and session management module in the SDIoT architecture. It is responsible for providing digital identity to the IoT nodes of smart services in terms of key pairs and session keys. It is also responsible for storing it in the local repository in order to validate security attributes during the interoperation of smart services. Algorithm-1 outlines the implementation of the key management module, where the module will create Public key and Private key pairs for SDN controllers, IoT nodes, and smart services in the SDIoT architecture to provide them with digital identities.

We used the Elliptic Curve Cryptography (ECC) cryptographic algorithm for generating these key pairs. ECC is known for its strong security and smaller key sizes compared to traditional cryptographic algorithms, making it well-suited for resource-constrained IoT devices. We are using different key lengths such as ECC (128, 192, 256) in order to provide variation in authentication between different interoperable services.

Algorithm 1 : Key and Session Management

Require: $node_{ij}$ **Where** i,j is communicating nodes
Ensure: Session id for communicating nodes
Generate keys for controllers, IoT nodes for Key Length 128, 192, 256 bit
1: $Publickey_{ij}$= Openssl.generatePublic (ECC)
2: $Privatekey_{ij}$= Openssl.generatePrivate (ECC)
3: **Store in Key in repository accordingly**
Binding identities of nodes with smart service
4: Initialize Service JSON
5: Read the Keys from the repository accordingly
6: Service JSON=$Publickey_{ij}$, $Publickey_{ij}$

After generating the key pairs, the key management module generates session keys in order to provide secure communication between IoT nodes during collaborative tasks through the controller. Following is the high-level overview to distribute the session key from the key management module to IoT nodes securely,

1) At the beginning of the system, when IoT nodes are initialised, they need to join the network in the SDIoT architecture. IoT nodes will initiate the process by sending the encrypted joining request message through the SDN controller's public key and signing it with their own private key. We assumed the SDN controller and IoT nodes knew each other's public keys in the system.
2) SDN controller decrypts the message with its private key and the IoT node's public keys. After a successful verification process, the Key management module generates session keys.
3) Key management modules then use Elliptic Curve Diffie-Hellman (ECDH) to distribute session keys securely to IoT nodes from an SDN controller by encrypting the session key with the public keys of IoT nodes and signing it with the controller's private key.
4) IoT nodes after successfully decrypting the message, IoT nodes are able to use session keys for secure communication in order to provide confidentiality to the collaborative message during the interoperation of smart services.

B. Smart Service Management

Developing sustainable and effective urban environments requires smart services that can work together. To accomplish this, we created a dynamic application module in the SDIoT architecture responsible for integrating multiple smart service APIs into the security framework in order to provide collaborative tasks during the interoperation of different smart services. We also introduced the trust factor associated with smart services in the application layer to ensure that trustworthy and reliable smart services are used for collaborative tasks

between IoT nodes. Algorithm-2 refers to the implementation of the Smart Service management module in the application of the SDIoT layer. We implement the module in such a manner that enables us to integrate as many smart service APIs into the application layer. After the API attaches to the SDIoT application layer, the Service security module is responsible to provide security requirements with the help of the key management module along with the trust variable. We used trust variables for smart services in order to make trustworthy interoperability possible for collaborative tasks.

Algorithm 2 : Smart Service Management

Require: Service APIs
Ensure: Add Trust values to smart services
For adding multiple smart APIs
1: Service JSON[[Service APIs= APIs]
2: API= Number of smart APIs
3: **while** API>= 0 **do**
4: Service= API
5: Key Management (Service JSON)
6: Append Service JSON[[Trust= 0]
7: API= API-1
8: **end while**

C. Adaptive Security Engines

Adaptive engines are software components that facilitate the integration of the SDIoT architecture and the local and global blockchain. The engine is responsible for providing adaptive security solutions for distributed, decentralised smart services and collaborative task context execution during the interoperation of smart services. The adaptive engine is composed of a rule engine and a context engine. The rule engine is responsible for creating adaptive rules based on the security features of smart services such as authentication, authorization, and access control, also called service security contracts. The context engine is responsible for executing the crucial processes of authentication, authorization, and access control during collaborative tasks between different smart services.

In order to integrate the local and global blockchains with SDIoT architecture, there is a need for digital identity binding with the local and global blockchains; therefore, we used the blockchain's own key management modules in order to generate digital identities in the form of key pairs. The adaptive engine is responsible for sharing the public identities (public keys) of the local and blockchain systems with the SDIoT architecture through the same procedure as we did in the SDIoT architecture.

1) Rule Engine: The rule engine is responsible for generating security rules, both local and global, based on the dynamic security requirements of smart services during collaborative tasks in interoperability scenarios. Algorithm-3 outlines the implementation of service security contracts, which involves using an array of dictionary objects to add new security requirements for smart services, including authentication, authorization, and access control mechanisms. In our proposed framework, authentication requirements are met using public keys and session keys, while access control is determined based on the service trust factor threshold. After creating the service security rules for smart services, multi-chain client agents are responsible for creating a transaction in order to upload the latest service security rule for validation to the local blockchain and the server blockchain in the global adaptive engine. It is also responsible for receiving the updated service security rule called the global service security rule from the server blockchain of the global adaptive engine in the policy engine.

Algorithm 3 : Rule Engine

Require: Authentication, Authorization, Access
Ensure: Local Service Security Contract
For Local Adaptive Engine
1: **Define Authentication JSON object**
Read the SDN Controller Public Keys (128,192,256) bits
Read the Local Blockchain Public Keys (128) bits
Read the Global Blockchain Public Keys (128) bits
2: **Define Authorization JSON object**
Service Trust= Define the Trust Parameter;
3: **Define Access JSON object**
Adding Collaborative Service;
4: Compile the JSON as Local Service Security
5: GLobalService= MultichainClient(JSON)

2) Policy Engine: The policy engine in our proposed security framework defines as a set of rules, guidelines, or principles based on the combination of local and global security rules during the interoperation of smart services for the collaborative task in a smart city. The Rule engine is responsible to provide the Local security rule, and the global adaptive engine provides the mechanism through the global blockchain in order to enforce the global security rule to the local security rule. The Policy Engine is responsible to fetch the latest updated service security from the Global Adaptive Engine server Blockchain and convert it into JSON Format as shown in the Algorithm 4. In order to execute of Policy the module is responsible to forward the policy to the Execution Engine. The engine has its own policy management operations (such as adding, deleting, and appending new policies) in order to provide flexibility during the interoperation of smart services.

Algorithm 4 : Policy Engine

Require: Fetch the service security from the Blockchain
Ensure: Send the JSON Contract to the Execution Engine
1: Fetch the contract from the Blockchain
2: Convert the contract in JSON format
3: Initialize Global Service, JSON=[]
4: a= Use Multichain command to fetch the contract
5: Append Global Service, JSON= a

3) Context Engine: The context engine is responsible for executing the crucial processes of authentication, authorization, and access control for the collaborative task during the interaction of multiple services by fetching the service security

contracts from the local rule engine and global rule engine. Algorithm-5 refers to the implementation of the context engine where authentication of the identities of the SDN controller, blockchain, and communicating nodes through the local and global blockchain is the first step. After successful verification of authentication, the authorization is completed with the help of trust assessment in order to access the collaborative task during the interoperation of smart services.

For trust assessment we are using a modified version of Boa et al. [108] as defined in Definition. 1.

Definition 1 (Trust Assessment): Given two smart services i and j, a time point t, a time difference to a previous trust assessment ΔT, a *direct trust assessment* $D_{ij}(t) \in [0,1]$ at time point t, and a *trust factor* $\alpha \in [0,1]$ (indicating how much the trust assessment depends on direct assessment), we define the *trust assessment of service interaction* (between smart services i and j at time point t), denoted by $T_{ij}(t) \in [0,1]$ (with 0 called *untrusted*, 0.5 called *semi-trusted*, and 1 called *trusted*) as follows:

$$T_{ij}(t) = \alpha\, D_{ij}(t) + (1-\alpha)\, T_{ij}(t - \ \Delta T) \qquad (1)$$

Algorithm 5 : Local Context Engine

Require: Authentication, Authorization,Access
Ensure: Context Execution

```
Trust=0
while True do
    Bool=Verfication of Authentication Variable
    if Bool == "True" then
        x= 1 − α
        ServiceTrust= x ∗ Trust + α*0.001
        Trust= Service Trust
        Send the Service Trust to Blockchain
        if Trust >= "PragramaticalVariable" then
            Access = Grant
            Call collaborative task
        end if
    end if
end while
```

V. Workflow of Smart Contract Execution

In this section, we discuss the execution of the smart contracts present in the engines for enacting smart governance in a smart city. We implement the concept of smart governance in a global adaptive engine responsible for enforcing the smart city governance rule on the smart services running in the smart city. Smart contracts are self-executing programs that run on adaptive engines in our proposed security framework, designed in a way to provide automation for security. We implement the smart contract with the help of a Python script that integrates with "multi chain" APIs. Following is the workflow of smart contract execution involved in the interoperation of smart services and smart governance in a global adaptive engine:

1) Generation session token
2) Adding local service security contract
3) Adding global service security contract
4) Governance execution process

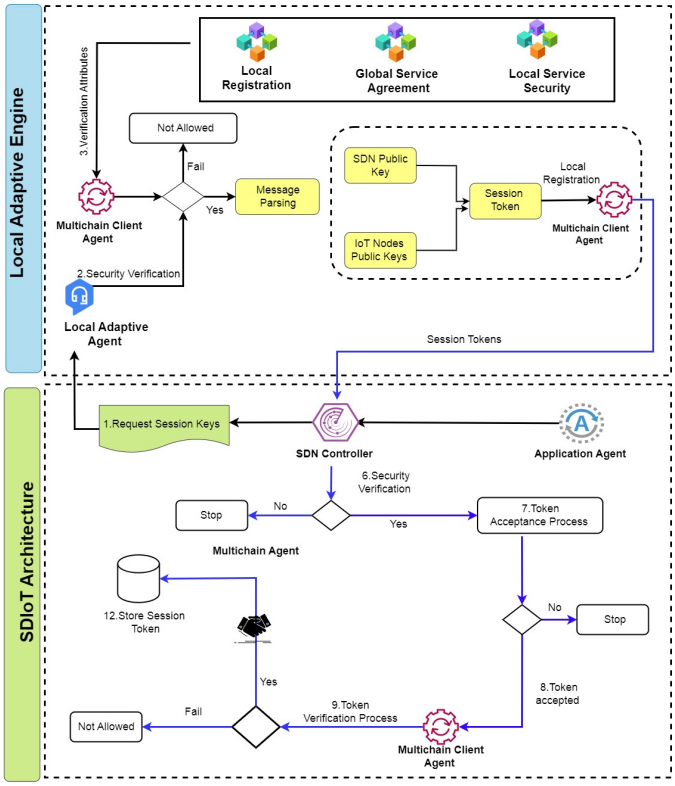

Fig. 4. Workflow of session token generation.

A. Generation Session Token

In the proposed security framework, we used session keys in conjunction with private and public keys for communication between SDIoT architecture and adaptive engines on blockchain in order to provide additional data security features during a collaborative task. Figure 4 represents the workflow to generate a secure session token for communication. The following are cryptographic steps involved to generate a secure session token for communication

1) The SDN-wise controller generates key pairs and sends a request message by including the public identities of nodes in the message as $node_{ij}(pub)$ and the hash of the SDN controller public key. The request message is then encrypted with the public keys of the local blockchain and signed with the controller's secret key. Equation (2) shows the structure of the request message. It is assumed that the local global blockchain and SDN controllers have already shared their public keys with each other by sharing the hash of their public keys.

$$Message_{Enc} = \left[(node_{ij}(pub)||hash(SDN_{pub}))_{sk}\right]_{pb} \qquad (2)$$

2) The local adaptive engine receives a request message from the controller and decrypts the received message through the private key of the local blockchain and the public key of the SDN controller as shown in Equation (3).

$$Message_{dec} = \left[(node_{ij}(pub)||hash(SDN_{pub}))\right] \qquad (3)$$

3) After successfully decrypting the request message, the adaptive engines first verify the legitimacy of the SDN

controller by fetching the SDN controller's public key from the validation chain of the local blockchain.

4) After successful validation of the SDN public key from the local blockchain, local adaptive engines call smart contracts for session token generation. We implement it with the help of multi-signature, this requires the public keys of the local blockchain, along with the public key of IoT nodes as shown in Equation (4).

$$Session_{ID} = \left[(Multisig(node_{ij}(pub), (chain_{pub}))\right] \quad (4)$$

5) Multi-signature protocol returns the session id which then stores in the validation chain of local and global chains and forwarded to the SDN-wise controller.
6) The Encrypted $Session_{id}$, then send back to the controller by encrypting with the public key SDN controllers and signing it with the Local and the global blockchain private key. Equation (5) shows the sent message structure.

$$SessionContract_{Enc} = \left[(Session_{sk})\right]_{pb} \quad (5)$$

7) The SDN controller verifies the legitimacy of the received message.
8) Session token acceptance process is started.
9) If the session token is accepted, the legitimacy of the session token security attribute will be again verified through the multichain client agent.
10) After successful verification of the security attribute of the session token from the multichain client agent, the session token is stored in the local repository.
11) After successfully validating of $session_{id}$ from the local and global validation chains, the $session_{id}$ is stored in the local repository of the controller. In last the controller call the key distribution module in order to distribute the session key to the IoT nodes for secure communication during the interoperation of smart services during a collaborative task.

B. Adding Local Service Security Contract

Interoperability between services have require an agreement when adding new security requirements to the service of a smart city. In our proposed framework we create a smart contract for adding new security requirements that generate a service-level agreement for interoperability. For access to collaborative tasks, interoperable services should agree on the service-level agreement. Figure 5 shows the workflow steps. The cryptographic steps involved in the execution of the workflow are given below,

1) The application agent sends the request message to the SDN controller. The SDN controller is responsible to provide integrity, and confidentiality through ECC cryptographic suit by adding a security tag of session-id to the requested message and encrypting the request message with the public keys of the local blockchains. After providing confidentiality and integrity to the requested message SDN controller signed the message with the secret key of the IoT nodes and will send it

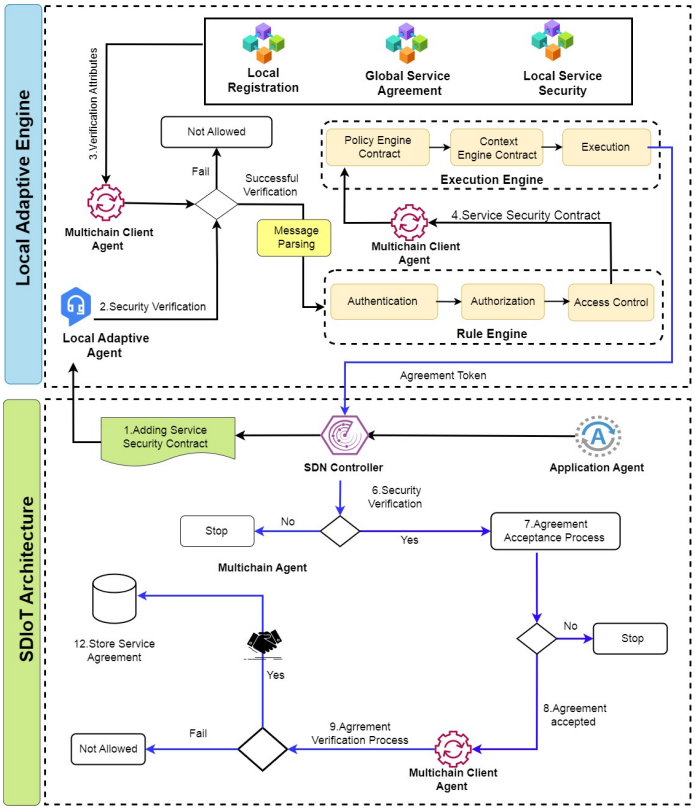

Fig. 5. Workflow of adding local service security.

to the Local adaptive engine. Equation (6) shows the structure of the request message. It is assumed that the local global blockchain and application agent have already shared their public keys with each other by sharing the hash of their public keys.

$$Message_{Enc} = \left[(Request||Hash(SessionKey))_{sk}\right]_{pb} \quad (6)$$

2) The local adaptive engine receives a request message from the application agent and decrypts the received message through the private key of the local blockchain and public key of IoT nodes as shown in Equation (7) as,

$$Message_{dec} = \left[(Request||Hash(SessionKey))\right] \quad (7)$$

3) After successfully decrypting the request message, local adaptive engines verify the legitimacy of the session keys by fetching the session key from the validation chain of local blockchains and then comparing its hash with the received hash of the session key in the decrypted message.
4) After successful validation of the hash of the session key from the local blockchain, The rule engine module is responsible to added new local security requirements for smart applications based on authentication, authorization, and access control also called local service security contracts. The "multi chain" client agent generates a transaction of Local service security contract in the local blockchain and sends it to the global adaptive engine in

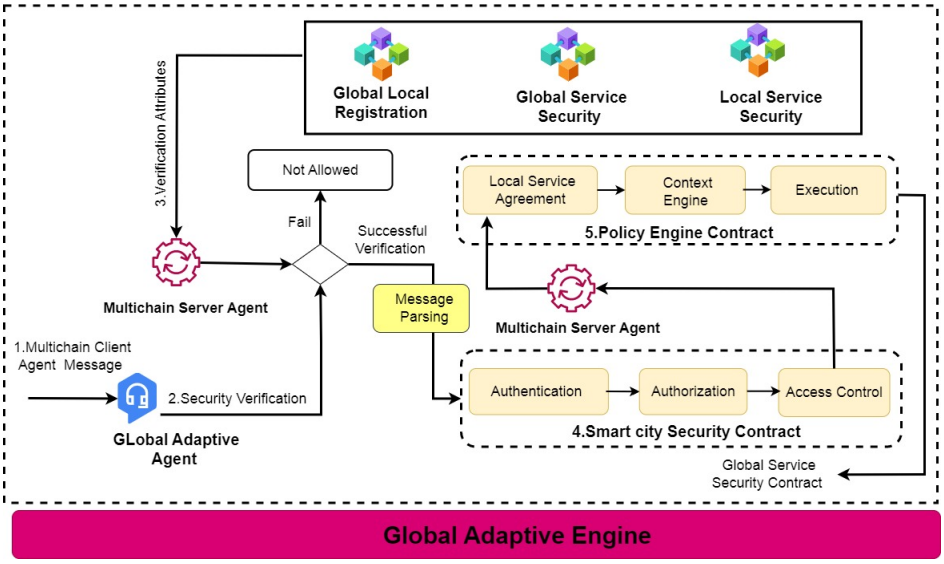

Fig. 6. Workflow of global service security contract.

order to add administrative security requirements to the local service security.

5) In the next step, the policy engine module fetches the updated global service security contract from the "multi chain" client agent along with the contract transactional Id also called the Service agreement. The policy engine sends the service agreement message to the context engine in order to provide confidentiality, and integrity to the service-level agreement message. Context Engine encrypts the message with the public keys of the SDN controller and provides integrity by integrating the message with the hash of the session id. Equation-8 shows the send message structure,

$$Contract_{Enc} = \left[(Contract||Hash(SessionKey))_{sk}\right]_{pb} \tag{8}$$

6) The application agent is responsible to decrypt the received agreement message from the local adaptive agent.
7) After successfully decrypting the message, the agreement message is forwarded to the acceptance process.
8) If the agreement is accepted then again the verification process is performed in order to confirm the legitimacy of the service agreement token with the help of "multi chain" client agent.
9) After successful validation of the service agreement token, the service agreement token is accepted and stored in the local repository.

C. Adding Global Service Security Contract

The global security requirements for smart services entail combining the local service security requirements of smart services with the administrative service security requirements for collaborative tasks. These requirements must align with the service requester's specifications for service interoperability. In our proposed security framework we integrate the global adaptive engine with the local adaptive engine through a "multi chain" client agent. Figure 6 shows the pictorial view Adding Global Service security contract.

The Global adaptive engine is responsible to create smart city administrative service security contracts in order to enforce smart city administrative security policies during the interoperation of smart services. It is also responsible to receive a message from a "multi chain" client agent through a global adaptive agent. Following are the cryptographic steps involved in the making of global security rules for collaborative tasks during the interoperation of smart services.

1) The local adaptive engine incorporates "multi-chain" client agents to facilitate the creation of transactions for local service security contracts. These transactions are intended for storing the most recent local service security contract within both the local blockchain and the global blockchain, utilizing a client-server architecture. Once the transaction is generated, the "multi-chain" client agent within the local adaptive engine encrypts the message using the public key of the global blockchain and signs it with the private key of the IoT nodes. To ensure data integrity, the message is integrated with the hash of the session ID of the IoT nodes. The structure of the received message is depicted in Equation 9 as,

$$M_{Enc} = \left[(Agreement_{Txid}||Hash(Session))_{sk}\right]_{pb} \tag{9}$$

2) The global adaptive engine decrypts the received message through the private key of the global blockchain and the public key of IoT nodes' public key as shown in (10) as,

$$M_{dec} = \left[(Agreement_{Txid}||(Hash(Session))\right] \tag{10}$$

3) After successfully decrypting the request message, global adaptive engines verify the legitimacy of the session keys by fetching the session keys from the global blockchain.

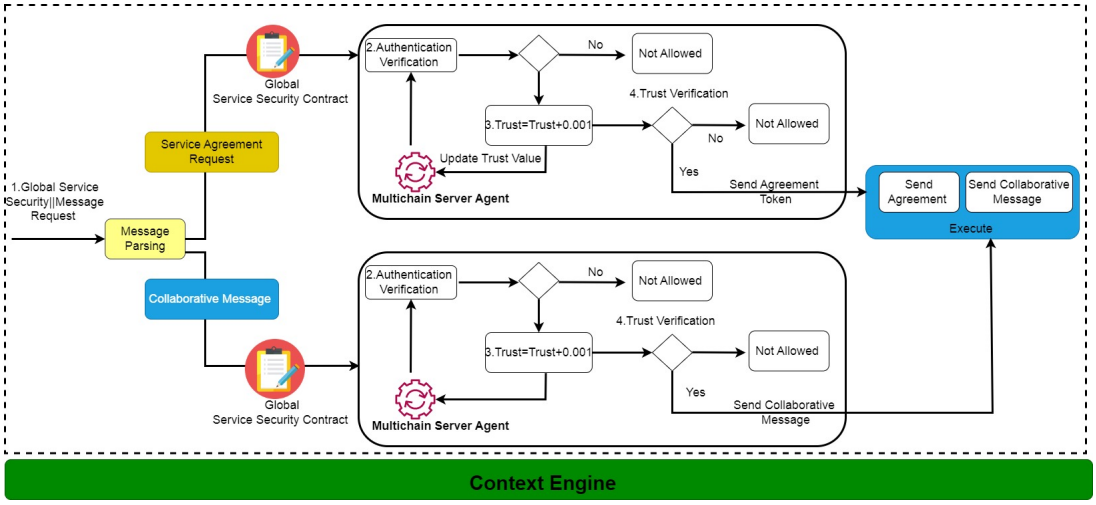

Fig. 7. Workflow of context engine.

4) Once the transactional ID is successfully validated, the global adaptive agent retrieves the most recent service security contract from the global blockchain using this ID. Subsequently, the global adaptive agent merges this local service security contract with a smart city administrative security contract to form a global service security contract. The structure of the global service security contract message is represented by Equation 11, where A represents the local service security contract and B represents the administrative security contract of the smart city.

$$Global\ Contract = \left[A||B\right] \tag{11}$$

5) Subsequently, the context engine initiates the generation of global service security transactions to store the global security contract in both the local and global blockchains. The context engine securely forwards the transaction ID to the local adaptive engine for validation purposes. This is accomplished by encrypting the transaction ID using the public key of the local blockchain and signing it with the secret key of the IoT nodes.

D. Governance Execution Process

The governance execution process is facilitated by the context engine, which plays a crucial role in coordinating and executing collaborative tasks between smart services. One of the key processes within the local adaptive engine is the execution of the context execution contract, which enables the execution of collaborative tasks. Within the context engine, we have implemented two sub-smart contracts that respond to different message requests. The first sub-smart contract handles requests for service agreement collaboration, while the second sub-smart contract handles requests for executing collaborative messages after the agreement has been established. Figure 7 shows the workflow of the context engine execution process. Following are the steps involved in the execution of the context engine,

1) The request message is forwarded to the context engine after successful security verification from the local blockchain. At the beginning of the context engine, the requested message authenticity will verify first.
2) After successful verification of the authenticity of the requested message, the trust assessment process is started. The trust assessment process is based on an iterative procedure in which the trust assessment value will increase continuously as an incentive if authenticity is verified. When the required threshold for collaborative trust is reached, the message is forwarded to the execution process.
3) In the execution process collaborative response message is created. We are implementing two types of response messages based on the request message such as a Service agreement token response message and a collaborative message response (Access grant, Access not grant). The execution process is also responsible to provide security requirements to respond to messages.

VI. Description of the Use Cases

The demand for emergency response systems in smart cities has significantly increased due to factors such as the rapid growth of urban populations and the escalating risks associated with emergencies and disasters [109], [110]. As smart city technologies continue to advance, there is a growing need for interconnected services to operate seamlessly and collaboratively, particularly in emergency situations. The effectiveness of a collaborative emergency response system within a smart city hinges on its ability to swiftly and efficiently execute responses to actual or anticipated emergency situations, taking into account the interoperation of multiple smart services. The timely and coordinated execution of emergency response actions is vital in ensuring the safety and well-being of citizens and minimizing the impact of emergencies on infrastructure and resources. The speed at which a collaborative emergency response system can execute is of utmost importance. It determines the system's ability to gather and analyze relevant

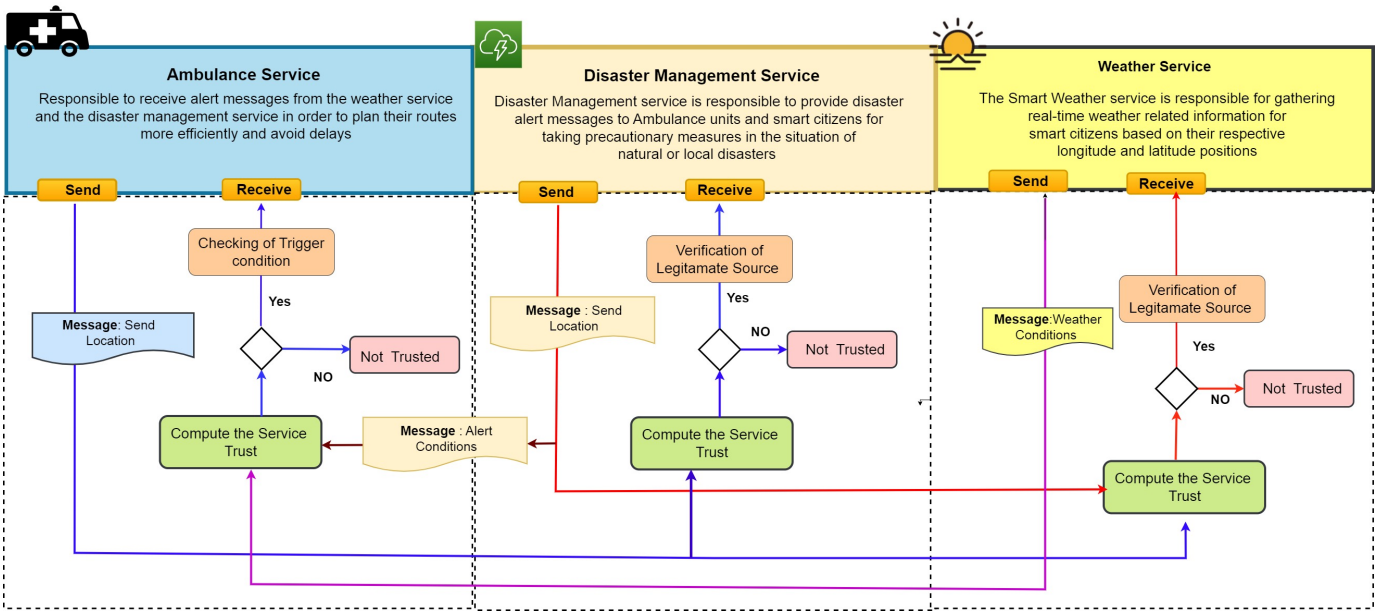

Fig. 8. Communication workflow of interoperable services for smart emergency response.

data, assess the severity of the situation, and coordinate response efforts across multiple services and stakeholders. By ensuring a fast and efficient execution process, smart cities can enhance their emergency preparedness and response capabilities, ultimately safeguarding the lives and properties of their residents.

In order to evaluate the feasibility of our proposed security framework, we consider the use case of collaborative emergency response services in a smart city, as shown in Figure. 8, where three services interact with each other in order to execute disaster emergency response systems. Following is the detailed information of the three interoperable services involved in the execution of emergency response systems.

1) Smart Disaster Management Service: In our research, we have developed a disaster management service using Python socket programming to establish a server-client code structure. This service operates as a server node that collaborates with other smart services. To facilitate collaborative request tasks during interoperation, we have implemented two distinct functions: send and receive. The receive function plays a crucial role in acquiring essential data for disaster management. It receives weather data from the weather service, enabling the system to monitor current weather conditions. Furthermore, it receives the current location data of ambulance users from the ambulance services, providing real-time information on the location of potential victims. Conversely, the send function is responsible for transmitting alert messages based on the received weather data from the weather service to the ambulance service. This allows the ambulance service to proactively respond to potential emergencies by taking necessary precautionary measures. Additionally, the send function also relays the current location of the disaster server to the weather service, facilitating effective coordination between the two services. Through the integration of these functionalities, our disaster management service enhances the interoperation between various smart services, enabling real-time data exchange and collaborative decision-making during emergency situations.
2) Smart Weather Service: This service plays a vital role in collecting real-time weather-related information for smart citizens based on their respective longitude and latitude positions. To implement this service, we integrate publicly available Openweather APIs, which offer various attributes. For our purposes, we focus on the weather.id feature, which provides current and predicted information about rainfall based on the smart citizen's geographical coordinates. The weather.id feature classifies weather conditions into three distinct ranges: 800, indicating "locally drizzling"; 900, indicating a prediction of "locally heavy rain"; and 1000, indicating a prediction of "urban flood." By utilizing this feature, we can effectively determine the intensity of rainfall in a given location.
In addition to API integration, we have developed two essential functions: send and receive. The receive function is responsible for acquiring data from the weather API, allowing us to retrieve real-time weather information. It also receives the current location of the Ambulance user from the ambulance service, enabling us to monitor their geographical position. On the other hand, the send function handles the task of sending alert messages to both the disaster services and ambulance services when the weather API data indicates "locally heavy rain predicted," "urban flood predicted," or "drizzling" conditions. This proactive communication ensures that the appropriate authorities and services are promptly alerted to potential risks or emergencies. By implementing these functions and integrating the

Openweather APIs, the Smart Weather service enhances the overall coordination and response capabilities of smart city systems, ensuring that citizens and relevant services are well-informed and prepared for weather-related events.

3) Smart Ambulance service: This service ensures effective communication and efficient response during emergency situations. By actively exchanging information with the disaster services and the smart weather service, the Smart Ambulance service can provide timely assistance and contribute to the overall safety and well-being of smart city residents. The smart ambulance service is a crucial component of our system, implemented using Python socket programming in a server-client code structure. As a client node, it interacts with both the disaster services and the smart weather service to facilitate collaborative task requests. To achieve this, we have developed two distinct functions: send and receive. The receive function of the smart ambulance service plays a pivotal role in receiving alert messages from the disaster service and the smart weather service. These alert messages provide valuable information about potential emergencies or weather-related events that require the attention of the ambulance service. By receiving and processing these messages, the smart ambulance service can quickly respond to critical situations. Conversely, the send function is responsible for sending beacon messages to the connected services. These beacon messages serve as updates or status reports from the Smart Ambulance service, allowing other services to stay informed about its current activities and availability. This facilitates seamless coordination and collaboration among the different components of the system.

Table IV presents the distinct security policies employed by each service involved in our proposed emergency response system during the interoperation of smart services in the smart city. Moving forward, the subsequent section will focus on the testbed scenarios meticulously crafted to assess the efficacy of the proposed security framework. We will explore the evaluation parameters utilized to gauge the performance of the security framework across various scenarios.

TABLE II
SYNTACTICALLY INTEROPERABLE SECURITY RULES FOR COLLABORATIVE TASK BETWEEN SMART SERVICES.

Smart Disaster Management Service	**Smart Weather Service**	**Smart Ambulance Service**
-Authentication with 128-bit ECC Keys -Trust 0.003 -IEEE 802.15.4	-.Authentication with 192-bit ECC Keys -Trust 0.003 -IEEE 802.15.4	-Authentication with 256-bit ECC Keys -Trust 0.003 -IEEE 802.15.4

VII. TESTBED AND IMPLEMENTATION DISCUSSION

We simulate a smart city network using the COOJA network simulator, which is an open-source tool based on the Contiki operating system [111]. COOJA is widely used for simulating wireless sensor networks and IoT networks [112]. The architecture of Contiki is built having multiple modules that offer features like process management, memory management, and inter-process communication to the IoT nodes or motes It also supports network protocols such as IPv6, RPL, 6LoWPAN, and CoAP for low-power, lossy networks [113]. We integrate the Contiki operating system with the SDN-Wise controller for SDIoT architecture. SDN-Wise is middleware it offers a programming abstraction that lets programmers build high-level Internet of Things services and applications while concealing the intricate network architecture that underlies them. The SDIoT network architecture and traffic flows are managed centrally by the SDN-Wise controller [114].In order to build the proposed security framework, we combine the "multi chain" blockchain with the SDN-WISE Contiki framework, in order to provide a secure mechanism during the interoperation of smart services for collaborative tasks in smart city.

To assess the effectiveness of our proposed security framework, we examine its system performance as well as the execution time performance of the smart contract implemented in the adaptive engines that utilize blockchain technology. The performance of the proposed framework is significantly dependent on the "multi chain" memory pool and the SDN controller's memory capacity during the collaborative requests/response messages flow. The "multi chain" memory pool temporarily stores unconfirmed transactions before they are published to the blockchain. However, as the number of collaborative requests/response messages from SDN-WISE to "multi chain" increases, the memory pool may become a bottleneck, leading to reduced system performance. In addition to this, the memory capacity of the SDN controller is critical to the performance of the framework as it tracks network topology and traffic flows.

Our testbed is designed to focus on the four critical processes of message flow necessary for achieving interoperability of services in smart cities. To evaluate the system's performance, we gradually increase the number of collaborative smart services and IoT nodes in the smart city. We apply varying message flow loads, ranging from 100 to 5000, with corresponding delay differences of 600ms, 120ms, 60ms, 30ms, 15ms, and 5ms. This allows us to accurately measure and analyze the system's performance under different scenarios and workloads. We are focusing on the following three workflows during the interoperation of services for emergency collaborative tasks in a smart city,

1) Service-level agreement between interoperable services.
2) Sending and receiving an emergency request during interoperation of services.
3) End-end message sending and receiving.

The testbed for our system was implemented across four distinct machines. Among these, three machines function as decentralized smart services client blockchain nodes, while the remaining machine operates as the server blockchain node, referred to as the global blockchain. The configuration details of each machine are provided in Table-III, which outlines the specifications and settings of the hardware used. To realize the network infrastructure required for our use case, we have

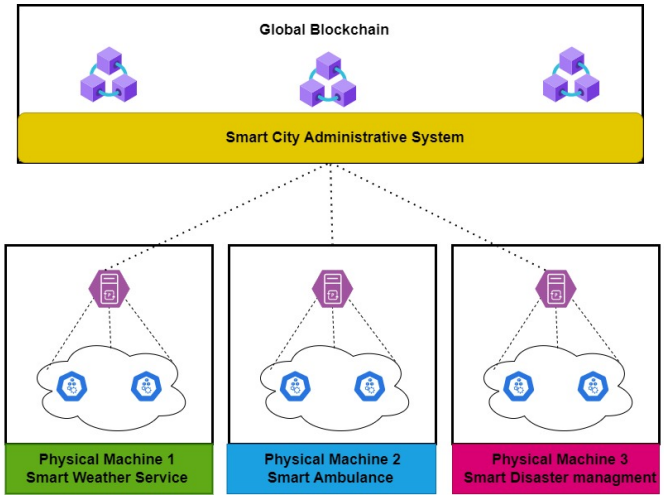

Fig. 9. An illustrative network architecture of the proposed use case.

presented an illustrative network architecture as depicted in Figure 9.

TABLE III
HARDWARE CONFIGURATION OF THE FOUR PHYSICAL MACHINES.

Physical Machines 1, 2, 3, 4	
CPU (Processor)	Intel® 7th Gen Intel® Core™ i7 (6700)
Memory	16 GB DDR
Chipset	Intel® H110 Chipset
Hardrive	256 GB Solid State Drive SATA

A. Service-Level Agreement Request between Interoperable Services

In order to achieve interoperability between smart services, it will be necessary to establish service agreements between them. The first step towards achieving interoperability is to generate a global service agreement that encompasses all relevant smart services. Figure 10 illustrates the workflow for generating and accepting the agreement for interoperability between these services

1) Service A is sent the request through the application agent to the SDN controller in the SDIOT architecture. The message structure of the request message is shown in (12) (where X represents the name of the interoperable service such as Service A).

$$M_{Enc} = \left[(Request||X||Hash(SessionKey))_{sk}\right]_{pb} \quad (12)$$

2) The Local Adaptive agent receives the encrypted request from the SDIoT architecture. The Local Adaptive agent first verifies the message authenticity along with the authenticity of IoT nodes with the help of local blockchain validation chains.
3) After successful verification of the authenticity of Iot nodes and request messages. The request is forwarded to the rule engine.
4) From the rule engine first the local security rule is generated in JSON format and sent to the "multi chain" client agent in order to store the latest local service security transaction to the Local blockchain.
5) As we implemented "multi chain" as client and server nodes in the proposed security framework. After creating a transaction of local service security in the local blockchain it will send a copy to the global blockchain in the global adaptive engine through the global adaptive agent in Step 5. The sent message structure is shown in (13) where Agreement Txid is the transactional id of the local service security.

$$M_{Enc} = \left[(Agreement_{Txid}||Hash(SessionKey))_{sk}\right]_{pb} \quad (13)$$

6) After security verification of the session id from the global blockchain, the request is forwarded to the Global rule engine. The global rule is responsible to enforce administrative security for local services in smart cities.
7) After the generation of the global security rule in JSON format from the rule engine, the Transaction is generated with the help of a "multi chain" server agent and forwarded to the execution engine of the global adaptive security engine.
8) The execution engine first searches the local service security contract with the help of the transactional id of the local service contract and then concatenates the contract with the global service contract in the context engine and forwarded it to the execution process.
9) Through the execution process, the global service security contract transaction ID is returned to the "multi chain" client agent through the global adaptive agent.
10) In Step-10, the Global security contract will be fetched from the local blockchain. Now the Local Execution engine incorporates the local security requirement along with the administrative security requirement and converts it into JSON format in order to forward the request to the local context engine.
11) The Local Context engine verifies the legitimacy of global service security contracts. Figure-7 shows the workflow of verification of global service security contracts and forwards it to the local execution process.
12) The Local execution process is responsible to forward the request to the requested service.
13) Service-B on receiving a request perform the same operation as was performed in steps 2-14 for service-A.
14) After successful security verification, the local adaptive agent sends the message back to the application agent of Service A in Step-13.
15) The application agent forwards the message to the SDN controller for security verification.
16) After successful security verification of the received message, the process of acceptance of the service agreement is started.
17) If the agreement is accepted, the legitimacy of the service level agreement security attribute will be again verified through the multichain client agent in Step-16.
18) The service level agreement after successful security verification is stored in the local repository.

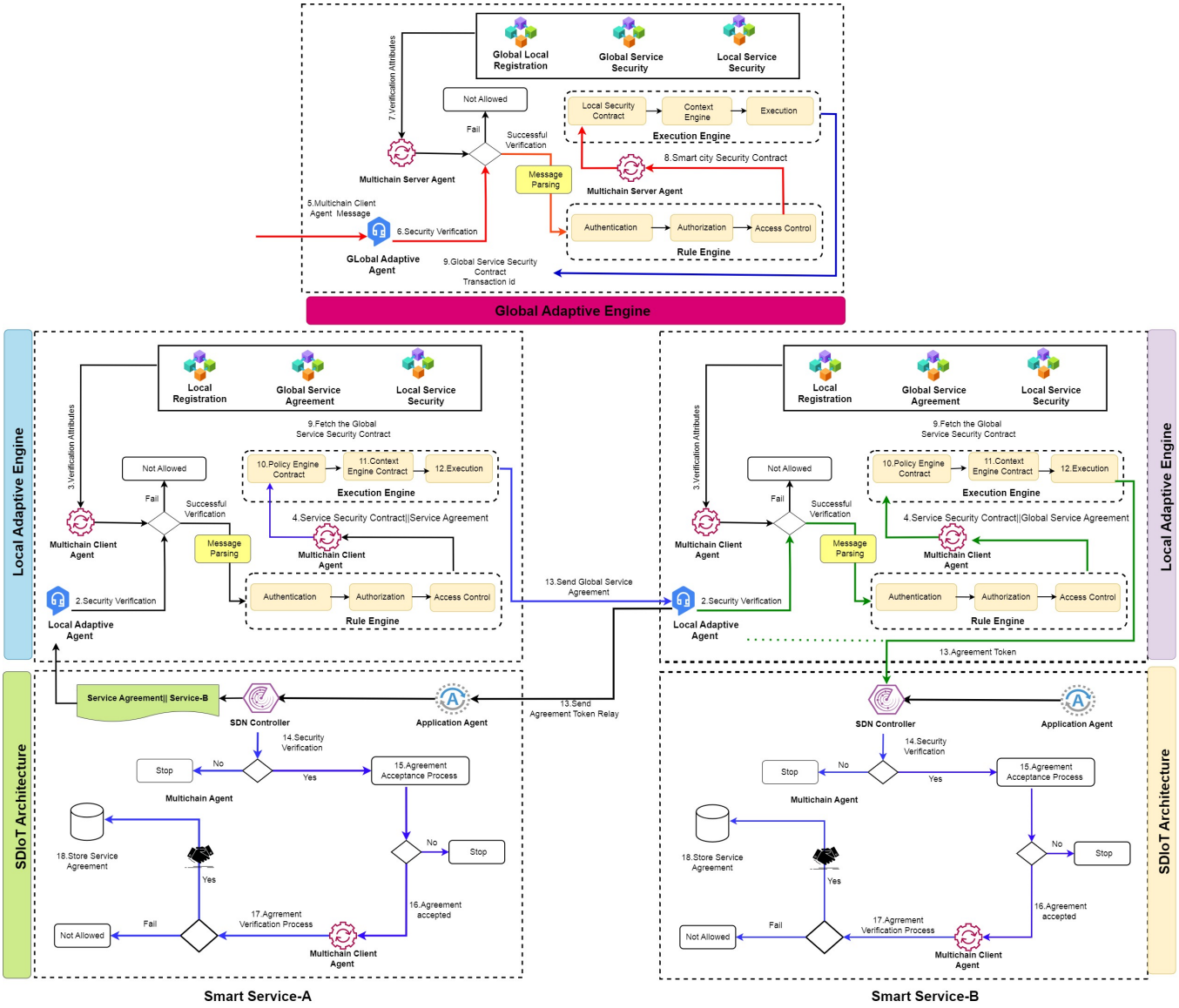

Fig. 10. Service-level agreement between interoperable services.

By analyzing the data presented in Table IV, we observed a notable increase in throughput when three services interacted to request a service-level agreement, particularly when each service was equipped with 50 sensing nodes. This indicates that the system was able to handle a higher volume of transactions and process them more efficiently. The improved throughput of the system had a direct impact on the execution time of smart contracts in both execution engines. With the increase in throughput, the execution time of smart contracts significantly improved, demonstrating the enhanced performance of the execution engines, particularly when faced with a dense influx of collaborative request messages. We noticed that the received request message delay from the global adaptive engine to interoperable service had also improved with the increased throughput of the system which actually represents the responsiveness of our proposed security framework. We also noticed that at the maximum number of requests, the performance matrix of the system goes down. Figure 11 shows the graphical representation of the tabular result. In the second experiment, the performance matrix is evaluated by increasing the number of sensing devices during the seamless interaction of smart services for service-level agreement requests. From Table V we noticed that the performance matrix is getting worst when the number of IoT nodes is increased if we compared it with the previous result, this depicts the SDIoT architecture plays a vital role in the performance matrix of the security framework. Figure 12 depicts the graphical representation of the performance matrix as the number of IoT nodes increases. This performance matrix provides valuable insights into the system's behavior and efficiency.

B. Sending and Receiving Emergency Request Process During Interoperation of Services

In the process of collaborative message sending and receiving during the interoperation of smart services, we consider the case where for collaborative interoperability, service-level

TABLE IV
PERFORMANCE MATRIX DURING SERVICE-LEVEL AGREEMENT WITH INCREASED NUMBER OF REQUESTS.

Number of requests	Throughput (per second)	Contract execution (ms)	Receive request delay (ms)
100	0.41	1.66	2.66
500	3.63	1.55	2.33
1000	5.25	1.29	1.67
2000	8.29	1.21	1.33
5000	6.27	2.68	2.66

TABLE V
PERFORMANCE MATRIX DURING SERVICE-LEVEL AGREEMENT WITH INCREASED NUMBER OF NODES.

Number of nodes	Throughput (per second)	Contract execution (ms)	Receive request delay (ms)
100	0.21	1.82	2.98
500	1.63	1.68	2.5
1000	3.65	1.32	2.0
2000	6.95	2.1	2.2
5000	3.2	2.3	2.8

agreement is already shared between interoperable services. Figure 13 shows the pictorial steps of sending and receiving an alert message between interoperable services. Table VI shows the performance matrix of the proposed security framework through increasing the number of collaborative alert request messages between interoperable services We noticed that the performance of the proposed framework is improved from the first two experiments in which the involvement of a global adaptive engine adds some time complexity to the running performance of the proposed framework. The execution time of a smart contract is much more improved due to less smart contract execution as compared to the first two experiments. From Table VII we find out that the performance matrix

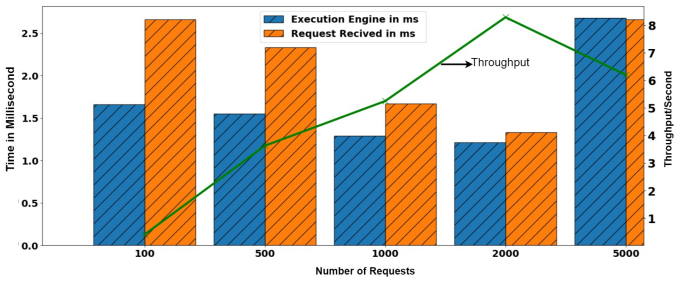

Fig. 11. Performance result with the increased number of requests.

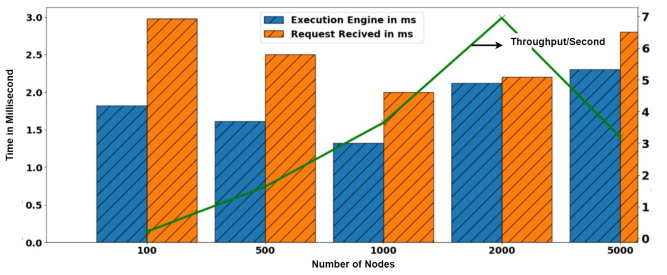

Fig. 12. Performance matrix during service-level agreement with increased number of nodes.

of the particular process is decreased. Figure 14 shows the graphical result of system performance when increasing the number of IoT nodes in smart services for interoperability. We also noticed that in both results of this particular process, throughput is decreased when we increase the number of IoT nodes along with an increased number of collaborative requests at maximum numbers

TABLE VI
PERFORMANCE MATRIX OF SENDING AND RECEIVING EMERGENCY REQUEST PROCESS DURING INTEROPERATION OF SERVICES WITH AN INCREASED NUMBER OF REQUESTS

Number of requests	Throughput (per second)	Contract execution (ms)	Receive request delay (ms)
100	1.65	0.98	1.68
500	7.12	0.86	0.98
1000	13.23	0.76	0.82
2000	26.53	0.72	0.79
5000	22.77	1.4	1.56

TABLE VII
PERFORMANCE MATRIX OF SENDING AND RECEIVING EMERGENCY REQUEST PROCESS DURING INTEROPERATION OF SERVICES WITH AN INCREASED NUMBER OF IOT NODES

Number of nodes	Throughput (per second)	Contract execution (ms)	Receive request delay (ms)
100	0.49	1.69	1.91
500	6.12	0.92	1.4
1000	11.21	0.82	1.31
2000	20.53	0.93	1.1
5000	21.77	0.92	2.5

C. End-End Message Sending and Receiving Process

In the context of collaborative message sending and receiving, we examine the scenario where interoperable services engage in bidirectional communication by sending and receiving collaborative messages. This specific use case emphasizes the significance of our proposed security framework in handling substantial workloads during the interoperation of smart services. Figure 16 shows the workflow of end-to-end sending and receiving alert messages during the interoperation of smart services. The workflow of the specific use case follows the following steps

1) Smart service-A prompted the alert message condition and the application agent generates the message which will broadcast to the connected services through the SDN controller by providing the required security requirement to the generated message.
2) The local adaptive agent is responsible to take the message from the SDN controller and performed security verification in order to verifiy the legitimacy of received message.
3) The security verification is performed with the help of the Local blockchain in order to fetch the validation attribute from the local blockchain and compare it with the received message.
4) After security verification the decrypted message is forwarded to the execution engine.

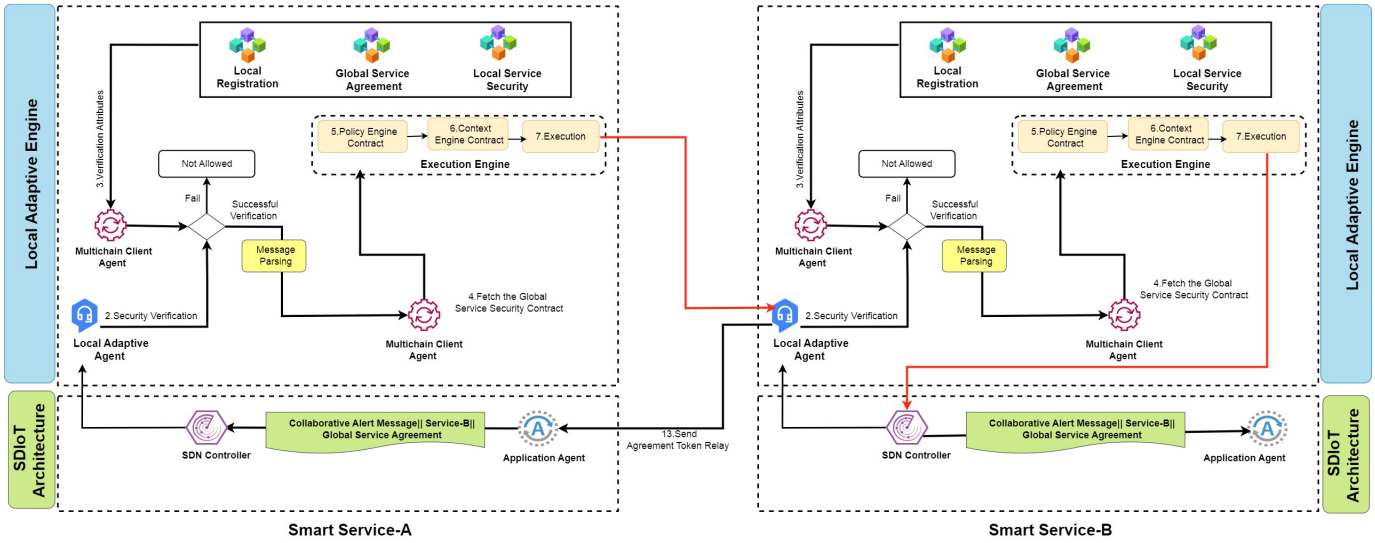

Fig. 13. Collaborative alert sending and receiving message process.

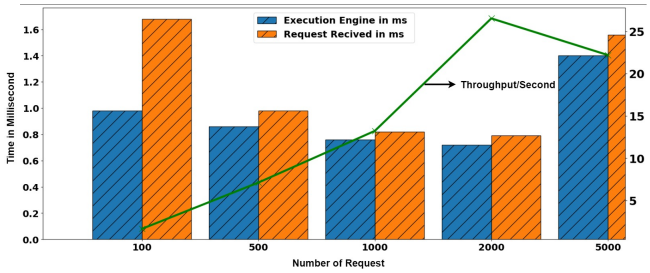

Fig. 14. Performance matrix of sending and receiving emergency request process with an increased number of requests.

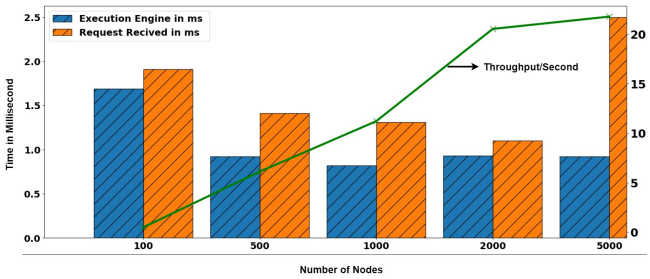

Fig. 15. Performance matrix of sending and receiving emergency request process with increased number of nodes.

5) At the execution engine first the Policy engine fetches the global security contract from the Local blockchain with the help of the service agreement transaction ID and converts it into JSON format in order to execute the policy in the context engine.
6) In the context engine, the authenticity of the global security verification is performed then the trust value is associated with the message and forwarded to the execution process.
7) In the execution process, The threshold of trust value in the alert message is continuously verified in order to forward it to the other service-B.
8) In end to end communication, service-B will perform the same process from Step-1 to Step-7 in response to the request message in order to provide an update to the requester service.

The performance of the specific operation deteriorated compared to the previous experiment, as observed from Table VIII. This decline can be attributed to increased computation workload in the End-to-End interaction, specifically the cryptographic encryption and decryption processes. Figure VIII provides a graphical representation of the performance matrix as the number of requests increases. Additionally, Table IX presents a tabular overview of the performance matrix as the number of nodes in the services increases. Notably, the throughput of the process decreased due to the increased number of nodes in the SDIoT architecture. Figure 19 visually depicts the performance matrix of the security framework when the number of requests is incrementally increased.

TABLE VIII
PERFORMANCE MATIX END-TO-END MESSAGE SENDING AND RECEIVING PROCESS WITH INCREASED NUMBER OF REQUEST

Number of requests	Throughput per second	Contract execution (ms)	Request response delay (ms)
100	1.35	1.28	1.91
500	5.12	1.12	2.12
1000	7.23	0.86	1.62
2000	13.53	1.26	2.12
5000	11.77	1.4	2.3

TABLE IX
PERFORMANCE MATIX END-TO-END MESSAGE SENDING AND RECEIVING PROCESS WITH INCREASED NUMBER OF IOT NODES

Number of nodes	Throughput per second	Contract execution (ms)	Response delay (ms)
100	0.21	1.72	2.5
500	3.12	1.52	2.3
1000	5.21	1.13	1.8
2000	7.53	1.53	2.3
5000	4.77	1.92	2.5

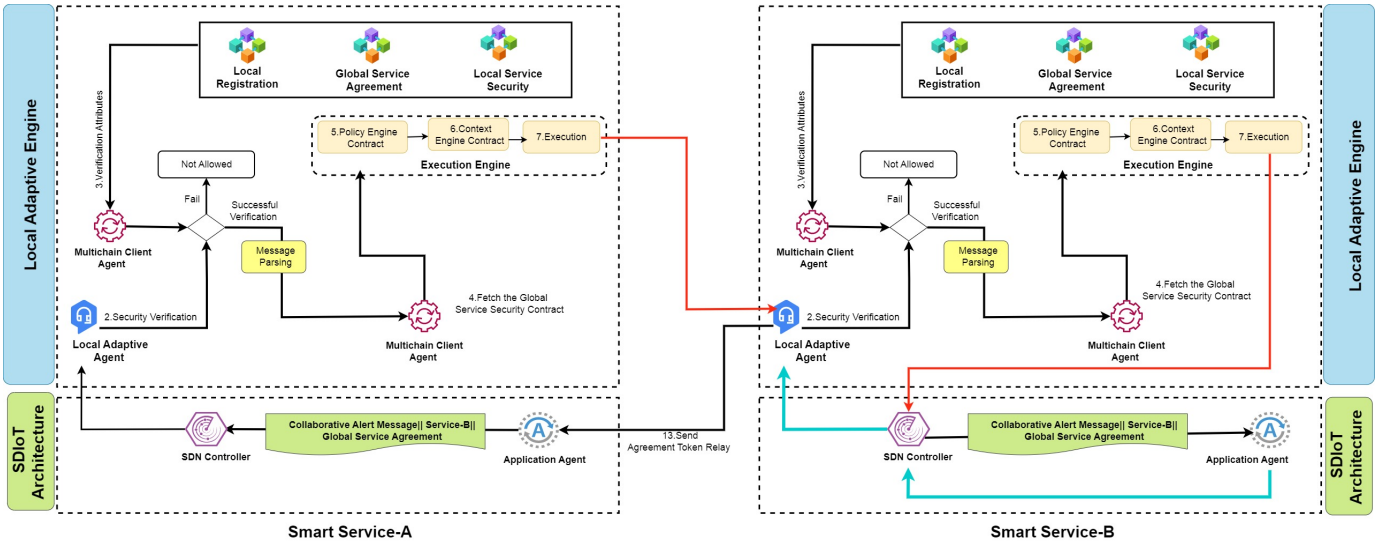

Fig. 16. End-to-End alert message workflow.

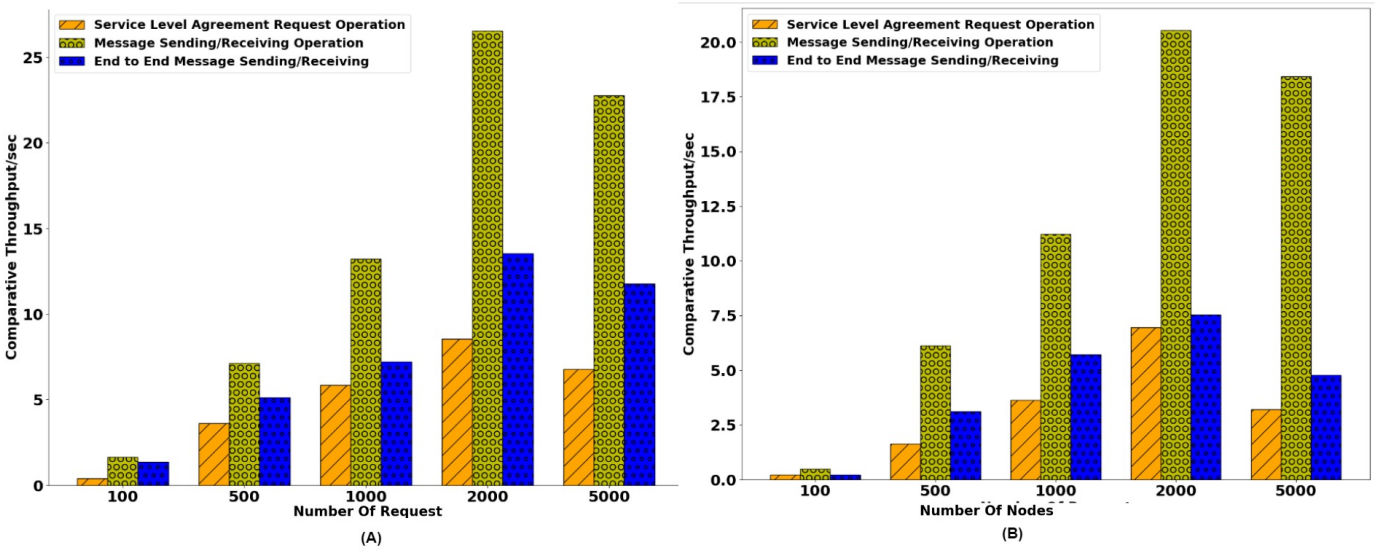

Fig. 17. Comparative system performance result with (A) increased number of requests and (B) increased number of nodes.

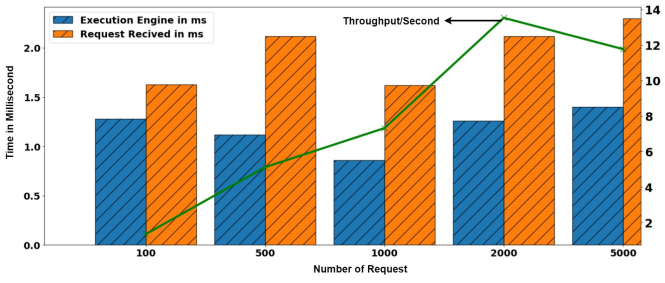

Fig. 18. Performance Matrix of End-End message sending and receiving with the increased number of requests.

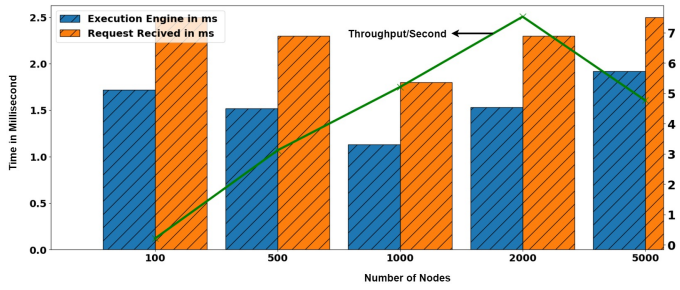

Fig. 19. Performance result with the increased number of nodes.

VIII. Comparative System Performance of the Proposed Security Framework

The system performance of our proposed security framework in the collaborative disaster management and response use cases in smart cities is evaluated based on the comparative throughput of all processes during the interoperation of smart services, as depicted in Figure 17. It is observed that the throughput during the process of fetching the service-level agreement is initially low due to the involvement of the global security adaptive engine with the local adaptive security

engine. However, once the service-level agreement is obtained, the throughput of all processes demonstrates improved results. This finding highlights the effectiveness of our proposed security framework, particularly in supporting delay-sensitive applications. In the SDIoT architecture, the memory pool has a significant impact on throughput compared to the memory pool of the "multi-chain" blockchain. When we increase the number of nodes in the SDIoT architecture while keeping the request messages constant, the throughput of the security framework decreases. However, the opposite happens when we increase the number of request messages while keeping the number of IoT nodes constant.

TABLE X
TRUST CONVERGENCE BY CHANGING THE NUMBER OF REQUESTS

Number of requests	Trust convergence/sec ECC(128 bit)	Trust convergence/sec ECC(192 bit)	Trust convergence/sec ECC(256 bit)
100	0.0035	0.0028	0.001
500	0.0052	0.0031	0.0026
1000	0.0072	0.0042	0.0028
2000	0.0091	0.006	0.004
5000	0.0062	0.003	0.002

TABLE XI
TRUST CONVERGENCE BY CHANGING THE NUMBER OF NODES

Number of requests	Trust convergence/sec ECC(128 bit)	Trust convergence/sec ECC(192 bit)	Trust convergence/sec ECC(256 bit)
100	0.0025	0.0019	0.0001
500	0.0045	0.0022	0.0015
1000	0.0062	0.0042	0.0018
2000	0.0071	0.0062	0.0041
5000	0.003	0.002	0.0022

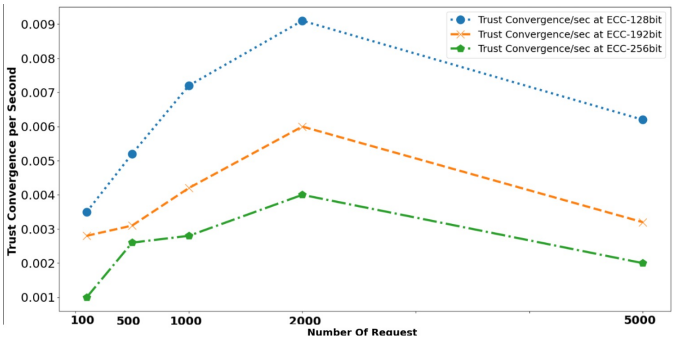

Fig. 20. Trust convergence by changing the number of requests.

IX. ADAPTIVENESS SECURITY ASSESSMENT

In order to evaluate the effective adaptability of smart services with different security policies during interoperation, we conduct adaptive security assessment experiments. The experiment aims to measure the speed and effectiveness of the service's adaptive capabilities. We created three syntactic security policies for each service that needs to be agreed upon during the interoperation of smart services. We utilized programmable trust conditions and ECC cryptographic suites of varying key lengths (128, 192, and 256) for authentication

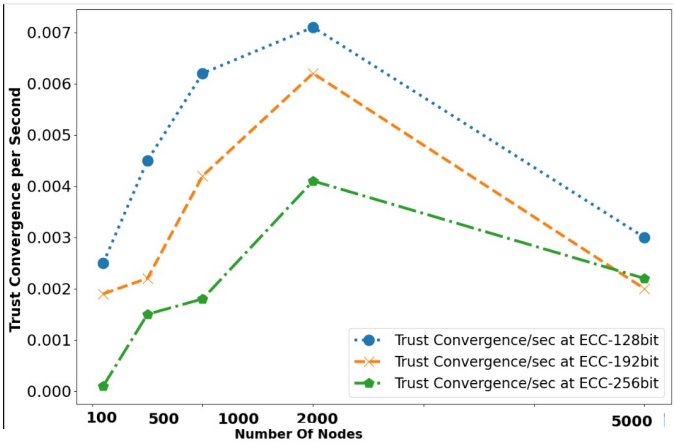

Fig. 21. Performance result with the increased number of nodes.

and trust binding to highlight the adaptiveness of the security framework for interoperable services involved in disaster management services. We set a common programmable collaborative service trust value of 0.003 for each interoperable service, which means each interoperable service is able to send and receive a message when its collaborative service trust attribute is converged up to the threshold value. Table X shows the result of trust convergence when the number of the request message is increased, we noticed the pattern of slow trust convergence per second during the smart service interoperations of the third process when authentication requirements change in terms of ECC cryptographic keys (128,192,256) bits. Figure 20 shows the graphical representation of trust convergence when the number of the request message is increased. We also noticed the same pattern in Table XI with more slow trust convergence per second as compared with the previous result that reflects the importance of value programmable value of the trust. Figure 21 shows the graphical representation of trust convergence when the number of IoT nodes is increased.

X. CONCLUSION

Smart cities strive to improve people's quality of life by integrating various aspects of urban life through interconnected smart services. To build new housing complexes, urban areas are already embracing the notion of smart cities. To accomplish this aim, they are deploying a variety of linked smart services, such as smart transportation systems, water management, waste management, E-governance, etc. However, this integration poses significant security risks, especially when it comes to collaborative services provided by multiple connected smart services. One key challenge is the need for adaptive security policies to ensure interoperability between these services. Our main motive for this article is to address these security issues through our proposed security framework. We present a use case focusing on emergency response in a smart city, where multiple services with different security policies exchange request and response messages for collaborative emergency tasks, exposing security vulnerabilities. Furthermore, we explore how smart contract-based systems can provide adaptive security in a unique manner. Our evaluation results prove that the proposed framework is scalable in

terms of multiple smart services integration in a smart city and also adaptive in nature through the interoperability of diverse security policies between smart services ensuring that the security measures could adapt to changing security threats, without impacting the functioning of the individual services. In our research, we explore the utilization of smart contract-based systems to provide adaptive security in a distinctive manner. By leveraging smart contracts, we can achieve adaptive security, where security policies and measures can dynamically adjust to changing conditions and emerging threats. Furthermore, our evaluation indicates that the SDN (Software-Defined Networking) controller and the Blockchain memory pool are crucial components in ensuring scalability of the proposed security framework. The SDN controller allows for centralized management and control of the network, facilitating efficient communication and coordination between smart services. The Blockchain memory pool, on the other hand, provides a distributed and tamper-proof ledger that ensures the integrity and immutability of security-related transactions and data. Looking ahead, we recommend the integration of a privacy management module during the interoperation of smart services as a future direction. By incorporating a privacy management module, we can establish robust mechanisms to protect user privacy, ensure data confidentiality, and comply with privacy regulations.

REFERENCES

[1] A. Kirimtat, O. Krejcar, A. Kertesz, and M. F. Tasgetiren, "Future trends and current state of smart city concepts: A survey," *IEEE Access*, vol. 8, pp. 86 448–86 467, 2020.

[2] P. Arthurs, L. Gillam, P. Krause, N. Wang, K. Halder, and A. Mouzakitis, "A taxonomy and survey of edge cloud computing for intelligent transportation systems and connected vehicles," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[3] M. M. Rathore, A. Paul, S. Rho, M. Khan, S. Vimal, and S. A. Shah, "Smart traffic control: Identifying driving-violations using fog devices with vehicular cameras in smart cities," *Sustainable Cities and Society*, vol. 71, p. 102986, 2021.

[4] M. M. Rathore, S. Attique Shah, A. Awad, D. Shukla, S. Vimal, and A. Paul, "A cyber-physical system and graph-based approach for transportation management in smart cities," *Sustainability*, vol. 13, no. 14, p. 7606, 2021.

[5] G. Viale Pereira, M. A. Cunha, T. J. Lampoltshammer, P. Parycek, and M. G. Testa, "Increasing collaboration and participation in smart city governance: A cross-case analysis of smart city initiatives," *Information Technology for Development*, vol. 23, no. 3, pp. 526–553, 2017.

[6] O. Bello and S. Zeadally, "Toward efficient smartification of the Internet of Things (IoT) services," *Future Generation Computer Systems*, vol. 92, pp. 663–673, 2019.

[7] S. L. Keoh, S. S. Kumar, and H. Tschofenig, "Securing the Internet of Things: A standardization perspective," *IEEE Internet of Things Journal*, vol. 1, no. 3, pp. 265–275, 2014.

[8] S. Singh, P. K. Sharma, B. Yoon, M. Shojafar, G. H. Cho, and I.-H. Ra, "Convergence of blockchain and artificial intelligence in IoT network for the sustainable smart city," *Sustainable Cities and Society*, vol. 63, p. 102364, 2020.

[9] M. J. Islam, A. Rahman, S. Kabir, M. R. Karim, U. K. Acharjee, M. K. Nasir, S. S. Band, M. Sookhak, and S. Wu, "Blockchain-sdn-based energy-aware and distributed secure architecture for iot in smart cities," *IEEE Internet of Things Journal*, vol. 9, no. 5, pp. 3850–3864, 2021.

[10] D. Mingxiao, M. Xiaofeng, Z. Zhe, W. Xiangwei, and C. Qijun, "A review on consensus algorithm of blockchain," in *2017 IEEE international conference on systems, man, and cybernetics (SMC)*. IEEE, 2017, pp. 2567–2572.

[11] B. Bhushan, C. Sahoo, P. Sinha, and A. Khamparia, "Unification of blockchain and Internet of Things (BIoT): requirements, working model, challenges and future directions," *Wireless Networks*, vol. 27, pp. 55–90, 2021.

[12] I. Makhdoom, I. Zhou, M. Abolhasan, J. Lipman, and W. Ni, "Privysharing: A blockchain-based framework for privacy-preserving and secure data sharing in smart cities," *Computers & Security*, vol. 88, p. 101653, 2020.

[13] H. Mora, J. C. Mendoza-Tello, E. G. Varela-Guzmán, and J. Szymanski, "Blockchain technologies to address smart city and society challenges," *Computers in Human Behavior*, vol. 122, p. 106854, 2021.

[14] D. C. Nguyen, P. N. Pathirana, M. Ding, and A. Seneviratne, "Blockchain for 5G and beyond networks: A state of the art survey," *Journal of Network and Computer Applications*, vol. 166, p. 102693, 2020.

[15] J. Xie, H. Tang, T. Huang, F. R. Yu, R. Xie, J. Liu, and Y. Liu, "A survey of blockchain technology applied to smart cities: Research issues and challenges," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2794–2830, 2019.

[16] L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo, "SDN-WISE: Design, prototyping and experimentation of a stateful SDN solution for wireless sensor networks," in *2015 IEEE conference on computer communications (INFOCOM)*. IEEE, 2015, pp. 513–521.

[17] F. Bannour, S. Souihi, and A. Mellouk, "Distributed SDN control: Survey, taxonomy, and challenges," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, pp. 333–354, 2017.

[18] P. K. Sharma, S. Singh, Y.-S. Jeong, and J. H. Park, "Distblocknet: A distributed blockchains-based secure SDN architecture for IoT networks," *IEEE Communications Magazine*, vol. 55, no. 9, pp. 78–85, 2017.

[19] J. Zhou, H. Jiang, J. Wu, L. Wu, C. Zhu, and W. Li, "SDN-based application framework for wireless sensor and actor networks," *IEEE Access*, vol. 4, pp. 1583–1594, 2016.

[20] T. Ali, M. Irfan, A. S. Alwadie, and A. Glowacz, "IoT-based smart waste bin monitoring and municipal solid waste management system for smart cities," *Arabian Journal for Science and Engineering*, vol. 45, no. 12, pp. 10 185–10 198, 2020.

[21] A. Yazdinejad, R. M. Parizi, A. Dehghantanha, Q. Zhang, and K.-K. R. Choo, "An energy-efficient SDN controller architecture for IoT networks with blockchain-based security," *IEEE Transactions on Services Computing*, vol. 13, no. 4, pp. 625–638, 2020.

[22] G. S. Aujla, M. Singh, A. Bose, N. Kumar, G. Han, and R. Buyya, "BlockSDN: Blockchain-as-a-service for software defined networking in smart city applications," *IEEE Network*, vol. 34, no. 2, pp. 83–91, 2020.

[23] S. A. Latif, F. B. X. Wen, C. Iwendi, F. W. Li-li, S. M. Mohsin, Z. Han, and S. S. Band, "Ai-empowered, blockchain and SDN integrated security architecture for IoT network of cyber physical systems," *Computer Communications*, vol. 181, pp. 274–283, 2022.

[24] W. Iqbal, H. Abbas, M. Daneshmand, B. Rauf, and Y. A. Bangash, "An in-depth analysis of IoT security requirements, challenges, and their countermeasures via software-defined security," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 10 250–10 276, 2020.

[25] E. Ismagilova, L. Hughes, N. P. Rana, and Y. K. Dwivedi, "Security, privacy and risks within smart cities: Literature review and development of a smart city interaction framework," *Information Systems Frontiers*, vol. 24, no. 2, pp. 393–414, 2022.

[26] S. H. Alsamhi, O. Ma, M. S. Ansari, and F. A. Almalki, "Survey on collaborative smart drones and Internet of Things for improving smartness of smart cities," *IEEE Access*, vol. 7, pp. 128 125–128 152, 2019.

[27] B. Tang, H. Kang, J. Fan, Q. Li, and R. Sandhu, "IoT passport: A blockchain-based trust framework for collaborative internet-of-things," in *Proceedings of the 24th ACM symposium on access control models and technologies*, 2019, pp. 83–92.

[28] A. O. Khadidos, S. Shitharth, H. Manoharan, A. Yafoz, A. O. Khadidos, and K. H. Alyoubi, "An intelligent security framework based on collaborative mutual authentication model for smart city networks," *IEEE Access*, vol. 10, pp. 85 289–85 304, 2022.

[29] S. Siddiqui, S. Hameed, S. A. Shah, A. K. Khan, and A. Aneiba, "Smart contract-based security architecture for collaborative services in municipal smart cities," *Journal of Systems Architecture*, vol. 135, p. 102802, 2023.

[30] I. P. Žarko, S. Mueller, M. Płociennik, T. Rajtar, M. Jacoby, M. Pardi, G. Insolvibile, V. Glykantzis, A. Antonić, M. Kušek, *et al.*, "The symbIoTe solution for semantic and syntactic interoperability of cloud-based IoT platforms," in *2019 Global IoT Summit (GIoTS)*. IEEE, 2019, pp. 1–6.

[31] M. Asif, Z. Aziz, M. Bin Ahmad, A. Khalid, H. A. Waris, and A. Gilani, "Blockchain-based authentication and trust management mechanism for smart cities," *Sensors*, vol. 22, no. 7, p. 2604, 2022.

[32] B. Bhushan, A. Khamparia, K. M. Sagayam, S. K. Sharma, M. A. Ahad, and N. C. Debnath, "Blockchain for smart cities: A review of architectures, integration trends and future research directions," *Sustainable Cities and Society*, vol. 61, p. 102360, 2020.

[33] S. Hameed, S. A. Shah, Q. S. Saeed, S. Siddiqui, I. Ali, A. Vedeshin, and D. Draheim, "A scalable key and trust management solution for IoT sensors using SDN and blockchain technology," *IEEE Sensors Journal*, vol. 21, no. 6, pp. 8716–8733, 2021.

[34] J. Sun, J. Yan, and K. Z. Zhang, "Blockchain-based sharing services: What blockchain technology can contribute to smart cities," *Financial Innovation*, vol. 2, no. 1, pp. 1–9, 2016.

[35] A. Rahman, A. Montieri, D. Kundu, M. Karim, M. Islam, S. Umme, A. Nascita, A. Pescapé, *et al.*, "On the integration of blockchain and SDN: Overview, applications, and future perspectives," *Journal of Network and Systems Management*, vol. 30, no. 4, pp. 1–44, 2022.

[36] C. Tselios, I. Politis, and S. Kotsopoulos, "Enhancing SDN security for IoT-related deployments through blockchain," in *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. IEEE, 2017, pp. 303–308.

[37] D. V. Medhane, A. K. Sangaiah, M. S. Hossain, G. Muhammad, and J. Wang, "Blockchain-enabled distributed security framework for next-generation IoT: An edge cloud and software-defined network-integrated approach," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6143–6149, 2020.

[38] O. Salman, I. H. Elhajj, A. Kayssi, and A. Chehab, "SDN controllers: A comparative study," in *2016 18th mediterranean electrotechnical conference (MELECON)*. IEEE, 2016, pp. 1–6.

[39] F. Ullah, J. Wang, M. Farhan, S. Jabbar, M. K. Naseer, and M. Asif, "Lsa based smart assessment methodology for SDN infrastructure in IoT environment," *International Journal of Parallel Programming*, vol. 48, pp. 162–177, 2020.

[40] S. Z. Marshoodulla and G. Saha, "Data heterogeneity handling in SDN-based IoT infrastructure," *NeuroQuantology*, vol. 20, no. 14, pp. 805–812, 2022.

[41] S. Benkhaled, M. Hemam, and M. Maimour, "SDN-based approaches for heterogeneity and interoperability in Internet of Things: An overview," *Distributed Sensing and Intelligent Systems: Proceedings of ICDSIS 2020*, pp. 489–499, 2022.

[42] G. Ali, N. Ahmad, Y. Cao, S. Khan, H. Cruickshank, E. A. Qazi, and A. Ali, "xdbauth: Blockchain based cross domain authentication and authorization framework for Internet of Things," *IEEE Access*, vol. 8, pp. 58 800–58 816, 2020.

[43] G. Wang, Q. Wang, and S. Chen, "Exploring blockchains interoperability: A systematic survey," *ACM Computing Surveys*, 2023.

[44] E. R. D. Villarreal, J. García-Alonso, E. Moguel, and J. A. H. Alegría, "Blockchain for healthcare management systems: A survey on interoperability and security," *IEEE Access*, vol. 11, pp. 5629–5652, 2023.

[45] K. Lin, J. Gao, G. Han, H. Wang, and C. Li, "Intelligent blockchain-enabled adaptive collaborative resource scheduling in large-scale industrial Internet of Things," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 12, pp. 9196–9205, 2022.

[46] H. D. Zubaydi, P. Varga, and S. Molnár, "Leveraging blockchain technology for ensuring security and privacy aspects in Internet of Things: A systematic literature review," *Sensors*, vol. 23, no. 2, p. 788, 2023.

[47] J. Koo and Y.-G. Kim, "Interoperability requirements for a smart city," in *Proceedings of the 36th Annual ACM Symposium on Applied Computing*, 2021, pp. 690–698.

[48] A. Imteaj, A. R. Shahid, and S. Zaman, "Leveraging blockchain interoperability for interdependent networks," *IEEE Consumer Electronics Magazine*, 2023.

[49] J. Huang, D. Fang, Y. Qian, and R. Q. Hu, "Recent advances and challenges in security and privacy for v2x communications," *IEEE Open Journal of Vehicular Technology*, vol. 1, pp. 244–266, 2020.

[50] M. Alsaeedi, M. M. Mohamad, and A. A. Al-Roubaiey, "Toward adaptive and scalable openflow-SDN flow control: A survey," *IEEE Access*, vol. 7, pp. 107 346–107 379, 2019.

[51] G. S. Aujla, A. Singh, M. Singh, S. Sharma, N. Kumar, and K.-K. R. Choo, "Blocked: Blockchain-based secure data processing framework in edge envisioned v2x environment," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 6, pp. 5850–5863, 2020.

[52] A. P. Balcerzak, E. Nica, E. Rogalska, M. Poliak, T. Klieštik, and O.-M. Sabie, "Blockchain technology and smart contracts in decentralized governance systems," *Administrative Sciences*, vol. 12, no. 3, p. 96, 2022.

[53] G. V. Pereira, P. Parycek, E. Falco, and R. Kleinhans, "Smart governance in the context of smart cities: A literature review," *Information Polity*, vol. 23, no. 2, pp. 143–162, 2018.

[54] A. Meijer and M. P. R. Bolívar, "Governing the smart city: a review of the literature on smart urban governance," *International review of administrative sciences*, vol. 82, no. 2, pp. 392–408, 2016.

[55] G. Rathee, A. Kumar, C. A. Kerrache, and R. Iqbal, "A trust-based mechanism for drones in smart cities," *IET Smart Cities*, 2022.

[56] P. Antonios, K. Konstantinos, and G. Christos, "A systematic review on semantic interoperability in the ioe-enabled smart cities," *Internet of Things*, p. 100754, 2023.

[57] T. K. Hui, R. S. Sherratt, and D. D. Sánchez, "Major requirements for building smart homes in smart cities based on internet of things technologies," *Future Generation Computer Systems*, vol. 76, pp. 358–369, 2017.

[58] P. M. Rao and B. Deebak, "Security and privacy issues in smart cities/industries: Technologies, applications, and challenges," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–37, 2022.

[59] R. S. P. Maciel, J. M. N. David, D. Claro, and R. Braga, "Full interoperability: Challenges and opportunities for future information systems," *Sociedade Brasileira de Computação*, 2017.

[60] R. Sánchez-Corcuera, A. Nuñez-Marcos, J. Sesma-Solance, A. Bilbao-Jayo, R. Mulero, U. Zulaika, G. Azkune, and A. Almeida, "Smart cities survey: Technologies, application domains and challenges for the cities of the future," *International Journal of Distributed Sensor Networks*, vol. 15, no. 6, p. 1550147719853984, 2019.

[61] E. Zadobrischi and M. Dimian, "Vehicular communications utility in road safety applications: a step toward self-aware intelligent traffic systems," *Symmetry*, vol. 13, no. 3, p. 438, 2021.

[62] J. Koo and Y.-G. Kim, "Resource identifier interoperability among heterogeneous IoT platforms," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 7, pp. 4191–4208, 2022.

[63] M. Ibrar, L. Wang, N. Shah, O. Rottenstreich, G.-M. Muntean, and A. Akbar, "Reliability-aware flow distribution algorithm in SDN-enabled fog computing for smart cities," *IEEE Transactions on Vehicular Technology*, 2022.

[64] P. Agbaje, A. Anjum, A. Mitra, E. Oseghale, G. Bloom, and H. Olufowobi, "Survey of interoperability challenges in the internet of vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 12, pp. 22 838–22 861, 2022.

[65] M. Tosic, F. A. Coelho, B. Nouwt, D. E. Rua, A. Tomcic, and S. Pesic, "Towards a cross-domain semantically interoperable ecosystem," in *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, 2022, pp. 1640–1641.

[66] M. Msahli, H. Labiod, and G. Ampt, "Security interoperability for cooperative its: Architecture and validation," in *2019 10th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*. IEEE, 2019, pp. 1–6.

[67] M. S. Rahman, M. Chamikara, I. Khalil, and A. Bouras, "Blockchain-of-blockchains: An interoperable blockchain platform for ensuring IoT data integrity in smart city," *Journal of Industrial Information Integration*, vol. 30, p. 100408, 2022.

[68] S. Karumba, R. Jurdak, S. Kanhere, and S. Sethuvenkatraman, "Bailif: A blockchain agnostic interoperability framework," in *Proceedings of the 5th IEEE International Conference on Blockchain and Cryptocurrency, Dubai, UAE, May, 2023*. Institute of Electrical and Electronics Engineers Inc., 2023.

[69] I. Guvenc, F. Koohifar, S. Singh, M. L. Sichitiu, and D. Matolak, "Detection, tracking, and interdiction for amateur drones," *IEEE Communications Magazine*, vol. 56, no. 4, pp. 75–81, 2018.

[70] N. H. Motlagh, T. Taleb, and O. Arouk, "Low-altitude unmanned aerial vehicles-based Internet of Things services: Comprehensive survey and future perspectives," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 899–922, 2016.

[71] H. Basheer and M. Itani, "Zero touch in fog, IoT, and manet for enhanced smart city applications: A survey," *Future Cities and Environment*, vol. 9, no. 1, p. 5, 2023.

[72] X. Chen, Y. Deng, H. Ding, G. Qu, H. Zhang, P. Li, and Y. Fang, "Vehicle as a service (vaas): Leverage vehicles to build service networks and capabilities for smart cities," *arXiv preprint arXiv:2304.11397*, 2023.

[73] R. M. Al Batayneh, N. Taleb, R. A. Said, M. T. Alshurideh, T. M. Ghazal, and H. M. Alzoubi, "It governance framework and smart services integration for future development of dubai infrastructure utilizing ai and big data, its reflection on the citizens standard of living," in *Proceedings of the International Conference on Artificial Intelligence and Computer Vision (AICV2021)*. Springer, 2021, pp. 235–247.

[74] A. Dua, N. Kumar, A. K. Das, and W. Susilo, "Secure message communication protocol among vehicles in smart city," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 5, pp. 4359–4373, 2017.

[75] F. A. Reegu, H. Abas, A. Jabbari, R. Akmam, M. Uddin, C.-M. Wu, C. Chin-Ling, and O. I. Khalaf, "Interoperability requirements for blockchain-enabled electronic health records in healthcare: A systematic review and open research challenges," *Security and Communication Networks*, vol. 2022, 2022.

[76] M. Sookhak, H. Tang, Y. He, and F. R. Yu, "Security and privacy of smart cities: a survey, research issues and challenges," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1718–1743, 2018.

[77] S. Kharche and P. Dere, "Interoperability issues and challenges in 6g networks," *Journal of Mobile Multimedia*, vol. 18, no. 5, pp. 1445–1470, 2022.

[78] Y. Alshboul, A. A. R. Bsoul, M. Al Zamil, and S. Samarah, "Cybersecurity of smart home systems: Sensor identity protection," *Journal of Network and Systems Management*, vol. 29, no. 3, pp. 1–27, 2021.

[79] P. Bellavista, C. Esposito, L. Foschini, C. Giannelli, N. Mazzocca, and R. Montanari, "Interoperable blockchains for highly-integrated supply chains in collaborative manufacturing," *Sensors*, vol. 21, no. 15, p. 4955, 2021.

[80] S. A. Knowles Flanagan, "Cooperative connected intelligent vehicles and infrastructure for road safety applications," Ph.D. dissertation, Aston University, 2022.

[81] R. Xu, Y. Chen, E. Blasch, and G. Chen, "Blendcac: A smart contract enabled decentralized capability-based access control mechanism for the IoT," *Computers*, vol. 7, no. 3, p. 39, 2018.

[82] S. M. M. Gilani, M. Usman, S. Daud, A. Kabir, Q. Nawaz, and O. Judit, "SDN-based multi-level framework for smart home services," *Multimedia Tools and Applications*, pp. 1–21, 2023.

[83] B. Rana and Y. Singh, "Interoperable agile IoT," *Agile Software Development: Trends, Challenges and Applications*, pp. 51–70, 2023.

[84] S. Shamsudheen, G. Karthik, A. Anoop, and P. Gobinathan, "Internet-of-things in emergency services: Architecture, applications, and research challenges," in *2023 1st International Conference on Advanced Innovations in Smart Cities (ICAISC)*. IEEE, 2023, pp. 1–6.

[85] M. A. Abid, N. Afaqui, M. A. Khan, M. W. Akhtar, A. W. Malik, A. Munir, J. Ahmad, and B. Shabir, "Evolution towards smart and software-defined Internet of Things," *AI*, vol. 3, no. 1, pp. 100–123, 2022.

[86] S. Banerjee, B. Bera, A. K. Das, S. Chattopadhyay, M. K. Khan, and J. J. Rodrigues, "Private blockchain-envisioned multi-authority cp-abe-based user access control scheme in IIoT," *Computer Communications*, vol. 169, pp. 99–113, 2021.

[87] S. KOZHEVNIKOV, M. SVITEK, and P. SKOBELEV, "Smart grid system for real-time adaptive utility management in smart cities," in *IMCIC 2022-13th International Multi-Conference on Complexity, Informatics and Cybernetics, Proceedings*, 2022, pp. 4–9.

[88] A. Buldas, D. Draheim, M. Gault, R. Laanoja, T. Nagumo, M. Saarepera, S. A. Shah, J. Simm, J. Steiner, T. Tammet, and A. Truu, "An ultra-scalable blockchain platform for universal asset tokenization: Design and implementation," *IEEE Access*, vol. 10, pp. 77 284–77 322, 2022.

[89] https://alphabill.org/, [last accessed: 07 June 2023].

[90] https://polkadot.network/, [last accessed: 07 June 2023].

[91] G. Wood, "Polkadot: Vision for a heterogeneous multi-chain framework, Draft 1," 2016, [last accessed: 3 Feb 2022] https://polkadot.network/PolkaDotPaper.pdf.

[92] https://www.multichain.com/, [last accessed: 07 June 2023].

[93] https://github.com/MultiChain/multichain-api-libraries/, [last accessed: 07 June 2023].

[94] L. EL-Garoui, S. Pierre, and S. Chamberland, "A new SDN-based routing protocol for improving delay in smart city environments," *Smart Cities*, vol. 3, no. 3, pp. 1004–1021, 2020.

[95] Ł. Ogrodowczyk, B. Belter, and M. LeClerc, "IoT ecosystem over programmable SDN infrastructure for smart city applications," in *2016 Fifth European Workshop on Software-Defined Networks (EWSDN)*. IEEE, 2016, pp. 49–51.

[96] T. Li, J. Chen, and H. Fu, "Application scenarios based on SDN: an overview," in *Journal of Physics: Conference Series*, vol. 1187, no. 5. IOP Publishing, 2019, p. 052067.

[97] H. Mostafaei and M. Menth, "Software-defined wireless sensor networks: A survey," *Journal of Network and Computer Applications*, vol. 119, pp. 42–56, 2018.

[98] H. Mrabet, S. Belguith, A. Alhomoud, and A. Jemai, "A survey of IoT security based on a layered architecture of sensing and data analysis," *Sensors*, vol. 20, no. 13, p. 3625, 2020.

[99] T. Stackpole, "What is Web3?" *Harvard Business Review*, vol. 10 May, 2022, https://hbr.org/2022/05/what-is-web3.

[100] L. Jin and K. Parrott, "Web3 is our chance to make a better Internet," *Harvard Business Review*, vol. 10 May, 2022, https://hbr.org/2022/05/web3-is-our-chance-to-make-a-better-internet.

[101] J. Esber and S. D. Kominers, "Why build in Web3," *Harvard Business Review*, vol. 16 May, 2022, https://hbr.org/2022/05/why-build-in-web3.

[102] G. Edelman, "Paradise at the cryto arcade," *Wired*, vol. June, 2022.

[103] A. Buldas, D. Draheim, M. Gault, and M. Saarepera, "Towards a foundation of Web3," in *Proceedings of FDSE'2022 – the 9th International Conference on Future Data and Security Engineering*, ser. CCIS, vol. 1688. Berlin Heidelberg New York: Springer, 2022, pp. 3–18.

[104] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web – a new form of Web content that is meaningful to computers will unleash a revolution of new possibilities," *Scientific American*, vol. 17 May, 2001.

[105] https://www.w3.org/standards/semanticweb/, [last accessed: 07 June 2023].

[106] W. Zou, D. Lo, P. S. Kochhar, X.-B. D. Le, X. Xia, Y. Feng, Z. Chen, and B. Xu, "Smart contract development: Challenges and opportunities," *IEEE Transactions on Software Engineering*, vol. 47, no. 10, pp. 2084–2106, 2019.

[107] S. N. Khan, F. Loukil, C. Ghedira-Guegan, E. Benkhelifa, and A. Bani-Hani, "Blockchain smart contracts: Applications, challenges, and future trends," *Peer-to-peer Networking and Applications*, vol. 14, pp. 2901–2925, 2021.

[108] F. Bao, R. Chen, and J. Guo, "Scalable, adaptive and survivable trust management for community of interest based internet of things systems," in *2013 IEEE 11th International Symposium on Autonomous Decentralized Systems (ISADS)*. IEEE, 2013, pp. 1–7.

[109] S. A. Shah, D. Z. Seker, M. M. Rathore, S. Hameed, S. B. Yahia, and D. Draheim, "Towards disaster resilient smart cities: Can internet of things and big data analytics be the game changers?" *IEEE Access*, vol. 7, pp. 91 885–91 903, 2019.

[110] S. A. Shah, D. Z. Seker, S. Hameed, and D. Draheim, "The rising role of big data analytics and IoT in disaster management: recent advances, taxonomy and prospects," *IEEE Access*, vol. 7, pp. 54 595–54 614, 2019.

[111] C. Thomson, I. Romdhani, A. Al-Dubai, M. Qasem, B. Ghaleb, and I. Wadhaj, *Cooja Simulator Manual, Version 1.0*. Edinburgh Napier University, 2016, https://www.napier.ac.uk/~/media/worktribe/output-299955/cooja-simulator-manual.pdf [last accessed: 08 June 2023].

[112] G. Oikonomou, S. Duquennoy, A. Elsts, J. Eriksson, Y. Tanaka, and N. Tsiftes, "The contiki-ng open source operating system for next generation IoT devices," *SoftwareX*, vol. 18, p. 101089, 2022.

[113] Y. B. Zikria, M. K. Afzal, F. Ishmanov, S. W. Kim, and H. Yu, "A survey on routing protocols supported by the contiki Internet of Things operating system," *Future Generation Computer Systems*, vol. 82, pp. 200–219, 2018.

[114] C. Durmaz, M. Challenger, O. Dagdeviren, and G. Kardas, "Modelling contiki-based IoT systems," in *6th symposium on languages, applications and technologies (SLATE 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

Shahbaz Siddqui received MS Degree in Telecommunication from Hamdard University Karachi, Pakistan. He is pursuing his Ph.D. in Computer Sciences from the National University of Computer and Emerging Sciences, Karachi. He currently works as an Assistant professor at the Department of Computer Science at the National University of Computer and Emerging Sciences in Karachi Pakistan. His research interests include the Internet of Things, SDN, and blockchain.

Sufian Hameed received the Ph.D. degree in networks and information security from the University of Göttingen, Germany. He works as an Associate Professor at the Department of Computer Science at the National University of Computer and Emerging Sciences, Pakistan. He also leads the IT Security Labs at NUCES. The research lab studies and teaches security problems and solutions for different types of information and communication paradigms. His research area includes network security, web security, mobile security, and security architectures and protocols for Cloud and IoTs.

Syed Attique Shah received the Ph.D. degree from the Institute of Informatics, Istanbul Technical University, Istanbul, Turkey. During his Ph.D., he studied as a Visiting Scholar at the University of Tokyo, Japan, the National Chiao Tung University, Taiwan, and the Tallinn University of Technology, Estonia, where he completed the major content of his thesis. He has worked as an Associate Professor and the Chairperson at the Department of Computer Science, BUITEMS, Quetta, Pakistan. He was also engaged as a Lecturer at the Data Systems Group, Institute of Computer Science, University of Tartu, Estonia. Currently, he is working as a Lecturer in Smart Computer Systems, at the School of Computing and Digital Technology, Birmingham City University, United Kingdom. He is a Senior Member, IEEE. His research interests include big data analytics, the Internet of Things, machine learning, network security, and information management.

Dirk Draheim received the Ph.D. degree from Freie Universität Berlin and habilitation degree from Universität Mannheim, Germany. Currently, he is a full professor of information systems and the head of the information systems group at Tallinn University of Technology, Estonia. The information systems group conducts research in large and ultra-large-scale IT systems. He is also an initiator and a leader of numerous digital transformation initiatives.