Sustainable Cities and Society 113 (2024) 105717

Contents lists available at ScienceDirect

Sustainable Cities and Society

journal homepage: www.elsevier.com/locate/scs

A smart-contract-based adaptive security governance architecture for smart city service interoperations

Shahbaz Siddiqui [a], Sufian Hameed [a], Syed Attique Shah [b,*], Junaid Arshad [b], Yussuf Ahmed [b], Dirk Draheim [c]

[a] Department of Computer Science, National University of Computer & Engineering Sciences, 75160, Karachi, Pakistan
[b] School of Computing and Digital Technology, Birmingham City University, STEAMhouse, Birmingham, B4 7RQ, United Kingdom
[c] Tallinn University of Technology, Akadeemia tee 15A, Tallinn, 12169, Estonia

ARTICLE INFO

Keywords:
Smart city
Blockchain
Software-defined networking
Internet of Things
Multi-chain blockchains
MultiChain
Collaborative services
Interoperability
Security adaptation

ABSTRACT

Smart cities represent a promising paradigm aimed at enhancing citizens' quality of life through cutting-edge infrastructure and technological advancements. Collaborative services serve as a cornerstone for any smart city, fostering seamless cooperation among diverse entities, including government agencies, businesses, and individuals, thereby enhancing community outcomes. These services are pivotal, promoting seamless communication and collaboration among various smart applications, and facilitating data exchange, resource sharing, and functional interactions within smart city environments to optimize efficiency, effectiveness, and user experiences. However, the development and deployment of secure, interoperable services in smart cities present significant challenges. These issues encompass, ensuring data security compliance during interoperation, effective management of interconnected services, securely handling sensitive data across services, and addressing issues related to confidentiality, integrity, and availability (CIA) traits. To tackle these challenges, this research proposes an innovative adaptive security governance framework tailored for smart cities. This framework relies on dynamic security policies implemented through smart contracts to guarantee data security and privacy during smart service interoperation. Real-world use cases in collaborative smart city environments validate the framework, integrating multi-chain blockchain technology, smart services APIs, and Software-Defined Networking (SDN), showcasing its ability to enhance security and efficiency in collaborative services. This study contributes to the development of safe and efficient collaborative services inside smart cities, tackling administrative issues while emphasizing data security and privacy. Smart cities may improve citizens' living conditions while successfully addressing crucial security problems in an ever-changing environment by using this architecture.

1. 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 (Joo, 2023). At the core of a smart city infrastructure, smart services and applications are being developed with interoperability in mind, enabling seamless communication and collaboration among diverse systems and devices (Choi, 2022; Kirimtat, Krejcar, Kertesz, & Tasgetiren, 2020). 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 (Javed et al., 2022; Kumar, Singh, Gupta, & Madaan, 2020). 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 (Arthurs et al., 2021). 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 (Rathore, Attique Shah, et al., 2021; Rathore, Paul, et al., 2021). Furthermore, the example taken as a case study in this research, which includes the secure integration of interoperable transportation and weather services within smart cities can potentially provide accurate weather information to users of smart transport services. Fig. 1 illustrates a representative depiction of a smart city consisting of different 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

* Corresponding author.
E-mail address: syedattique.shah@bcu.ac.uk (S.A. Shah).

https://doi.org/10.1016/j.scs.2024.105717
Received 31 December 2023; Received in revised form 27 July 2024; Accepted 28 July 2024
Available online 2 August 2024

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

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 (Viale Pereira, Cunha, Lampoltshammer, Parycek, & Testa, 2017). 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 (Bello & Zeadally, 2019; Keoh, Kumar, & Tschofenig, 2014). The absence of standardization can result in challenges related to data integration, data quality, and potential security vulnerabilities (Singh et al., 2020). 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 (Islam et al., 2021; Mingxiao, Xiaofeng, Zhe, Xiangwei, & Qijun, 2017). By 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 (Bhushan, Sahoo, Sinha, & Khamparia, 2021). 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 (Ante, 2021). This enables the enforcement of uniform data formats, protocols, and security measures across multiple systems and devices (Macrinici, Cartofeanu, & Gao, 2018; Makhdoom, Zhou, Abolhasan, Lipman, & Ni, 2020). 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 (Mora, Mendoza-Tello, Varela-Guzmán, & Szymanski, 2021). Using blockchain technology to address data security challenges in interoperable smart services holds significant promise. 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 (Nguyen, Pathirana, Ding, & Seneviratne, 2020). However, the adoption of blockchain technology in smart cities requires careful consideration of various factors such as scalability, interoperability, and regulatory compliance (Xie et al., 2019). 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. SDN proves to be a valuable asset in the implementation of network infrastructure for smart cities (Mamatas, Demiroglou, Kalafatidis, Skaperas, & Tsaoussidis, 2023). However, certain limitations of SDN pose significant challenges to its utilization in smart cities, with the risks introduced by single points of failure (Galluccio, Milardo, Morabito, & Palazzo, 2015). In contrast, blockchain technology, as a distributed ledger, offers a means to create a tamper-proof and secure network (Bannour, Souihi, & Mellouk, 2017). 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 (Ali, Irfan, Alwadie, & Glowacz, 2020; Sharma, Singh, Jeong, & Park, 2017; Zhou et al., 2016). Moreover, the establishment of a secure and efficient communication infrastructure is critical for the effective functioning of interoperable services within smart cities, and SDN plays a crucial role in enabling seamless communication and collaboration among various systems and devices.

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 (Yazdinejad, Parizi, Dehghantanha, Zhang, & Choo, 2020). 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 (Aujla, Singh, Bose, et al., 2020). 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 (Iqbal, Abbas, Daneshmand, Rauf, & Bangash, 2020; Latif et al., 2022).

1.1. Motivation

The increasing number of smart city 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 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 1. 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 (Alsaeedi, Mohamad, & Al-Roubaiey, 2019; Aujla, Singh, Singh, et al., 2020; Huang, Fang, Qian, & Hu, 2020). 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.

1.2. 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 maintain cooperative endeavours' integrity.
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 2 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 3 provides insights into the

Table 1
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 (Ismagilova, Hughes, Rana, & Dwivedi, 2022; Tang, Kang, Fan, Li, & Sandhu, 2019)	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 (Siddiqui, Hameed, Shah, Khan, & Aneiba, 2023; Žarko et al., 2019).
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, data consistency, integrity, and smart contract execution, contribute to a secure mechanism for syntactically interoperable smart services (Asif et al., 2022; Bhushan et al., 2020).	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 and interoperability. A blockchain-based regulatory mechanism is necessary to govern security rules in 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. (Medhane, Sangaiah, Hossain, Muhammad, & Wang, 2020; Salman, Elhajj, Kayssi, & Chehab, 2016).	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 (Benkhaled, Hemam, & Maimour, 2022; Marshoodulla & Saha, 2022; Ullah et al., 2020).
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 (Ali, Ahmad, et al., 2020; Wang, Wang, & Chen, 2023).	Existing literature also highlights the absence of an adaptive security solution for achieving interoperability of smart services in smart cities (Koo & Kim, 2021; Zubaydi, Varga, & Molnár, 2023).

integration and operation of SDIoT, multi-chain blockchain technology, and smart contracts within the architecture. Section 4 provides an overview of the security framework by describing the submodules of the overall architecture. Section 5 discusses the execution of smart contracts in the engines for enacting smart governance. In Section 6, we discuss the description of the use case. Section 7 provides the implementation details about the test bed used. Section 8 presents the evaluation results of the comparative system performance of the proposed framework. Section 9 presents the experiments conducted for adaptive security assessment and finally we conclude the paper in Section 11.

2. 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 (Balcerzak et al., 2022; Meijer & Bolívar, 2016; Pereira, Parycek, Falco, & Kleinhans, 2018). 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, Kumar, Kerrache, and Iqbal (2022) 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. Antonios, Konstantinos, and Christos (2023), Hui, Sherratt, and Sánchez (2017), Rao and Deebak (2022) extensively discuss the security and privacy challenges associated with achieving syntactic interoperability in smart cities. Maciel, David, Claro, and Braga (2017) highlight the importance of interoperability in the context of regulatory policies, addressing the limited adoption of commercial smart city technologies. On the other hand, (Agbaje et al., 2022; Ibrar et al., 2022; Tosic et al., 2022) 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, Labiod, & Ampt, 2019) specifically focuses 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. (2022) 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. (2023) present the Blockchain Agnostic Interoperability Framework (BAILIFF), focusing on notary services and cross-chain attestation for verification. In Guvenc et al. (2018), Motlagh, Taleb, and Arouk (2016), the authors explore interoperable services for drone tracking in smart city networks. Basheer and Itani (2023) discusses the integration of Fog Computing, IoT, and MANETs to achieve interoperable systems in sustainable cities. Chen et al. (2023) present Vehicle as a Service (VAAS) as an interoperable service for vehicles. Batayneh et al. (2021) highlights the lack of proper security governance in smart city development and its impact on collaborative tasks. Dua et al. (2017) propose a secure message transmission system using elliptic curve cryptography for interoperable cars in smart cities. Reegu et al. (2022) emphasize the role of blockchain in healthcare and its challenges in achieving syntactic interoperability. Alshboul, Bsoul, Al Zamil, and Samarah (2021), Kharche and Dere (2022), Sookhak, Tang, He, and Yu (2018) analyze security and privacy challenges in syntactically interoperable services using blockchain in smart cities. Bellavista et al. (2021) discuss the role of blockchain technology in the fourth industrial revolution (Industry 4.0) and highlight issues with interoperation between blockchains. Villarreal, García-Alonso, Moguel, and Alegría (2023) focus on blockchain's potential for interoperability and security

Table 2
Comparative analysis of existing solution with proposed solution.

S-no	Studies	Access level security			Information security			Governance mechanism	
		Authentication	Authorization	Trust	Confidentiality	Integrity	Availability	Security Governance	Adaptive Security Governance
1	Meijer and Bolívar (2016)	✓	✓	x	✓	x	x	✓	x
2	Rathee et al. (2022)	✓	x	✓	✓	x	x	✓	x
3	Msahli et al. (2019)	✓	x	✓	✓	x	x	✓	x
4	Rahman, Chamikara, Khalil, and Bouras (2022)	x	x	✓	✓	x	✓	x	x
5	Karumba, Jurdak, Kanhere, and Sethuvenkatraman (2023)	✓	✓	✓	✓	✓	x	✓	x
6	Guvenc, Koohifar, Singh, Sichitiu, and Matolak (2018)	✓	✓	✓	x	x	x	✓	x
7	Batayneh et al. (2021)	x	✓	x	x	x	x	x	x
8	Dua, Kumar, Das, and Susilo (2017)	✓	✓	x	✓	✓	✓	✓	x
9	Knowles Flanagan (2022)	✓	x	x	x	x	x	x	x
10	Xu, Chen, Blasch, and Chen (2018)	✓	✓	✓	✓	✓	✓	✓	x
11	Gilani et al. (2023)	x	✓	x	x	x	x	✓	x
12	Proposed	✓	✓	✓	✓	✓	✓	✓	✓

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.

Knowles Flanagan (2022) propose a decentralized protocol and data exchange framework for human-driven and connected autonomous vehicles to achieve interoperability in intersections. Xu et al. (2018) introduce BlendCAC, a decentralized capability-based access control mechanism for large-scale interoperable IoT systems using smart contracts. Gilani et al. (2023) propose a vertical SDN-based framework to enhance reliability and stability in smart home interoperable services. Rana and Singh (2023) focus on ensuring compatible and effective communication in the IoT during interoperation utilizing SDN-based architectures. Shamsudheen, Karthik, Anoop, and Gobinathan (2023) discuss the challenges of SDIoT in emergency services for disaster management. Banerjee et al. (2021) propose a secure and scalable scheme for data collection and access control in IIoT using blockchain technology. Latif et al. (2022) emphasize the importance of interoperability of blockchains and SDN in addressing energy and security issues in IoT networks. Kozhevnikov, Svitek, and Skobelev (2022) present a multi-agent system prototype that enables adaptive planning for the interoperable services of gas, water, and energy resources in smart cities.

Recently Buldas, Draheim, Gault, et al. (2022) 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 (Buldas, Draheim, Gault, et al., 2022). As such, it shares objectives with other multi-chain blockchain technologies such as Polkadot (Polkadot, 2023). In service of universal asset tokenization, the Alphabill platform (Buldas, Draheim, Gault, et al., 2022) 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 (Buldas, Draheim, Gault, et al., 2022).

The existing literature primarily emphasizes augmenting the communication architecture of SDIoT and the interface between SDIoT and Blockchain to enhance security measures. Their primary objective is to ensure interoperability among services in smart city networks. Yet, there is a discernible gap concerning the integration of adaptive security measures into established security governance standards, especially during service interoperability. Table 2 provides a comparative analysis between current solutions and the proposed approach. Our proposed security framework introduces an innovative approach by employing smart contracts to establish a dynamic security governance mechanism, enabling adaptive security measures.

3. System overview

This research work 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 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. The programmable scripts detailing the setup and parameters of our proposed solution are publicly available on both GitHub[1] and Code Ocean[2] platforms.

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

3.1. 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 (EL-Garoui, Pierre, & Chamberland, 2020). 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 (Ogrodowczyk, Belter, & LeClerc, 2016) 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, Fig. 3 depicts the usual SDIoT architecture.

3.1.1. Application layer

The application layer is a crucial component of the SDIoT architecture that enables the deployment of various smart city services (Li, Chen, & Fu, 2019). 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

[1] https://github.com/Shahbazdefender/A-Smart-Contract-Based-Adaptive-Security-Governance-Architecture-for-Smart-City-Service/blob/master/README.md

[2] https://codeocean.com/capsule/1559462/tree

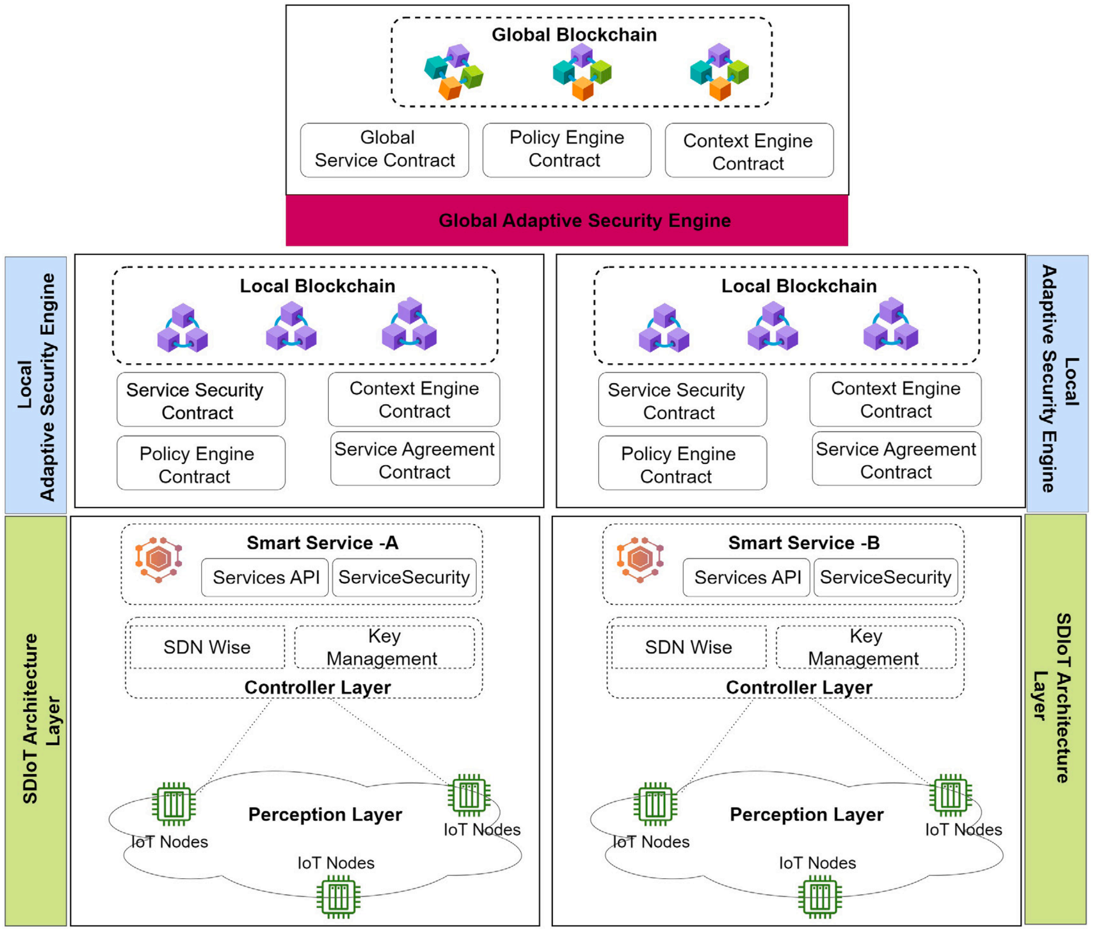

Fig. 2. Proposed security framework for collaborative services.

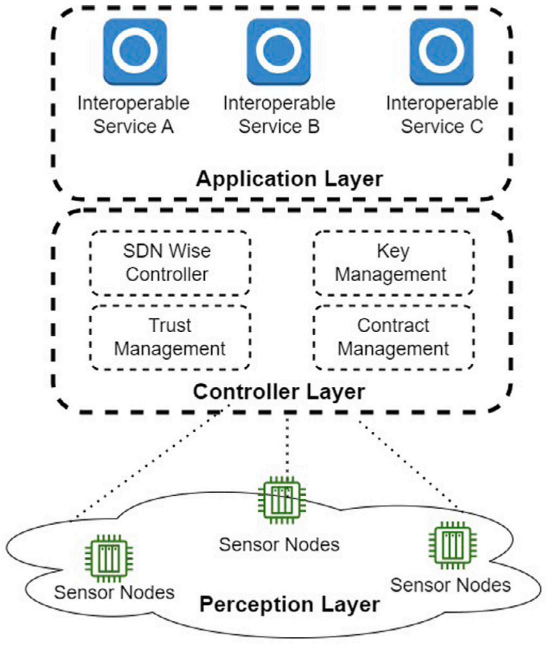

Fig. 3. Typical SDIoT architecture.

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.

3.1.2. 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 (Galluccio et al., 2015). 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 (Mostafaei & Menth, 2018).

3.1.3. 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 (Mrabet, Belguith, Alhomoud, & Jemai, 2020).

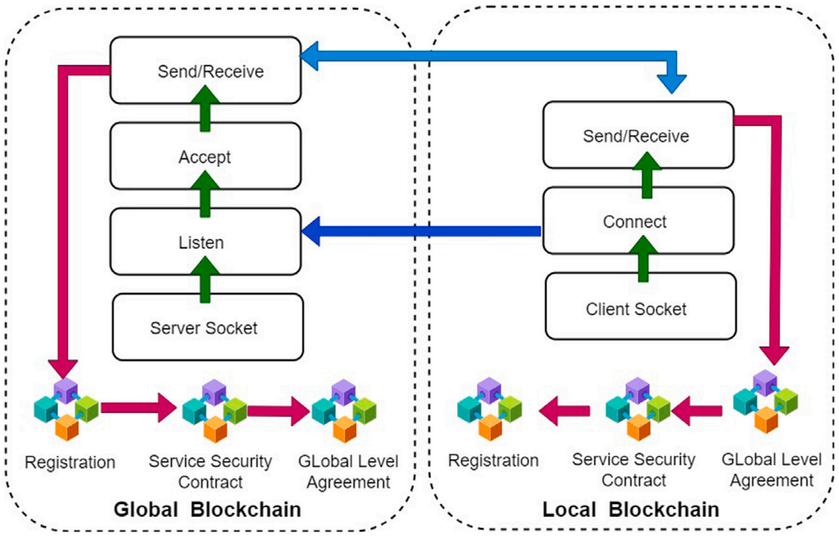

Fig. 4. Client–Server architecture for proposed security framework.

3.2. *Adaptive security engine layers*

On top of the SDIoT layer, our proposed security framework includes two additional levels: global and local adaptive engine layers. These layers include blockchain technology, which is vital for enforcing norms and policies during message exchange across collaborative and interoperable services in smart cities. To achieve this blockchain connection, we use Multichain 2.0, a strong and scalable blockchain-centric platform. Multichain 2.0 is purposely designed to provide a secure foundation for designing and running decentralized applications, as well as a variety of enhanced security features. Multichain 2.0 supports numerous chains, giving developers the ability to precisely design blockchain networks according to their own needs and preferences. This platform enables the autonomous operation and smooth interaction of various chains (MultiChain, 2023a, 2023b). Multi chains are currently emerging with Web3 (Buldas, Draheim, Gault, & Saarepera, 2022; Edelman, 2022; Esber & Kominers, 2022; Jin & Parrott, 2022; Stackpole, 2022) (not to be confused with Web 3.0 (Berners-Lee, Hendler, & Lassila, 2001; SemnaticWeb, 2023)), which takes blockchain to the 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 (Buldas, Draheim, Gault, & Saarepera, 2022; Edelman, 2022). As such, the Web3 vision is about consolidating and integrating the fragmented landscape of specific blockchain visions expressed in the many initial coin offerings (ICOs) that we have seen over the last decade; and multi-chains are the natural fit to form the technological basis of Web3.

Fig. 4 illustrates the internal client–server architecture of the integrated blockchain. In this architecture, client nodes serve as decentralized smart interoperable services, while server nodes act as decentralized servers responsible for security governance. Following are the validation chains in client and server multi-chain:

1. ***Registration***: The registration validation chain is responsible for maintaining 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 endeavours within smart cities.

3.2.1. *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 (Zou et al., 2019). 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 (Khan, Loukil, Ghedira-Guegan, Benkhelifa, & Bani-Hani, 2021). 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 the 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. ***SPolicy 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.

4. 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 the 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 detailing the implementation of security modules within the SDIoT and Adaptive Engines layers, accomplished through integration with the SDN-WISE controller based on the Java platform and the utilization of smart contracts based on the Python platform. This encompasses authentication and access control components, ensuring the system's secure and resilient operation. 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. This integration ensures that security policies are consistently applied across the system, enhancing overall reliability and trustworthiness.

4.1. SDIoT security modules

In this section, we will discuss the implementation details of the security modules residing in the SDIoT layer. These modules are implemented within the SDN-Wise controller by integrating them with the pre-existing modules already housed within the SDN-Wise controller.

4.1.1. 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.

Definition 1 (*Key Management*)**.** Let S_i and S_j represent two smart services, each with sets of client nodes denoted as C_i and C_j. Each client node has its own key-generating module, denoted by K_α, which acts as a repository of ECC keys (128, 192, 256 bits). The key-generating module executes a key generation function δ to generate keys for the services and their client nodes. These keys are formally represented as a set of 3-tuples (α, β, γ), where:

- α represents service key-pairs,
- β represents client key-pairs,
- γ represents SDN key-pairs.

Formally, the set of keys stored is defined as:

$$K(t) = \{(\alpha_i, \beta_i, \gamma_i) \mid i \in S, \exists j \in C\} \tag{1}$$

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 initialized, 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.

4.1.2. 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.

Definition 2 (*Smart Service Management*)**.** Let $\text{SmartService}_{i..n}$ represent a set of smart APIs denoted as $\text{API}_{i..n}$ responsible for executing a Smart Service management function M. This function M appends service trust keys and the Smart Service APIs. Formally:

$$M(\text{SmartService}_{i..n}) = \{\text{ServiceAPIs}, \text{Trust} = 0\} \tag{2}$$

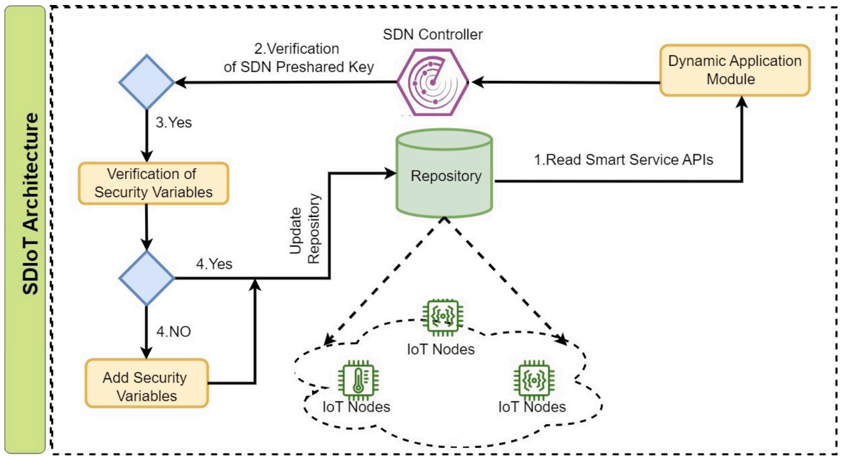

Fig. 5. Smart service management.

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 for providing 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. Fig. 5 displays the workflow of the smart service management.

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
```

4.2. Security modules of adaptive engine's layers

The adaptive engines encompass both the rule engine and the execution engine. The execution engine comprises the policy engine, context engine and process execution. Each module operates using smart contracts to execute specific security functionalities. The security validation mechanism between the local and global adaptive engines is established through the security validation chain embedded within the blockchain environment. Similarly, we adopt similar approaches to integrate these engines into the SDIoT architecture. We utilized the blockchain's inherent key management modules to generate digital identities for the global and local blockchains. This strategy aims to facilitate seamless operations by ensuring that both the global and local blockchains possess access to their respective keys.

4.2.1. 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.

Definition 3 (*Rule Engine*)**.** Let SmartService$_{i..n}$ represent a set of smart APIs denoted as API$_{i..n}$ responsible for executing a Smart Service management function Z. This function Z converts security rules from the service security definitions into two sets: a local service contract for the local adaptive engine and a global service contract for the global adaptive engine. Formally, the Rule Engine function Z is defined as:

$$Z(\text{SmartService}_{i..n}) = (\text{LocalServiceContract}, \text{GlobalServiceContract}) \tag{3}$$

where Local Service Contract and Global Service Contract represented as a set of 5-tuple $\alpha, \beta, ServiceAPIs, Trust$ where α represent services key-pairs β represent client key-pairs and γ represent SDN key-pairs

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)
```

Algorithm 3 outlines the implementation of service security contracts, involving the use of an array of dictionary objects to add new security requirements for smart services, including authentication, authorization, and access control mechanisms.

To activate the rule engine in both adaptive engines, we implement two separate smart contracts: the **Global Security Contract** within the local adaptive engine and the **Service Security Contract** within the global adaptive engine, facilitated by the **Service security.Json** file. These contracts ensure that critical security requirements, such as access control, trust for authorization, and authentication, are integrated into services enabling their seamless interaction. The validation attributes are securely stored within the validation chains of the **Global Service Agreement Contract** and the **Local Service Security Contract** in the blockchains. Eq. (4) depicts the structure of the contract's JSON messages. After this, the Rule engine transfers the pending request to the execution engine.

$$Contract = \begin{bmatrix} Service\ variable || Authentication\ variable \\ || Trust\ variable || Collaborative\ task \end{bmatrix} \tag{4}$$

$$\begin{cases} Service\ variable = \text{Security Validation attributes} \\ Authentication\ variable = \text{SDIoT, Blockchains(Public, Private Keys)} \\ Trust = \text{Value of trust Factor} \end{cases}$$

4.2.2. Execution engine

The execution engine comprises distinct smart contracts, that include the policy contract, the context execution contract, and the execution process contract. The policy engine contract in our proposed security framework is defined 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 for providing the local security rules, while the global adaptive engine uses the global blockchain to enforce the global security rules onto the local ones. 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 4 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.

The Policy Engine, on the other hand, is responsible for fetching the latest updated service security from the Global Adaptive Engine server Blockchain and converting it into JSON Format as shown in the Algorithm 5. To execute of Policy contract the module is responsible for forwarding the policy to the Execution Engine. The engine has its policy management operations (such as adding, deleting, and appending new policies) in order to provide flexibility during the interoperation of smart services.

Definition 4 (*Policy Engine*)**.** Let $\text{SmartService}_{i..n}$ represent a set of smart APIs denoted as $\text{API}_{i..n}$ responsible for executing a Smart Service management function X. This function X fetches contracts from blockchains, combines them, and converts them into a comprehensive policy. Formally:

$$X(\text{SmartService}_{i..n}) = \text{LocalServiceContract} \cup \text{GlobalServiceContract} \quad (5)$$

Algorithm 4 : Local Context Engine

```
Require: Authentication, Authorization, Access
Ensure: Context Execution
 1: $Trust = 0$
 2: while $True$ do
 3:     $Bool$ =Verification of Authentication Variable
 4:     if $Bool == True$ then
 5:         $x = 1 - \alpha$
 6:         $ServiceTrust = x * Trust + \alpha * 0.001$
 7:         $Trust = ServiceTrust$
 8:         Send the Service Trust to Blockchain
 9:         if $Trust \geq$ Pragramatical Variable then
10:             $Access = Grant$
11:             Call Collaborative Task
12:         end if
13:     end if
14: end while
```

After successful verification of authentication, the authorization is completed with the help of trust assessment to access the collaborative task during the interoperation of smart services. For trust assessment, we are using a modified version of Bao, Chen, and Guo (2013) as defined in the following Definition 5:

Definition 5 (*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:

$$T_{ij}(t) = \alpha\, D_{ij}(t) + (1-\alpha)\, T_{ij}(t - \Delta T) \quad (6)$$

Algorithm 5 : Policy Engine Contract

```
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
```

5. 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

5.1. 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. Fig. 6 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. Eq. (7) shows the structure of the request message. It is assumed that the local blockchain and SDN controllers have already shared their public keys with each other by sharing the hash of their public keys.

$$Message_{Enc} = \left[(node_{ij}(pub) || hash(SDN_{pub}))_{sk} \right]_{pb} \quad (7)$$

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 Eq. (8).

$$Message_{dec} = \left[(node_{ij}(pub) || hash(SDN_{pub})) \right] \quad (8)$$

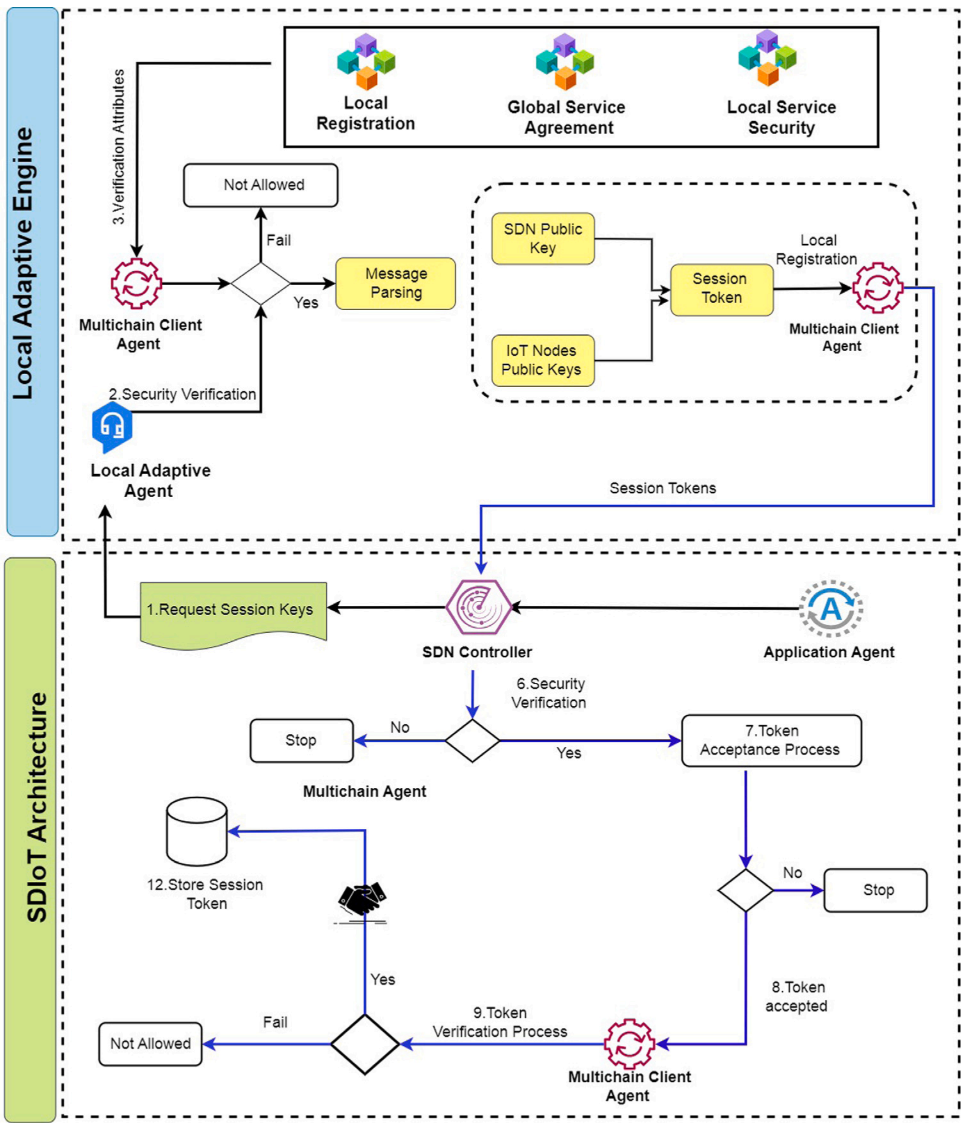

Fig. 6. Workflow of session token generation.

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 Eq. (9).

$$Session_{ID} = \left[(Multisig(node_{ij}(pub), (chain_{pub}))) \right] \tag{9}$$

5. Multi-signature protocol returns the session ID which is then stored in the validation chain of local and global chains and forwarded to the SDN-wise controller.
6. The Encrypted $Session_{id}$, is then sent back to the controller by encrypting with the public key SDN controllers and signing it with the Local and the global blockchain private key. Eq. (10) shows the sent message structure.

$$SessionContract_{Enc} = \left[(Session_{sk}) \right]_{pb} \tag{10}$$

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. Lastly, the controller calls 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.

5.2. Adding local service security contract

Interoperability between services requires 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. Fig. 7 shows the workflow steps.

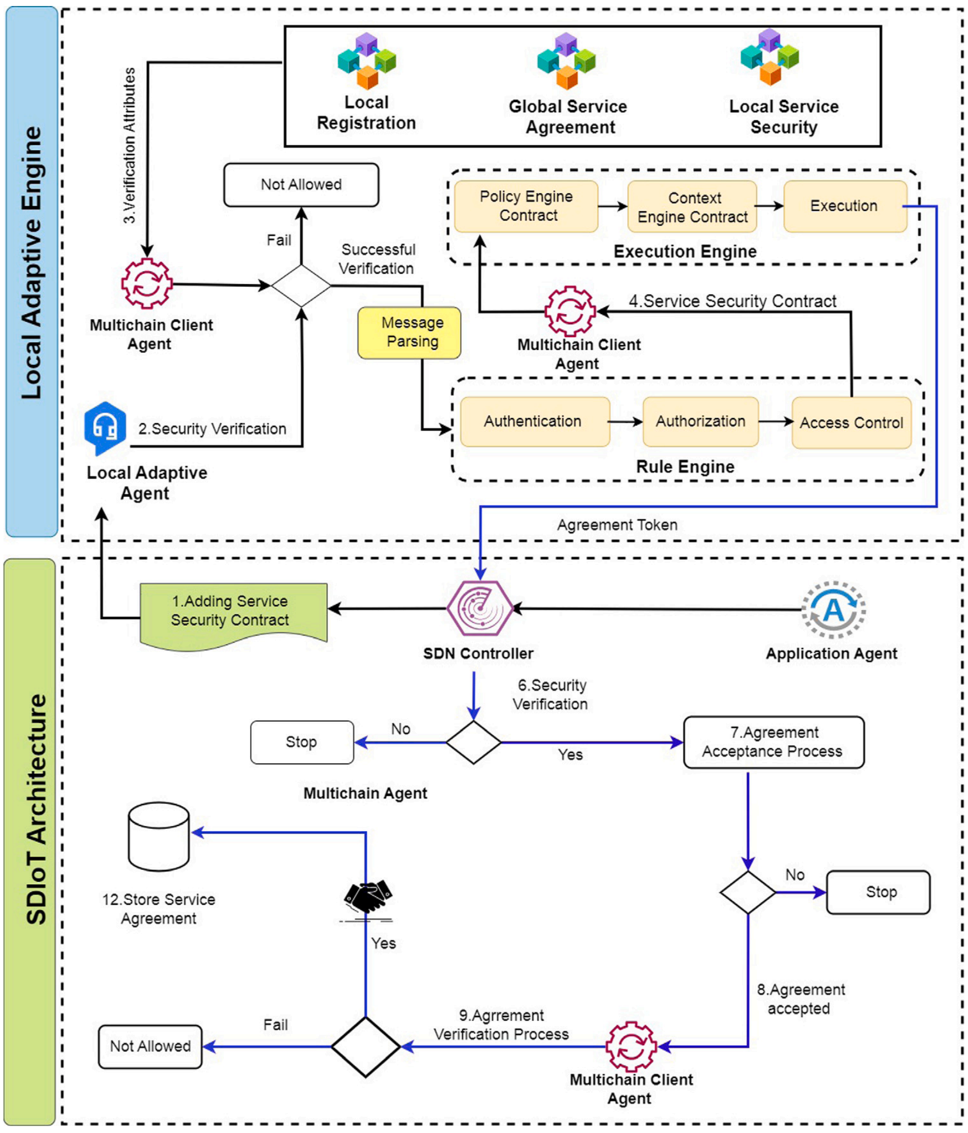

Fig. 7. Workflow of adding local service security.

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 for providing 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 to the Local adaptive engine. Eq. (11) shows the structure of the request message. It is assumed that the local blockchain and application agent have already shared their public keys with each other by sharing the hash of their public keys.

$$Message_{Enc} = \left[(Request||Hash(SessionKey))_{sk}\right]_{pb} \tag{11}$$

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 Eq. (12) as,

$$Message_{dec} = \left[(Request||Hash(SessionKey))\right] \tag{12}$$

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 for adding 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 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

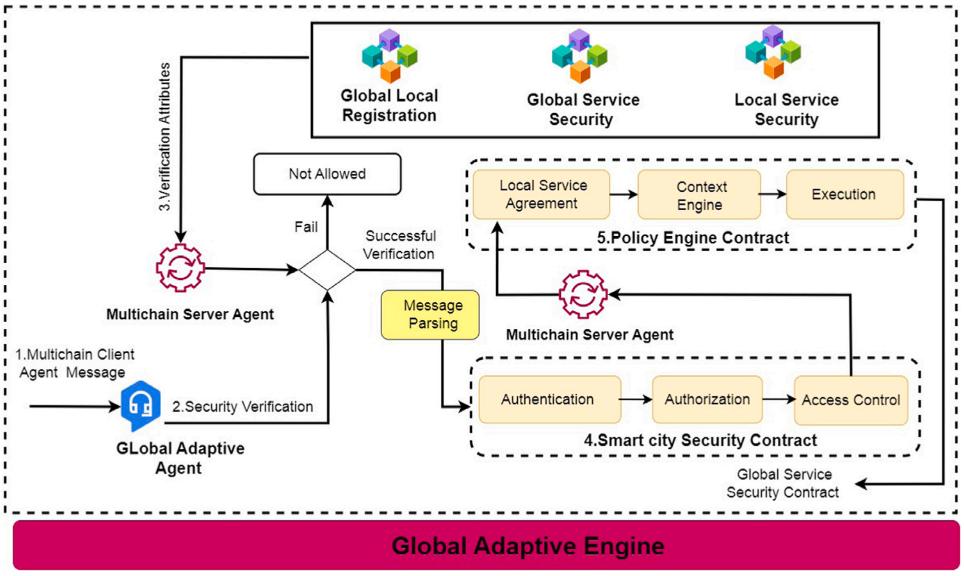

Fig. 8. Workflow of global service security contract.

the hash of the session ID. Eq. (13) shows the send message structure,

$$Contract_{Enc} = \left[(Contract||Hash(SessionKey))_{sk}\right]_{pb} \tag{13}$$

6. The application agent is responsible for decrypting 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 a "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.

5.3. 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. Fig. 8 shows the pictorial view Adding Global Service security contract. The Global adaptive engine is responsible for creating smart city administrative service security contracts to enforce smart city administrative security policies during the interoperation of smart services. It is also responsible for receiving 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 Eq. (14) as,

$$M_{Enc} = \left[(Agreement_{Txid}||Hash(Session))_{sk}\right]_{pb} \tag{14}$$

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 Eq. (15) as,

$$M_{dec} = \left[(Agreement_{Txid}||(Hash(Session)))\right] \tag{15}$$

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.
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 Eq. (16), where A represents the local service security contract and B represents the administrative security contract of the smart city.

$$GlobalContract = \left[A||B\right] \tag{16}$$

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.

5.4. 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

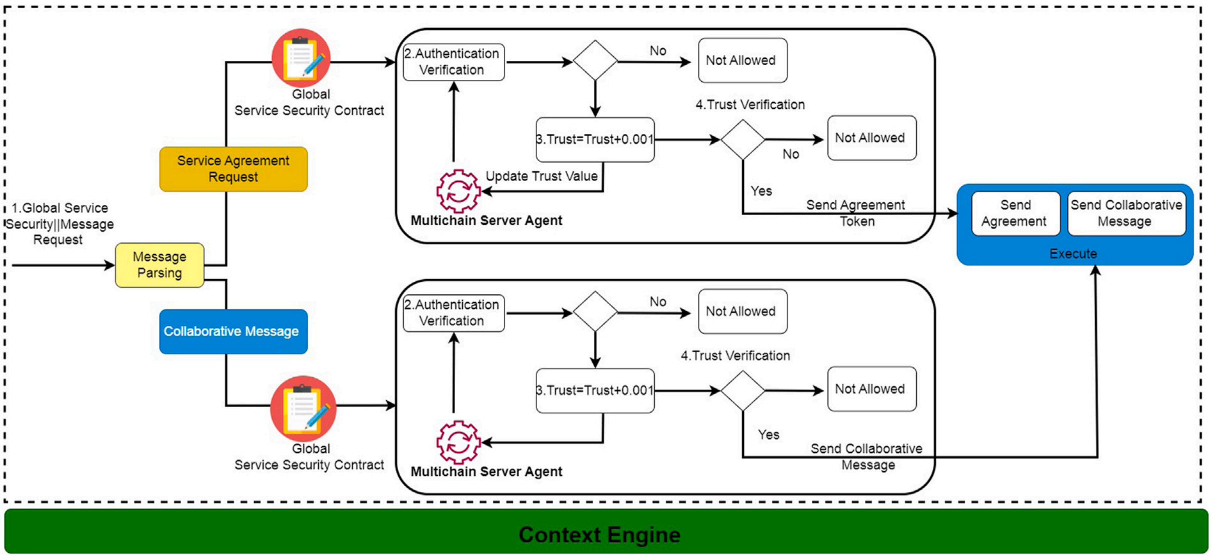

Fig. 9. Workflow of context engine.

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. Fig. 9 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 be verified 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 for providing security requirements to respond to messages.

6. 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 (Shah, Seker, Hameed, & Draheim, 2019; Shah, Seker, Rathore, et al., 2019). As smart city technologies continue to advance, there is a growing need for interconnected services to operate seamlessly and collaboratively, particularly in emergencies (Kashef, Visvizi, & Troisi, 2021). 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 data, assess the severity of the situation, and coordinate response efforts across multiple services and stakeholders (Costa et al., 2022). 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 Fig. 10, where three services interact with each other in order to execute disaster emergency response systems. Following is detailed information on 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

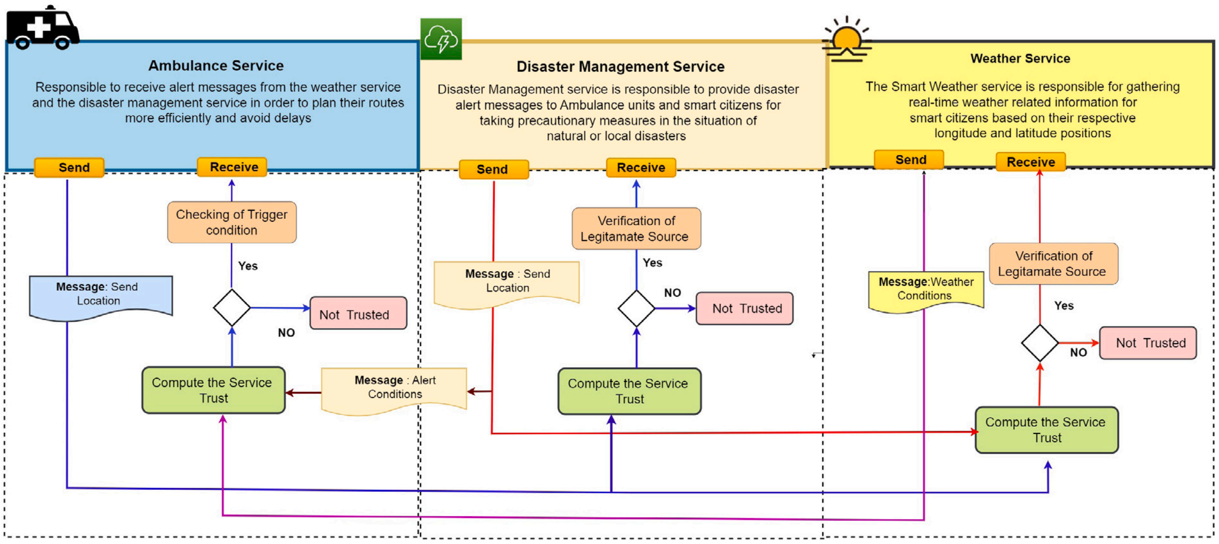
Ambulance Service
Responsible to receive alert messages from the weather service and the disaster management service in order to plan their routes more efficiently and avoid delays
Send
Receive
Checking of Trigger condition
Yes
NO
Not Trusted
Message: Send Location
Compute the Service Trust
Message : Alert Conditions
Disaster Management Service
Disaster Management service is responsible to provide disaster alert messages to Ambulance units and smart citizens for taking precautionary measures in the situation of natural or local disasters
Send
Receive
Verification of Legitamate Source
Message : Send Location
Yes
NO
Not Trusted
Compute the Service Trust
Weather Service
The Smart Weather service is responsible for gathering real-time weather related information for smart citizens based on their respective longitude and latitude positions
Send
Receive
Message:Weather Conditions
Verification of Legitamate Source
Yes
NO
Not Trusted
Compute the Service Trust

Fig. 10. Communication workflow of interoperable services for smart emergency response.

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.

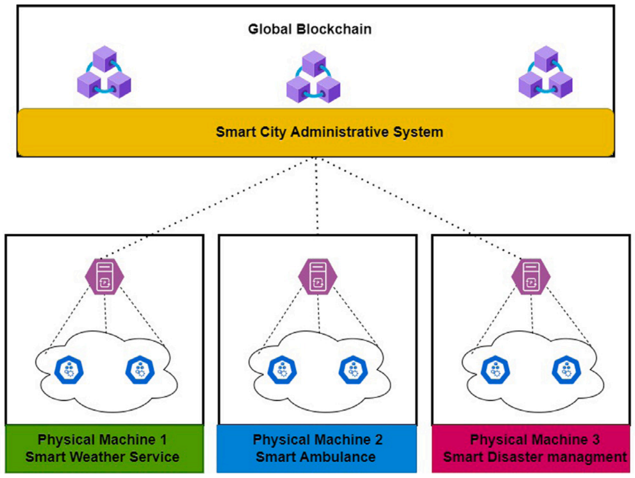

Fig. 11. An illustrative network architecture of the proposed use case.

Table 3 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.

7. Testbed and implementation discussion

We simulate a smart city network using the COOJA network simulator, an open-source tool based on the Contiki operating system (Thomson et al., 2016). COOJA, based on the Contiki operating system (Thomson et al., 2016), is widely used for simulating wireless sensor networks and IoT networks (Oikonomou et al., 2022). The Contiki architecture comprises multiple modules offering features such as process management, memory management, and inter-process communication to IoT

Table 3
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

nodes or motes. It supports network protocols such as IPv6, RPL, 6LoWPAN, and CoAP for low-power, lossy networks (Zikria, Afzal, Ishmanov, Kim, & Yu, 2018). We integrate the Contiki operating system with the SDN-Wise controller for the SDIoT architecture. SDN-Wise acts as middleware, offering a programming abstraction that allows programmers to build high-level IoT services and applications while hiding the intricate network architecture underlying them. In our Contiki setup, a network of IoT nodes serves as client nodes for interoperable services, with SDN-Wise controllers facilitating communication mechanisms between them. Additionally, all client nodes utilize the key management module for their identities through public and private keys, and smart services management modules in the SDN controller enable them to access the interoperable services effectively.

In our Programming environment, we establish a decentralized client–server interoperable service environment, where the disaster management service functions as the server while the others operate as clients. Each of these decentralized client–server interoperable services utilizes the MultiChain blockchain to securely store all transactions in a decentralized manner. This ensures that the transaction records remain immutable and transparent across the entire network, enhancing trust and reliability. MultiChain primarily employs a default consensus mechanism known as practical Byzantine fault tolerance (PBFT). PBFT ensures that all nodes in the network reach a consensus on the validity of transactions and the order in which they are added to the blockchain. This consensus mechanism is renowned for its efficiency and resilience in permissioned blockchain networks like MultiChain. Additionally, within this environment, each service runs subprocesses, including a rule engine, policy engine, and context engine, which work in tandem to execute the interoperable service ecosystem efficiently. Together with smart contracts, these components facilitate seamless operations and interaction among the services while ensuring data integrity and security.

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 600 ms, 120 ms, 60 ms, 30 ms, 15 ms, and 5 ms. 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,

Table 4
Hardware configuration of the four physical machines.

	Physical machines 1, 2, 3, 4
CPU (Processor)	Intel® 6th Gen Intel® Core™ i7 (6700)
Memory	16 GB DDR
Chipset	Intel® H110 Chipset
Hardrive	256 GB Solid State Drive SATA

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 4, which outlines the specifications and settings of the hardware used. To realize the network infrastructure required for our use case, we have presented an illustrative network architecture as depicted in Fig. 11.

7.1. 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. Fig. 12 illustrates the workflow for generating and accepting the agreement for interoperability between these services.

1. Service A sends the request through the application agent to the SDN controller in the SDIOT architecture. The message structure of the request message is shown in Eq. (17) (where X represents the name of the interoperable service such as Service A).

$$M_{Enc} = \left[(Request \| X \| Hash(SessionKey))_{sk}\right]_{pb} \tag{17}$$

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 (18) where Agreement Txid is the transactional ID of the local service security.

$$M_{Enc} = \left[(Agreement_{Txid} || Hash(SessionKey))_{sk}\right]_{pb} \tag{18}$$

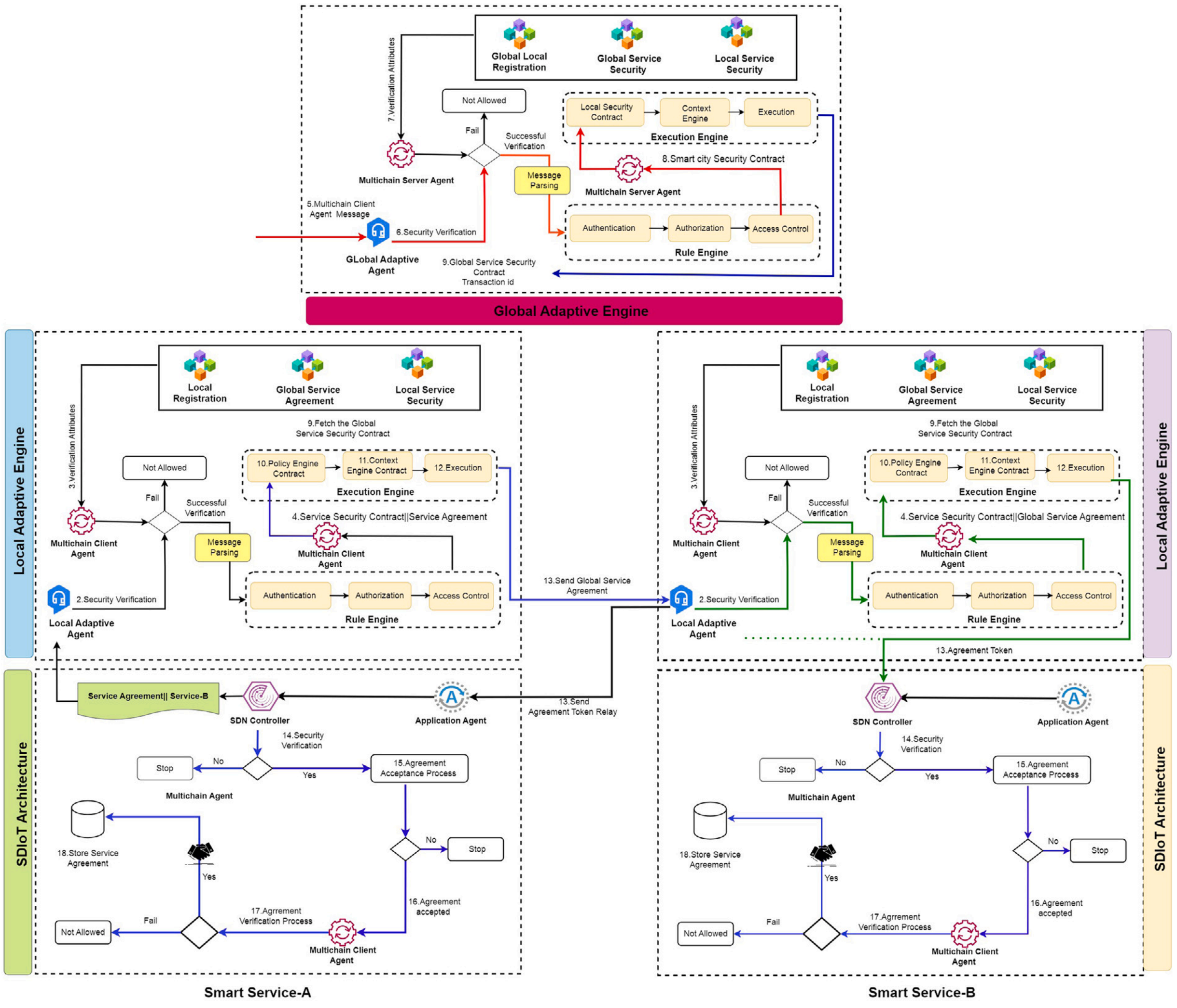

Fig. 12. Service-level agreement between interoperable services.

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 for enforcing 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 forwards 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. Fig. 12 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 for forwarding 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.

By analyzing the data presented in Table 5, 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

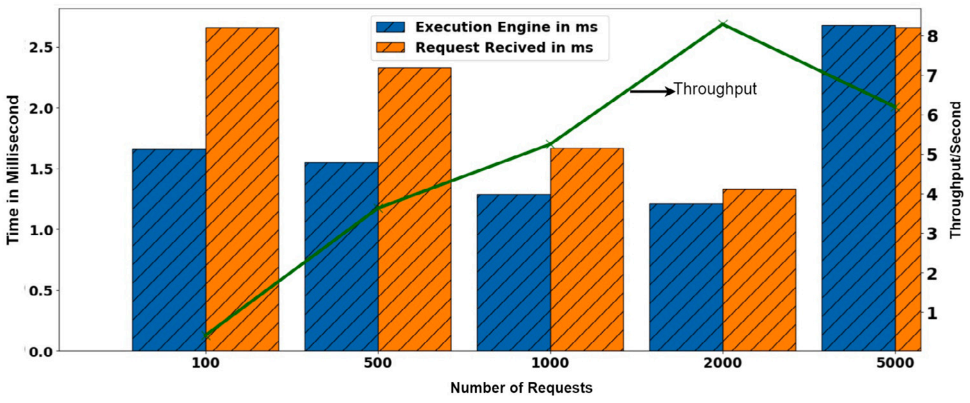

Fig. 13. Performance result with the increased number of requests.

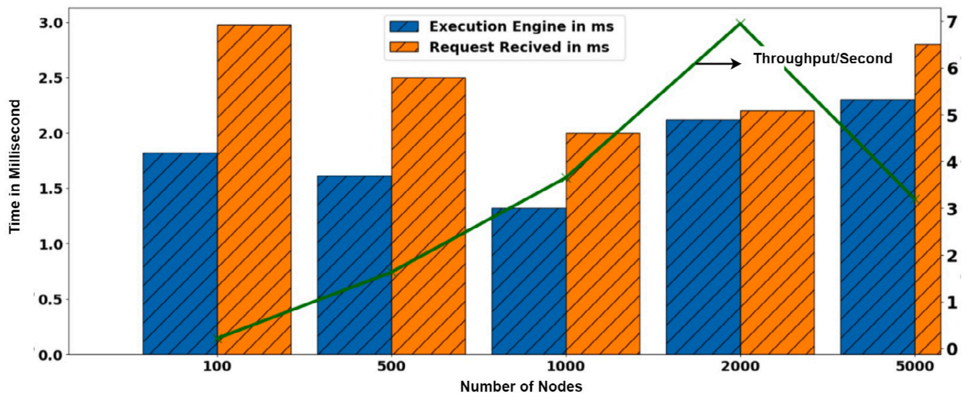

Fig. 14. Performance matrix during service-level agreement with the increased number of nodes.

Table 5
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

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 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. Fig. 13 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 6 we can notice that the performance matrix gets worse when the number of IoT nodes is increased if we compare it with the previous result, this depicts the SDIoT architecture plays a vital role in the performance matrix of the security framework. Fig. 14 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.

7.2. Sending and receiving emergency request 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 agreement is already shared between interoperable services. Fig. 15 shows the pictorial steps of sending and receiving an alert message between interoperable services. Table 7 shows the performance matrix of the proposed security framework by increasing the number of collaborative alert request messages between interoperable services. We observed from the graphical results, as shown in Fig. 16, 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 8 we find out that the performance matrix of the particular process is decreased. Fig. 17 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 6
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

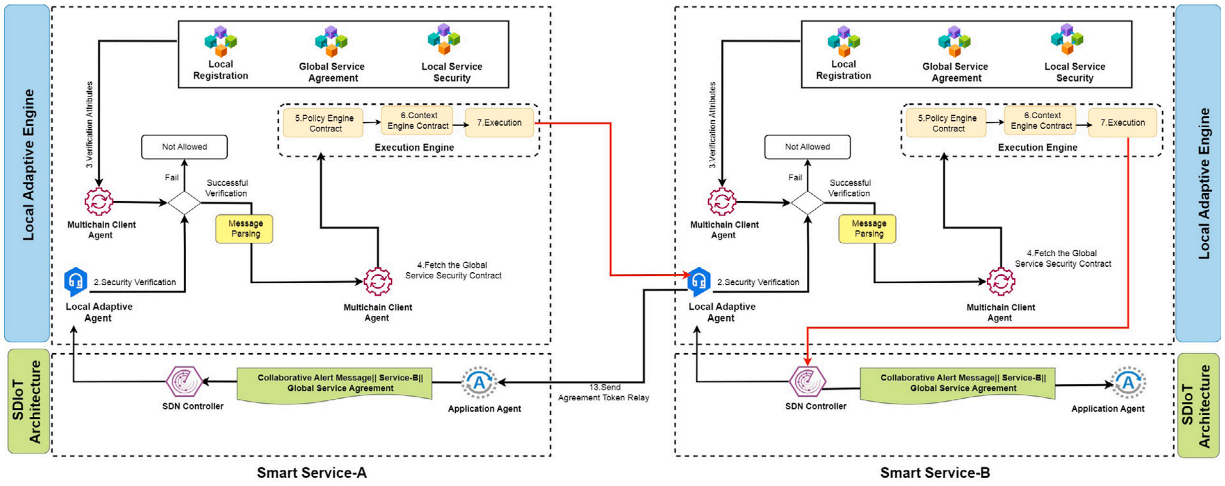

Fig. 15. Collaborative alert sending and receiving message process.

Table 7
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 8
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

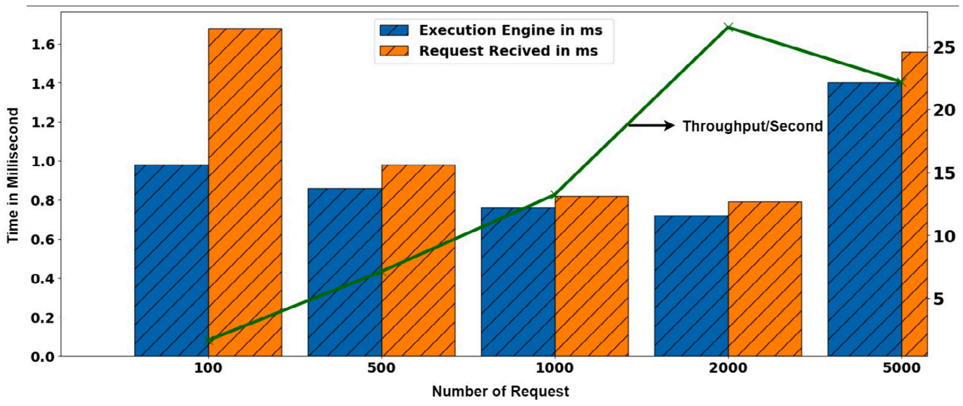

Fig. 16. Performance matrix of sending and receiving emergency request process with an increased number of requests.

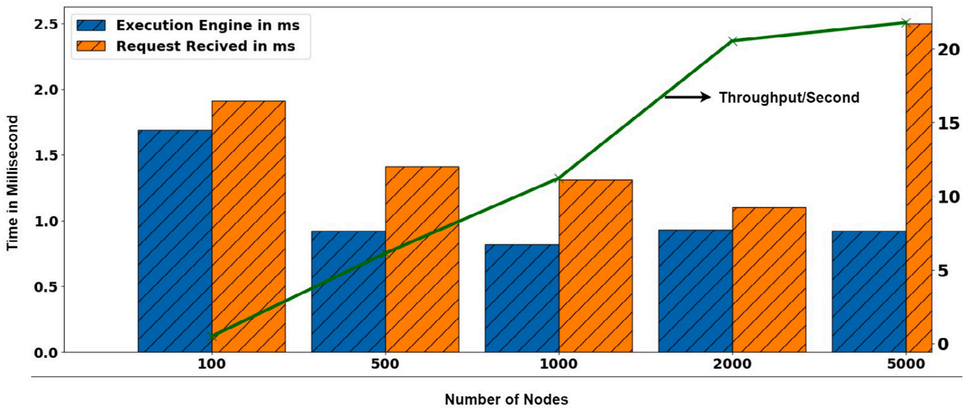

Fig. 17. Performance matrix of sending and receiving emergency request process with increased number of nodes.

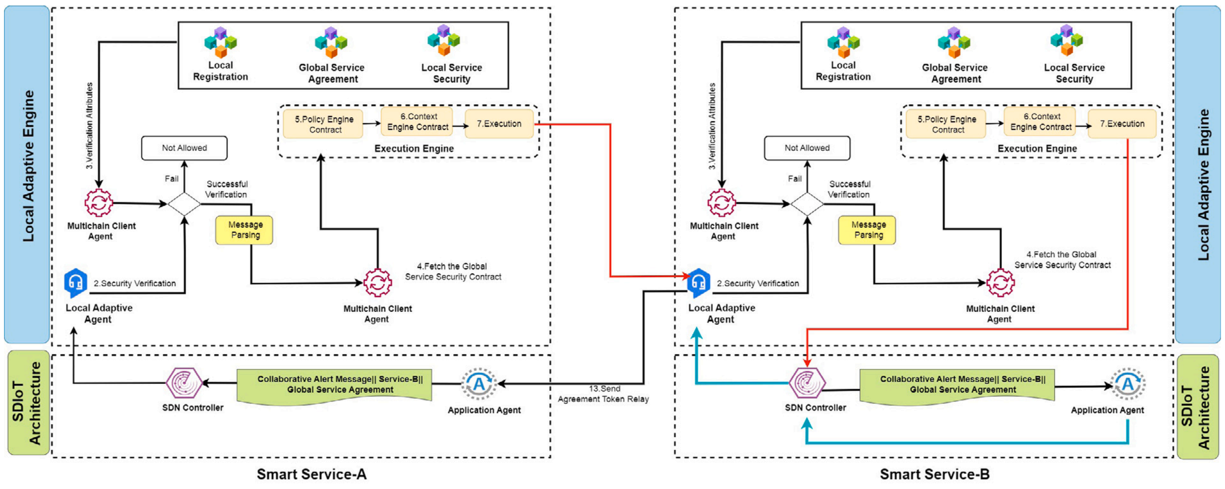

Fig. 18. End-to-End alert message workflow.

7.3. 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. Fig. 18 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 for taking the message from the SDN controller and performing security verification in order to verify the legitimacy of the 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.
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 9. This decline can be attributed to increased computation workload in the End-to-End interaction, specifically the cryptographic encryption and decryption processes. Fig. 20 provides a graphical representation of the performance matrix as the number of requests increases. Additionally, Table 10 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. Fig. 21 visually depicts the performance matrix of the security framework when the number of requests is incrementally increased.

8. Comparative throughput performance of the proposed security framework

The system performance of our proposed security framework in collaborative disaster management and response use cases in smart

Table 9
Performance matrix of end-to-end message sending and receiving process with increased number of requests.

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 10
Performance matrix of 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

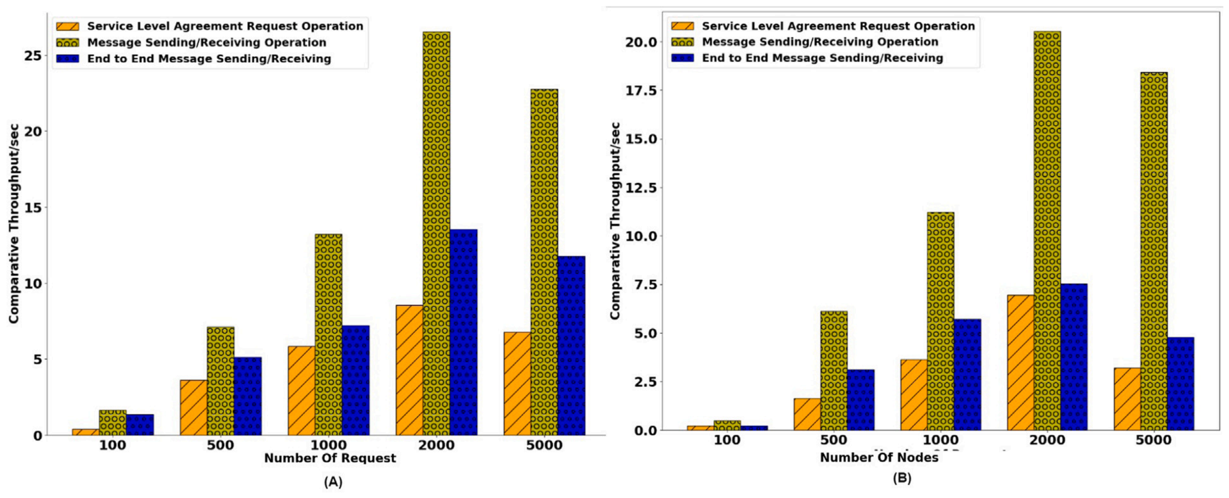

Fig. 19. Comparative system performance result with (A) increased number of requests and (B) increased number of nodes.

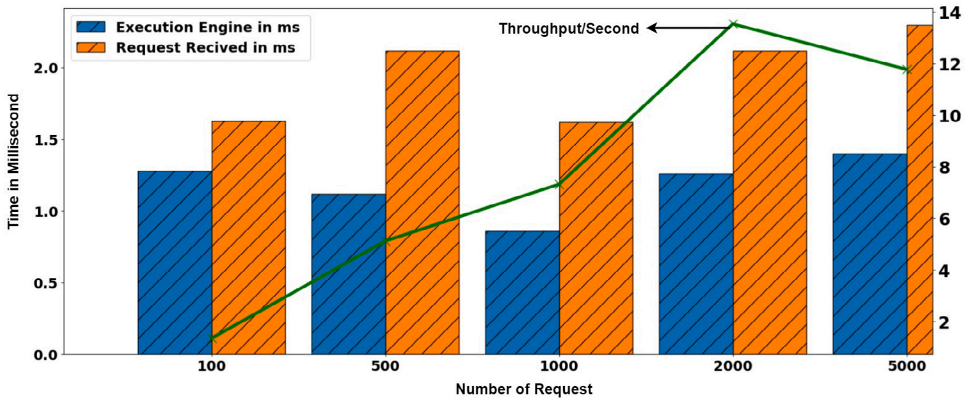

Fig. 20. Performance Matrix of End-End message sending and receiving with the increased number of requests.

cities is evaluated based on the comparative throughput of all processes in two scenarios: one with an increase in the number of requests and the other with an increase in the number of nodes during the interoperation of smart services, as depicted in Fig. 19(A), (B). Initially, the throughput during the process of fetching the service-level agreement is 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 improves, highlighting the effectiveness of our proposed security framework, particularly in supporting delay-sensitive applications. In the SDIoT architecture, the memory pool significantly impacts throughput compared to the memory pool of the "multi-chain" blockchain. When we increase the number

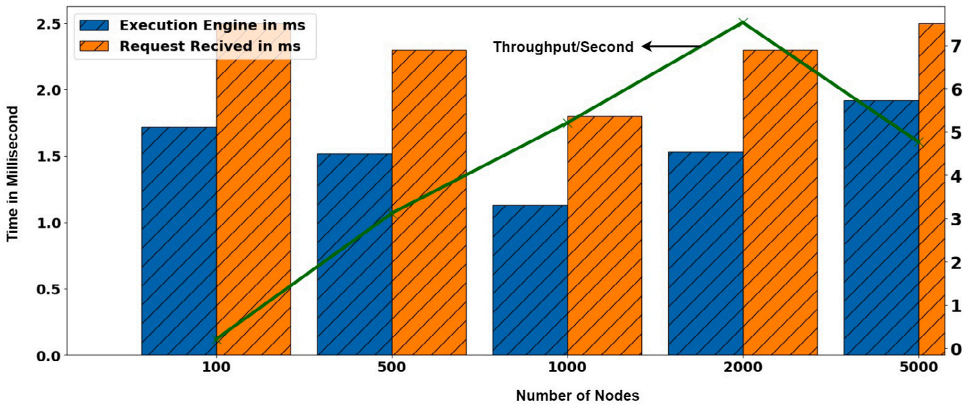

Fig. 21. Performance Matrix of End-End message sending and receiving with the increased number of nodes.

Algorithm 6 : Access Level Security Policy

```
Require: User, Service
Ensure: True
 1: if $Service == disaster\ management$ then
 2:     if $user\ key\ length == 128\ and\ trust >= 0.003$ then
 3:         return True
 4:     end if
 5: end if
 6: if $Service == Weather\ service$ then
 7:     if $user\ key\ length == 192\ and\ trust >= 0.003$ then
 8:         return True
 9:     end if
10: end if
11: if $Service == Ambulance\ service$ then
12:     if $user\ key\ length == 192\ and\ trust >= 0.003$ then
13:         return True
14:     end if
15: end if
```

of nodes in the SDIoT architecture while keeping the request messages constant, the throughput of the security framework decreases. However, the opposite occurs when we increase the number of request messages while keeping the number of IoT nodes constant.

9. Adaptiveness security assessment

To evaluate the adaptability of interoperable smart services during interoperability, we conducted adaptive security assessment experiments. These experiments aimed to measure the speed and effectiveness of the service's adaptive capabilities. Specifically, we implemented an access level security policy mechanism within the interoperable service for collaborative tasks during interoperability, as defined in the following Definition 6:

Definition 6 (*Access Level Security Policy*). Consider two smart services denoted as $Service_i$ and $Service_j$ where $Service_i$ represents the subject (requester) and $Service_j$ represents the object (responder) of collaborative tasks during interoperability. Each service has its own access level security policy function denoted γ and Δ which is represented as a set of 3-tuple (α, β, ζ) where α represent service identification β represent as a set of authentication scheme repository (128,192,256) ζ represent the trust value for authorization to access the collaborative task. The access policy function$\theta(t)$ at the point time$t$ is defined as a combination of the γ and Δ policies for access level security to access the data.

$$\theta(t) = \gamma_{\text{Subject}} || \Delta_{\text{object}} \tag{19}$$

The specific authentication and authorization details for each service are outlined below:

1. The smart disaster service management utilizes an ECC-128-bit key for authentication.
2. The smart weather service utilizes an ECC-192-bit key for authentication.
3. The smart ambulance service requires an ECC-256-bit key for authentication.
4. For authorization to access the service, a trust value of 0.003 is used.

Algorithm 6 illustrates the implementation of service-level security mechanisms in decentralized interoperable services, ensuring that each service has its own unique access level security authentication and authorization policies to access collaborative tasks. This configuration empowers the users of each interoperable service to authenticate using their designated authentication mechanism. Consequently, when a user of a specific service seeks access to another interoperable service, they retain the capability to authenticate with the target interoperable service. This adaptive approach facilitates seamless interaction and interoperability among the various services within the ecosystem.

Table 11 displays the results of trust convergence as the number of request messages from the client of interoperable services to access other interoperable services gradually increases. We observed a pattern of slow trust convergence per second during the smart service interoperations of the third process when authentication requirements changed to ECC-256 bits. Fig. 22 shows the graphical representation of trust convergence when the number of the request message is increased. We also noticed the same pattern in Table 12 with slower trust convergence per second as compared with the previous result which reflects the importance of the programmable value of the trust. Fig. 23 shows the graphical representation of trust convergence when the number of IoT nodes is increased.

Smart contract implementation time is a critical metric within our proposed security framework, serving as an indicator of resource allocation and the effectiveness of security-related operations. This metric is particularly important for applications sensitive to delays, requiring rapid response times with minimal latency. With its robustness, adaptability, scalability, and compatibility, our framework ensures that smart contract execution minimally impacts the system's performance. By emphasizing streamlined implementation processes, we strike a balance between maintaining stringent security protocols, safeguarding application performance, and ensuring the system's scalability. This approach enables the system to uphold robust security measures while preserving operational efficiency and accommodating increasing user demand. The duration of smart contract implementation provides valuable insights into the framework's ability to efficiently execute security measures and hence, its comprehensive nature makes it a suitable metric for evaluating the security effectiveness and scalability of the proposed solution.

Table 11
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

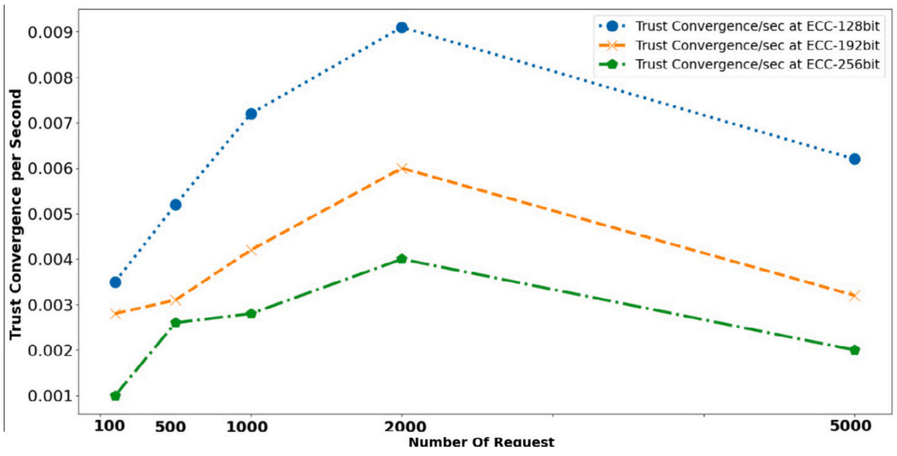

Fig. 22. Trust convergence by changing the number of requests.

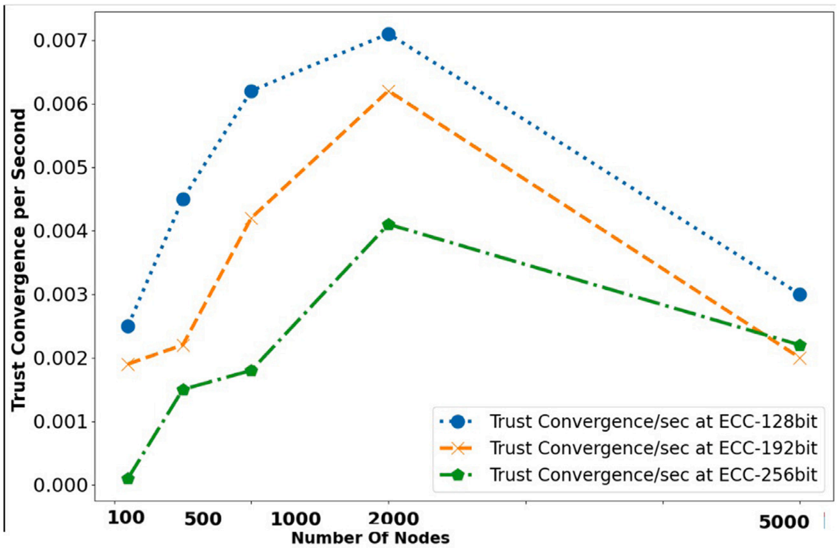

Fig. 23. Performance result with the increased number of nodes.

Table 12
Trust convergence by changing the number of nodes.

Number of nodes	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

10. CIA triad-based cybersecurity assessment

In this study, we propose an adaptive security framework designed to enhance the security and privacy of interoperable smart services within smart cities. This section evaluates the proposed framework's effectiveness in upholding the fundamental principles of cybersecurity as defined by the CIA triad: confidentiality, integrity, and availability.

1. **Confidentiality:** Confidentiality is one of the most critical elements of the framework of cybersecurity, which puts access constraints on information considered sensitive and hence keeps

Table 13
CIA traits and corresponding modules.

CIA trait	Module name	Technique
Confidentiality	- Message Encryption	Utilizes ECC (Elliptic Curve Cryptography) which relies on the algebraic structure of elliptic curves over finite fields. ECC provides strong encryption with smaller key sizes, enhancing security and efficiency.
	- Key and Session Management	Employs ECC for generating public and private keys, and ECDH (Elliptic Curve Diffie–Hellman) for securely establishing session keys. This approach ensures that session keys are exchanged securely and used for encrypting subsequent communications.
Integrity	- Local Blockchain Ledger	Uses a decentralized local ledger with consensus mechanisms (e.g., Proof of Work, Proof of Stake) to validate and record transactions. This ensures that all local transactions are transparent, verifiable, and immutable, preventing unauthorized alterations.
	- Global Blockchain Ledger	Employs a decentralized global ledger to synchronize data across all smart city services. The global ledger uses advanced consensus algorithms to maintain a tamper-proof record of all transactions, ensuring consistency and integrity across services.
	- Smart Contracts	Implements automated rules and protocols using scripts on the blockchain that execute actions based on predefined conditions. These contracts ensure that all operations follow the agreed-upon rules, preserving the integrity of interactions between services.
Availability	- SDN Controller	Provides dynamic and efficient network management, ensuring high availability by automatically routing traffic and managing network resources to prevent bottlenecks and failures. Utilizes algorithms for load balancing, fault tolerance, and traffic optimization.
	- Adaptive Security Policies	Continuously adjusts security measures in response to evolving threats and network conditions. Employs machine learning algorithms to predict and mitigate potential disruptions, ensuring that services remain available and resilient against attacks or failures.

it from unauthorized entities (Al-Muhtadi et al., 2021). The proposed framework employs multiple encryption methods, including Elliptic Curve Cryptography (ECC), to secure data transmission between services. Three key lengths will be used with ECC, 128-bit, 192-bit, and 256-bit, for different services, ensuring a robust level of confidentiality. For instance, ECC-128 is applied in smart disaster management services, while ECC-192 is used in smart weather services and ECC-256 in smart ambulance services.

2. **Integrity:** Integrity implies that the data does not change in the process during a principal activity of transmission and storage (Huang, Fan, et al., 2020). Data integrity is maintained through the use of blockchain technology, which is a fundamental component of the framework. As a decentralized ledger, blockchain ensures that all transactions are transparent and immutable, meaning any attempt to alter data would be immediately detectable and preventable (Singh, Hosen, & Yoon, 2021). The rules and protocols governing data exchange are enforced through smart contracts. These contracts in the proposed framework ensure that data integrity is upheld across different smart services by defining strict conditions for data transactions. Any deviation from these conditions would invalidate the transaction, thereby preserving the integrity of the data.
3. **Availability:** Availability means that smart city services are available whenever needed and ensures the continuous running of the normal system without any disruptions (Jararweh, Otoum, & Al Ridhawi, 2020). In the proposed framework, SDN, as a kind of network management, makes management dynamic and efficient. Decentralized management and communication between smart services ensure high availability and resilience against failure from the SDN controller. Adaptive security policies on top decide, catching up with evolving threats, and hence make services available even under adverse conditions.

Adaptive security policies are crucial for ensuring the CIA triads of smart city infrastructures. These policies are enforced through smart contracts, which dynamically adjust security measures in response to the evolving threat landscape. Our adaptive assessment experiments have demonstrated the proposed framework's capability to detect and respond to security threats effectively. Additionally, the framework's ability to further adapt these policies enhances system resilience and overall effectiveness. Table 13 details the CIA traits and outlines the encryption and decryption techniques employed to prevent data breaches, demonstrating the robustness of our proposed framework. This comprehensive analysis underscores the significant contribution of our work to the cybersecurity landscape in smart cities.

11. Conclusion

Smart contract-based systems offer a unique approach to adaptive security, dynamically adjusting security policies and measures to changing conditions and emerging threats. Our evaluation highlights the scalability of the proposed security framework, facilitated by crucial components such as the SDN controller and Blockchain memory pool. The SDN controller enables centralized management and efficient communication between smart services, while the Blockchain memory pool ensures the integrity and immutability of security-related transactions and data. Looking forward, integrating a privacy management module during smart service interoperation could enhance user privacy, data confidentiality, and compliance with privacy regulations. However, it is important to acknowledge the limitations of our framework. The emergency response use case, while syntactically designed, may not fully capture all possible scenarios and variations in real-world implementations. Performance evaluation results may vary based on the specific design of the emergency response system, affecting metrics such as throughput and contract execution time. Additionally, external factors such as the environment of simulating tools and system resources can impact experiment outcomes. Therefore, careful consideration of system design and resource allocation is crucial for ensuring optimal performance and effectiveness when implementing our proposed security framework.

CRediT authorship contribution statement

Shahbaz Siddiqui: Writing – original draft, Visualization, Validation, Resources, Methodology, Formal analysis, Data curation, Conceptualization. **Sufian Hameed:** Writing – review & editing, Supervision, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Syed Attique Shah:** Writing – review & editing, Writing – original draft, Validation, Supervision, Resources, Project administration, Methodology, Investigation, Formal analysis. **Junaid Arshad:** Writing – review & editing. **Yussuf Ahmed:** Writing – review & editing. **Dirk Draheim:** Writing – review & editing, Validation, Supervision, Project administration, Methodology.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: SYED ATTIQUE SHAH reports was provided by Birmingham City University. SYED ATTIQUE SHAH reports a relationship with Birmingham City University that includes: employment. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

Agbaje, P., Anjum, A., Mitra, A., Oseghale, E., Bloom, G., & Olufowobi, H. (2022). Survey of interoperability challenges in the internet of vehicles. *IEEE Transactions on Intelligent Transportation Systems*, *23*(12), 22838–22861.

Al Batayneh, R. M., Taleb, N., Said, R. A., Alshurideh, M. T., Ghazal, T. M., & Alzoubi, H. M. (2021). 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* (pp. 235–247). Springer.

Al-Muhtadi, J., Saleem, K., Al-Rabiaah, S., Imran, M., Gawanmeh, A., & Rodrigues, J. J. (2021). A lightweight cyber security framework with context-awareness for pervasive computing environments. *Sustainable Cities and Society*, *66*, Article 102610.

Ali, G., Ahmad, N., Cao, Y., Khan, S., Cruickshank, H., Qazi, E. A., et al. (2020). xDBAuth: Blockchain based cross domain authentication and authorization framework for Internet of Things. *IEEE Access*, *8*, 58800–58816.

Ali, T., Irfan, M., Alwadie, A. S., & Glowacz, A. (2020). IoT-based smart waste bin monitoring and municipal solid waste management system for smart cities. *Arabian Journal for Science and Engineering*, *45*(12), 10185–10198.

Alsaeedi, M., Mohamad, M. M., & Al-Roubaiey, A. A. (2019). Toward adaptive and scalable OpenFlow-SDN flow control: A survey. *IEEE Access*, *7*, 107346–107379.

Alshboul, Y., Bsoul, A. A. R., Al Zamil, M., & Samarah, S. (2021). Cybersecurity of smart home systems: Sensor identity protection. *Journal of Network and Systems Management*, *29*(3), 1–27.

Ante, L. (2021). Smart contracts on the blockchain–A bibliometric analysis and review. *Telematics and Informatics*, *57*, Article 101519.

Antonios, P., Konstantinos, K., & Christos, G. (2023). A systematic review on semantic interoperability in the IoE-enabled smart cities. *Internet of Things*, Article 100754.

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

Asif, M., Aziz, Z., Bin Ahmad, M., Khalid, A., Waris, H. A., & Gilani, A. (2022). Blockchain-based authentication and trust management mechanism for smart cities. *Sensors*, *22*(7), 2604.

Aujla, G. S., Singh, M., Bose, A., Kumar, N., Han, G., & Buyya, R. (2020). BlockSDN: Blockchain-as-a-service for software defined networking in smart city applications. *IEEE Network*, *34*(2), 83–91.

Aujla, G. S., Singh, A., Singh, M., Sharma, S., Kumar, N., & Choo, K. K. R. (2020). BloCkEd: Blockchain-based secure data processing framework in edge envisioned V2X environment. *IEEE Transactions on Vehicular Technology*, *69*(6), 5850–5863.

Balcerzak, A. P., Nica, E., Rogalska, E., Poliak, M., Kliештik, T., & Sabie, O. M. (2022). Blockchain technology and smart contracts in decentralized governance systems. *Administrative Sciences*, *12*(3), 96.

Banerjee, S., Bera, B., Das, A. K., Chattopadhyay, S., Khan, M. K., & Rodrigues, J. J. (2021). Private blockchain-envisioned multi-authority CP-ABE-based user access control scheme in IIoT. *Computer Communications*, *169*, 99–113.

Bannour, F., Souihi, S., & Mellouk, A. (2017). Distributed SDN control: Survey, taxonomy, and challenges. *IEEE Communications Surveys & Tutorials*, *20*(1), 333–354.

Bao, F., Chen, R., & Guo, J. (2013). 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* (pp. 1–7). IEEE.

Basheer, H., & Itani, M. (2023). Zero touch in fog, IoT, and MANET for enhanced smart city applications: A survey. *Future Cities and Environment*, *9*(1), 5.

Bellavista, P., Esposito, C., Foschini, L., Giannelli, C., Mazzocca, N., & Montanari, R. (2021). Interoperable blockchains for highly-integrated supply chains in collaborative manufacturing. *Sensors*, *21*(15), 4955.

Bello, O., & Zeadally, S. (2019). Toward efficient smartification of the Internet of Things (IoT) services. *Future Generation Computer Systems*, *92*, 663–673.

Benkhaled, S., Hemam, M., & Maimour, M. (2022). SDN-based approaches for heterogeneity and interoperability in Internet of Things: An overview. In *Distributed sensing and intelligent systems: Proceedings of ICDSIS 2020* (pp. 489–499). Springer.

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The Semantic Web – A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities. *Scientific American*, *17 May*.

Bhushan, B., Khamparia, A., Sagayam, K. M., Sharma, S. K., Ahad, M. A., & Debnath, N. C. (2020). Blockchain for smart cities: A review of architectures, integration trends and future research directions. *Sustainable Cities and Society*, *61*, Article 102360.

Bhushan, B., Sahoo, C., Sinha, P., & Khamparia, A. (2021). Unification of Blockchain and Internet of Things (BIoT): requirements, working model, challenges and future directions. *Wireless Networks*, *27*, 55–90.

Buldas, A., Draheim, D., Gault, M., Laanoja, R., Nagumo, T., Saarepera, M., et al. (2022). An ultra-scalable blockchain platform for universal asset tokenization: Design and implementation. *IEEE Access*, *10*, 77284–77322.

Buldas, A., Draheim, D., Gault, M., & Saarepera, M. (2022). Towards a foundation of Web3. In *CCIS*: *vol. 1688*, *Proceedings of FDSE'2022 – the 9th international conference on future data and security engineering* (pp. 3–18). Berlin Heidelberg New York: Springer.

Chen, X., Deng, Y., Ding, H., Qu, G., Zhang, H., Li, P., et al. (2023). Vehicle as a service (VaaS): Leverage vehicles to build service networks and capabilities for smart cities. arXiv preprint arXiv:2304.11397.

Choi, J. (2022). Enablers and inhibitors of smart city service adoption: A dual-factor approach based on the technology acceptance model. *Telematics and Informatics*, *75*, Article 101911.

Costa, D. G., Peixoto, J. P. J., Jesus, T. C., Portugal, P., Vasques, F., Rangel, E., et al. (2022). A survey of emergencies management systems in smart cities. *IEEE Access*, *10*, 61843–61872.

Dua, A., Kumar, N., Das, A. K., & Susilo, W. (2017). Secure message communication protocol among vehicles in smart city. *IEEE Transactions on Vehicular Technology*, *67*(5), 4359–4373.

Edelman, G. (2022). Paradise at the cryto arcade. *Wired*, *June*.

EL-Garoui, L., Pierre, S., & Chamberland, S. (2020). A new SDN-based routing protocol for improving delay in smart city environments. *Smart Cities*, *3*(3), 1004–1021.

Esber, J., & Kominers, S. D. (2022). Why build in Web3. *Harvard Business Review*, *16 May*, https://hbr.org/2022/05/why-build-in-web3.

Galluccio, L., Milardo, S., Morabito, G., & Palazzo, S. (2015). SDN-WISE: Design, prototyping and experimentation of a stateful SDN solution for WIreless SEnsor networks. In *2015 IEEE conference on computer communications* (pp. 513–521). IEEE.

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

Guvenc, I., Koohifar, F., Singh, S., Sichitiu, M. L., & Matolak, D. (2018). Detection, tracking, and interdiction for amateur drones. *IEEE Communications Magazine*, *56*(4), 75–81.

Huang, P., Fan, K., Yang, H., Zhang, K., Li, H., & Yang, Y. (2020). A collaborative auditing blockchain for trustworthy data integrity in cloud storage system. *IEEE Access*, *8*, 94780–94794.

Huang, J., Fang, D., Qian, Y., & Hu, R. Q. (2020). Recent advances and challenges in security and privacy for V2X communications. *IEEE Open Journal of Vehicular Technology*, *1*, 244–266.

Hui, T. K., Sherratt, R. S., & Sánchez, D. D. (2017). Major requirements for building smart homes in smart cities based on Internet of Things technologies. *Future Generation Computer Systems*, *76*, 358–369.

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

Iqbal, W., Abbas, H., Daneshmand, M., Rauf, B., & Bangash, Y. A. (2020). An in-depth analysis of IoT security requirements, challenges, and their countermeasures via software-defined security. *IEEE Internet of Things Journal*, *7*(10), 10250–10276.

Islam, M. J., Rahman, A., Kabir, S., Karim, M. R., Acharjee, U. K., Nasir, M. K., et al. (2021). Blockchain-SDN-based energy-aware and distributed secure architecture for IoT in smart cities. *IEEE Internet of Things Journal*, *9*(5), 3850–3864.

Ismagilova, E., Hughes, L., Rana, N. P., & Dwivedi, Y. K. (2022). Security, privacy and risks within smart cities: Literature review and development of a smart city interaction framework. *Information Systems Frontiers*, *24*(2), 393–414.

Jararweh, Y., Otoum, S., & Al Ridhawi, I. (2020). Trustworthy and sustainable smart city services at the edge. *Sustainable Cities and Society*, *62*, Article 102394.

Javed, A. R., Shahzad, F., ur Rehman, S., Zikria, Y. B., Razzak, I., Jalil, Z., et al. (2022). Future smart cities: Requirements, emerging technologies, applications, challenges, and future aspects. *Cities*, *129*, Article 103794.

Jin, L., & Parrott, K. (2022). Web3 is our chance to make a better internet. *Harvard Business Review*, *10 May*, https://hbr.org/2022/05/web3-is-our-chance-to-make-a-better-internet.

Joo, Y. M. (2023). Developmentalist smart cities? The cases of Singapore and Seoul. *International Journal of Urban Sciences*, *27*(sup1), 164–182.

Karumba, S., Jurdak, R., Kanhere, S., & Sethuvenkatraman, S. (2023). BAILIF: A blockchain agnostic interoperability framework. In *Proceedings of the 5th IEEE international conference on blockchain and cryptocurrency*. Institute of Electrical and Electronics Engineers Inc..

Kashef, M., Visvizi, A., & Troisi, O. (2021). Smart city as a smart service system: Human-computer interaction and smart city surveillance systems. *Computers in Human Behavior*, *124*, Article 106923.

Keoh, S. L., Kumar, S. S., & Tschofenig, H. (2014). Securing the Internet of Things: A standardization perspective. *IEEE Internet of Things Journal*, *1*(3), 265–275.

Khan, S. N., Loukil, F., Ghedira-Guegan, C., Benkhelifa, E., & Bani-Hani, A. (2021). Blockchain smart contracts: Applications, challenges, and future trends. *Peer-to-peer Networking and Applications*, *14*, 2901–2925.

Kharche, S., & Dere, P. (2022). Interoperability issues and challenges in 6G networks. *Journal of Mobile Multimedia*, *18*(5), 1445–1470.

Kirimtat, A., Krejcar, O., Kertesz, A., & Tasgetiren, M. F. (2020). Future trends and current state of smart city concepts: A survey. *IEEE Access*, *8*, 86448–86467.

Knowles Flanagan, S. A. (2022). *Cooperative connected intelligent vehicles and infrastructure for road safety applications* (Ph.D. thesis), Aston University.

Koo, J., & Kim, Y. G. (2021). Interoperability requirements for a smart city. In *Proceedings of the 36th annual ACM symposium on applied computing* (pp. 690–698).

Kozhevnikov, S., Svitek, M., & Skobelev, P. (2022). 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* (pp. 4–9).

Kumar, H., Singh, M. K., Gupta, M., & Madaan, J. (2020). Moving towards smart cities: Solutions that lead to the smart city transformation framework. *Technological Forecasting and Social Change*, *153*, Article 119281.

Latif, S. A., Wen, F. B. X., Iwendi, C., Li-li, F. W., Mohsin, S. M., Han, Z., et al. (2022). AI-empowered, blockchain and SDN integrated security architecture for IoT network of cyber physical systems. *Computer Communications*, *181*, 274–283.

Li, T., Chen, J., & Fu, H. (2019). Application scenarios based on SDN: an overview. *Journal of Physics: Conference Series*, *1187*, Article 052067.

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

Macrinici, D., Cartofeanu, C., & Gao, S. (2018). Smart contract applications within blockchain technology: A systematic mapping study. *Telematics and Informatics*, *35*(8), 2337–2354.

Makhdoom, I., Zhou, I., Abolhasan, M., Lipman, J., & Ni, W. (2020). PrivySharing: A blockchain-based framework for privacy-preserving and secure data sharing in smart cities. *Computers & Security*, *88*, Article 101653.

Mamatas, L., Demiroglou, V., Kalafatidis, S., Skaperas, S., & Tsaoussidis, V. (2023). Protocol-adaptive strategies for wireless mesh smart city networks. *IEEE Network*, *37*(2), 136–143.

Marshoodulla, S. Z., & Saha, G. (2022). Data heterogeneity handling in SDN-based IoT infrastructure. *NeuroQuantology*, *20*(14), 805–812.

Medhane, D. V., Sangaiah, A. K., Hossain, M. S., Muhammad, G., & Wang, J. (2020). Blockchain-enabled distributed security framework for next-generation IoT: An edge cloud and software-defined network-integrated approach. *IEEE Internet of Things Journal*, *7*(7), 6143–6149.

Meijer, A., & Bolívar, M. P. R. (2016). Governing the smart city: a review of the literature on smart urban governance. *International Review of Administrative Sciences*, *82*(2), 392–408.

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

Mora, H., Mendoza-Tello, J. C., Varela-Guzmán, E. G., & Szymanski, J. (2021). Blockchain technologies to address smart city and society challenges. *Computers in Human Behavior*, *122*, Article 106854.

Mostafaei, H., & Menth, M. (2018). Software-defined wireless sensor networks: A survey. *Journal of Network and Computer Applications*, *119*, 42–56.

Motlagh, N. H., Taleb, T., & Arouk, O. (2016). Low-altitude unmanned aerial vehicles-based Internet of Things services: Comprehensive survey and future perspectives. *IEEE Internet of Things Journal*, *3*(6), 899–922.

Mrabet, H., Belguith, S., Alhomoud, A., & Jemai, A. (2020). A survey of IoT security based on a layered architecture of sensing and data analysis. *Sensors*, *20*(13), 3625.

Msahli, M., Labiod, H., & Ampt, G. (2019). Security interoperability for cooperative ITS: Architecture and validation. In *2019 10th IFIP international conference on new technologies, mobility and security* (pp. 1–6). IEEE.

MultiChain (2023a). https://www.multichain.com/. (Last accessed 07 June 2023).

MultiChain (2023b). https://github.com/MultiChain/multichain-api-libraries/. (Last accessed 07 June 2023).

Nguyen, D. C., Pathirana, P. N., Ding, M., & Seneviratne, A. (2020). Blockchain for 5G and beyond networks: A state of the art survey. *Journal of Network and Computer Applications*, *166*, Article 102693.

Ogrodowczyk, Ł., Belter, B., & LeClerc, M. (2016). IoT ecosystem over programmable SDN infrastructure for smart city applications. In *2016 fifth European workshop on software-defined networks* (pp. 49–51). IEEE.

Oikonomou, G., Duquennoy, S., Elsts, A., Eriksson, J., Tanaka, Y., & Tsiftes, N. (2022). The Contiki-NG open source operating system for next generation IoT devices. *SoftwareX*, *18*, Article 101089.

Pereira, G. V., Parycek, P., Falco, E., & Kleinhans, R. (2018). Smart governance in the context of smart cities: A literature review. *Information Polity*, *23*(2), 143–162.

Polkadot (2023). https://polkadot.network/. (Last accessed 07 June 2023).

Rahman, M. S., Chamikara, M., Khalil, I., & Bouras, A. (2022). Blockchain-of-blockchains: An interoperable blockchain platform for ensuring IoT data integrity in smart city. *Journal of Industrial Information Integration*, *30*, Article 100408.

Rana, B., & Singh, Y. (2023). Interoperable agile IoT. In *Agile software development: Trends, challenges and applications* (pp. 51–70). Wiley Online Library.

Rao, P. M., & Deebak, B. (2022). Security and privacy issues in smart cities/industries: Technologies, applications, and challenges. *Journal of Ambient Intelligence and Humanized Computing*, 1–37.

Rathee, G., Kumar, A., Kerrache, C. A., & Iqbal, R. (2022). A trust-based mechanism for drones in smart cities. *IET Smart Cities*.

Rathore, M. M., Attique Shah, S., Awad, A., Shukla, D., Vimal, S., & Paul, A. (2021). A cyber-physical system and graph-based approach for transportation management in smart cities. *Sustainability*, *13*(14), 7606.

Rathore, M. M., Paul, A., Rho, S., Khan, M., Vimal, S., & Shah, S. A. (2021). Smart traffic control: Identifying driving-violations using fog devices with vehicular cameras in smart cities. *Sustainable Cities and Society*, *71*, Article 102986.

Reegu, F. A., Abas, H., Jabbari, A., Akmam, R., Uddin, M., Wu, C. M., et al. (2022). Interoperability requirements for blockchain-enabled electronic health records in healthcare: A systematic review and open research challenges. *Security and Communication Networks*, *2022*.

Salman, O., Elhajj, I. H., Kayssi, A., & Chehab, A. (2016). SDN controllers: A comparative study. In *2016 18th mediterranean electrotechnical conference* (pp. 1–6). IEEE.

SemnaticWeb (2023). https://www.w3.org/standards/semanticweb/. (Last accessed 07 June 2023).

Shah, S. A., Seker, D. Z., Hameed, S., & Draheim, D. (2019). The rising role of big data analytics and IoT in disaster management: recent advances, taxonomy and prospects. *IEEE Access*, *7*, 54595–54614.

Shah, S. A., Seker, D. Z., Rathore, M. M., Hameed, S., Yahia, S. B., & Draheim, D. (2019). Towards disaster resilient smart cities: Can internet of things and big data analytics be the game changers? *IEEE Access*, *7*, 91885–91903.

Shamsudheen, S., Karthik, G., Anoop, A., & Gobinathan, P. (2023). Internet-of-things in emergency services: Architecture, applications, and research challenges. In *2023 1st international conference on advanced innovations in smart cities* (pp. 1–6). IEEE.

Sharma, P. K., Singh, S., Jeong, Y. S., & Park, J. H. (2017). Distblocknet: A distributed blockchains-based secure SDN architecture for IoT networks. *IEEE Communications Magazine*, *55*(9), 78–85.

Siddiqui, S., Hameed, S., Shah, S. A., Khan, A. K., & Aneiba, A. (2023). Smart contract-based security architecture for collaborative services in municipal smart cities. *Journal of Systems Architecture*, *135*, Article 102802.

Singh, S., Hosen, A. S., & Yoon, B. (2021). Blockchain security attacks, challenges, and solutions for the future distributed IoT network. *IEEE Access*, *9*, 13938–13959.

Singh, S., Sharma, P. K., Yoon, B., Shojafar, M., Cho, G. H., & Ra, I. H. (2020). Convergence of blockchain and artificial intelligence in IoT network for the sustainable smart city. *Sustainable Cities and Society*, *63*, Article 102364.

Sookhak, M., Tang, H., He, Y., & Yu, F. R. (2018). Security and privacy of smart cities: a survey, research issues and challenges. *IEEE Communications Surveys & Tutorials*, *21*(2), 1718–1743.

Stackpole, T. (2022). What is web3? *Harvard Business Review*, *10 May*, https://hbr.org/2022/05/what-is-web3.

Tang, B., Kang, H., Fan, J., Li, Q., & Sandhu, R. (2019). 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* (pp. 83–92).

Thomson, C., Romdhani, I., Al-Dubai, A., Qasem, M., Ghaleb, B., & Wadhaj, I. (2016). *Cooja simulator manual, version 1.0*. Edinburgh Napier University, https://www.napier.ac.uk/~/media/worktribe/output-299955/cooja-simulator-manual.pdf. (Last accessed 08 June 2023).

Tosic, M., Coelho, F. A., Nouwt, B., Rua, D. E., Tomcic, A., & Pesic, S. (2022). Towards a cross-domain semantically interoperable ecosystem. In *Proceedings of the fifteenth ACM international conference on web search and data mining* (pp. 1640–1641).

Ullah, F., Wang, J., Farhan, M., Jabbar, S., Naseer, M. K., & Asif, M. (2020). LSA based smart assessment methodology for SDN infrastructure in IoT environment. *International Journal of Parallel Programming*, *48*, 162–177.

Viale Pereira, G., Cunha, M. A., Lampoltshammer, T. J., Parycek, P., & Testa, M. G. (2017). Increasing collaboration and participation in smart city governance: A cross-case analysis of smart city initiatives. *Information Technology for Development*, *23*(3), 526–553.

Villarreal, E. R. D., García-Alonso, J., Moguel, E., & Alegría, J. A. H. (2023). Blockchain for healthcare management systems: A survey on interoperability and security. *IEEE Access*, *11*, 5629–5652.

Wang, G., Wang, Q., & Chen, S. (2023). Exploring blockchains interoperability: A systematic survey. *ACM Computing Surveys*.

Xie, J., Tang, H., Huang, T., Yu, F. R., Xie, R., Liu, J., et al. (2019). A survey of blockchain technology applied to smart cities: Research issues and challenges. *IEEE Communications Surveys & Tutorials*, *21*(3), 2794–2830.

Xu, R., Chen, Y., Blasch, E., & Chen, G. (2018). Blendcac: A smart contract enabled decentralized capability-based access control mechanism for the IoT. *Computers*, *7*(3), 39.

Yazdinejad, A., Parizi, R. M., Dehghantanha, A., Zhang, Q., & Choo, K. K. R. (2020). An energy-efficient SDN controller architecture for IoT networks with blockchain-based security. *IEEE Transactions on Services Computing*, *13*(4), 625–638.

Žarko, I. P., Mueller, S., Płociennik, M., Rajtar, T., Jacoby, M., Pardi, M., et al. (2019). The symbIoTe solution for semantic and syntactic interoperability of cloud-based IoT platforms. In *2019 global IoT summit* (pp. 1–6). IEEE.

Zhou, J., Jiang, H., Wu, J., Wu, L., Zhu, C., & Li, W. (2016). SDN-based application framework for wireless sensor and actor networks. *IEEE Access*, *4*, 1583–1594.

Zikria, Y. B., Afzal, M. K., Ishmanov, F., Kim, S. W., & Yu, H. (2018). A survey on routing protocols supported by the Contiki Internet of Things operating system. *Future Generation Computer Systems*, *82*, 200–219.

Zou, W., Lo, D., Kochhar, P. S., Le, X. B. D., Xia, X., Feng, Y., et al. (2019). Smart contract development: Challenges and opportunities. *IEEE Transactions on Software Engineering*, *47*(10), 2084–2106.

Zubaydi, H. D., Varga, P., & Molnár, S. (2023). Leveraging blockchain technology for ensuring security and privacy aspects in Internet of Things: A systematic literature review. *Sensors*, *23*(2), 788.