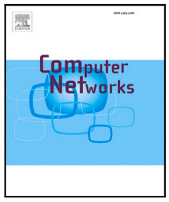
Computer Networks 241 (2024) 110197

Contents lists available at ScienceDirect

Computer Networks

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



Implementing zero trust security with dual fuzzy methodology for trust-aware authentication and task offloading in Multi-access Edge Computing

Belal Ali [a,*,1], Mark A. Gregory [a,2], Shuo Li [a,1], Omar Amjad Dib [b,1]

[a] School of Engineering, RMIT University, Melbourne, Australia
[b] School of Engineering, KCST University, Doha, Kuwait

ARTICLE INFO

Keywords:
Multi-access edge computing
Security
Task offloading
Zero trust
Trust-aware

ABSTRACT

This paper proposes an efficient trust-aware authentication and task offloading scheme for Multi-Access Edge Computing (MEC) using the Zero Trust Security (ZTS) principles. The proposed method uses a dual fuzzy logic system to evaluate the trustworthiness of edge servers. Devices connected to the edge servers are authenticated using identity, biometrics and Physical Unclonable Function (PUF) measures. After authentication, tasks can be offloaded from the devices to the most trustworthy edge server. The proposed scheme also considers the resource constraints of the edge servers and aims to minimise the overall task completion time. The experimental results show that the proposed scheme outperforms existing schemes regarding authentication accuracy, task completion time, and energy consumption.

1. Introduction

Multi-Access Edge Computing (MEC) has become a paradigm for an edge solution that supports real-time and low-latency applications [1]. MEC enables computation and storage capabilities at the network edge, reducing the end-to-end delay and improving User Equipment (UE) Quality of Service (QoS) [2]. However, security and trust issues are significant challenges in MEC environments, especially in the authentication and task offloading processes [3]. In this paper, we review the related literature on trust-aware authentication and task offloading in MEC and propose an efficient scheme using the dual fuzzy method.

The Zero Trust Security (ZTS) framework [4] is a security approach that assumes that all devices and UE are untrusted until proven otherwise. In MEC, all edge servers and users must be authenticated and verified before accessing or performing tasks within the MEC network. In a ZTS framework, every request for access or task execution is treated as coming from an untrusted source. The edge servers and UE must provide proof of their identity and authenticity. This proof may come from various sources, such as biometric authentication, UE characteristics, network conditions, and security mechanisms. The ZTS framework aims to minimise the attack surface and reduce the risk of security breaches by verifying edge servers and UE before access is provided to supporting systems [5]. The ZTS framework can enhance MEC security by preventing unauthorised access, detecting and responding to security threats, and ensuring that sensitive data is protected [6].

To establish trusted MEC management platforms, blockchain technology has been introduced to record and audit the trust of MEC entities in the network [7]. Blockchain is a promising technology that can be used to increase security in distributed MEC networks [8]. Blockchain is a decentralised digital ledger that records transactions in one or more journals that can be stored in the cloud or on the distributed systems participating in the transactions. It is designed to be secure, transparent, and tamper-proof. Each blockchain block contains several transactions and a unique cryptographic signature called a “hash” that links it to the previous block to enforce privacy-preservation. Once a block is added to the blockchain, its information cannot be altered, deleted, or tampered with [9]. Blockchain is suitable for large-scale MEC networks to enhance the QoS [10].

In the context of trust-aware task offloading in MEC, blockchain can establish and maintain trust between edge servers and the central management or orchestrator layer. It can also help mitigate the risk of processing malicious tasks. It provides a secure and transparent way to record and maintain trust information about edge servers and devices, and this information can be used to make informed decisions about task

* Corresponding author.
E-mail addresses: s3775159@student.rmit.edu.au (B. Ali), mark.gregory@rmit.edu.au (M.A. Gregory), shuo.li2@rmit.edu.au (S. Li), 211489@student.kcst.edu.kw (O.A. Dib).
[1] Member, IEEE.
[2] Senior Member, IEEE.

https://doi.org/10.1016/j.comnet.2024.110197
Received 25 August 2023; Received in revised form 12 January 2024; Accepted 16 January 2024
Available online 2 February 2024

Table 1
List of main abbreviations.

Abbreviation	Full name
ACP	Access Control Policy
AMF	Access Management Function
API	Application Programmable Interface
eNodeB	Evolved Node Base-station
MEC	Multi-access Edge Computing
NFs	Network Functions
NRF	Network Resource Functions
PAP	Policy Administration Point
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PIP	Policy Information Point
PUF	Physical Unclonable Function
QoS	Quality of Service
RAM	Resource Allocation Manager
RAN	Radio Access Network
RSU	Road Side Unit
SARSA	State-Action-Reward-State-Action
UDM	Unified Data Management
UPF	User Plane Function
ZTO	Zero Trust Orchestrator
ZTS	Zero Trust Security

offloading [7]. This helps to ensure that only trusted and verified edge servers can process UE tasks, reducing the risk of security breaches and improving MEC deployment efficiency.

The rest of this paper is organised as follows. Section 2 introduces the ZTS framework model for MEC. Section 3 presents the proposed approach. Section 4 describes the experimental results and provides a comparative analysis. Based on our findings, Section 5 discusses the open issues and concludes the paper. Table 1 summarises the abbreviations used in this paper.

1.1. Motivation and contribution

We propose a trust-aware task offloading and UE authentication framework to address the MEC server trust, high availability, and seamless task offloading challenges. First, UE are validated, registered, and then authenticated against the Zero Trust Orchestrator (ZTO) to ensure that only legitimate UE can submit requests. Then, the ZTO computes the trust value for the edge servers and stores it in the blockchain. Finally, the task offloading process is performed based on the estimated trust value. The proposed framework was simulated, and the results and analysis are provided. The main contributions of this article are summarised as follows.

- A new ZTO component is introduced at the MEC host level. It validates the UE identity before tasks are allocated and processed. The ZTO component is aligned with the zero trust architecture requirements for a multi-layer cybersecurity approach, including defence-in-depth controls [11]. The UE are authenticated using identity, biometrics, and Physical Unclonable Function (PUF) [12]. A bio-key is generated from the user's fingerprint to ensure that legitimate users can be identified.
- A trust-aware task offloading framework is proposed that utilises blockchain to validate the trustworthiness of UE and MEC servers. A ZTO was developed that uses a trusted communication channel to the blockchain journal and is responsible for estimating the server trust values, making task offloading decisions, recording the transactions and providing continuous monitoring.
- A decision execution method that estimates trust values for MEC servers and continuously assesses the trust values based on the dual fuzzy method was developed. We validated the trust value integrity, a composite of four trust measures (direct, recommendation, indirect, and transmission trust) and then stored the results in the blockchain. A trust-based task offloading was performed with the State-Action-Reward-State-Action (SARSA) algorithm.
- Selected Proof-of-Concept (PoC) experiments in an MEC environment were carried out to validate the proposed approach. Scenarios were created that included UE requests with diverse requirements in a high-mobility environment. The proposed framework aims to improve task offloading performance with enhanced security.

1.2. Related work and research gaps

In this section, we focus on the primary research works relevant to the paper. Numerous trust architectures, frameworks, and models have been suggested to create effective trust strategies and task offloading in an MEC environment. However, existing approaches that integrate zero trust principles and MEC, as described in Table 2, generally do not consider the use of blockchain and various trust levels.

In [13], a lightweight fuzzy collaborative trust evaluation model was developed to cater for the dynamic and heterogeneous edge computing environment and the efficient collaborative scheduling of computing power resources in the MEC network. The proposal addressed only the MEC trust aspect but can scale to consider the UE authentication. In [4], a Zero Trust concept was developed, focusing on secure access by enforcing identity verification and location-based authentication to ensure seamless service continuity across the MEC deployment. The model used a lightweight PRESENT algorithm with an 80-bit key size and 64-bit block size and processed 32 cycles per block, which reduced the authentication time. However, the proposal addressed only one aspect of the Zero Trust guiding principles. In [14], a three-tier trust evaluation framework (identity, capability and behaviour trust) was proposed based on a reputation trust evaluation mechanism to ensure service interactions are qualified, capable, and reliable for the edge servers that join the MEC deployment. The same MEC server could be selected multiple times due to the best bidding value.

In [15], a Neuro-Fuzzy based on blockchain was developed to adapt the different elements' security levels in an MEC environment. The authors demonstrated the possibility of generating a security index to measure protection and attack prediction by combining neural networks with fuzzy logic. However, illegitimate users submitting malicious packets to edge servers could impact trust values. Also, evaluating recommendation trust based on similarity measures may be complex when comparing vectors of different sizes.

In [16], a Reputation-Based Trust Evaluation and Management (RTEM) system that incorporates identity, capability, and behaviour trust was developed. The three trusts are computed sequentially. Task interaction is executed only when all three trust values are satisfied. Identity trust is based on the device's ID number, which, although not a strong security credential, was used for device authentication. Capability trust depends on resources such as CPU, memory, and online time, while behaviour trust is derived from the device's historical behaviour. In identity trust, devices are authenticated solely based on their ID numbers, which is a less secure method. This approach poses security risks, as identity forging and guessing become relatively simple with advanced technologies. Although identity trust is straightforward, it still allows chances for illegitimate users. The estimation of each trust is performed one after the other for each device, and if any trust value is not satisfied, the evaluation is repeated.

In [9], a reputation system based on blockchain and CPU allocation was presented using a reinforcement learning algorithm. This work solves two direct attacks: selfish edge attacks and faked service record attacks. The proposed reinforcement learning algorithm involves a task offloading process that securely uses the signature verification and Diffie–Hellman algorithm. The proposal is vulnerable to other attacks, e.g., data tampering, private data leakage and replication. In [8], three trust computations were involved: entity trust, data trust, and privacy trust. The trust computations use a set of attributes. The transactions were validated using the signature and the trust values. The hashing of UE credentials uses the traditional SHA-256 algorithm. SHA-256 is

Table 2
Proposal compared to the literature.

Reference no.	MEC & Blockchain integration	MEC trust evaluation	MEC resource mgmt	QoS	Privacy and security	Continuous monitoring	Authentication algorithm	Approach proposed
[13]		✔	✔		✔			Fuzzy Collaborative Trust Evaluation
[4]		✔			✔	✔	✔	Zero Trust Security
[14]		✔						An Active and Verifiable Trust Evaluation
[15]	✔	✔			✔			Neuro Fuzzy Systems based Blockchain
[16]		✔			✔	✔		A reputation-based trust evaluation system
[9]	✔	✔		✔	✔		✔	A Reinforcement Learning
[8]	✔	✔			✔		✔	A privacy preserving blockchain
[17]		✔	✔		✔		✔	A Reinforcement Learning
[18]			✔	✔		✔	✔	Identity Access Management
[11]	✔	✔	✔		✔		✔	Novel End-to-End Trusted Authentication
Proposed	✔	✔	✔	✔	✔	✔	✔	Dual Fuzzy Zero Trust Security

a conventional hashing that requires extensive computations. Hence, it is not suitable for lightweight edge devices. In [18], an authentication mechanism has been proposed that provides security by authenticating UE using two-factor authentication, i.e., password and smart card. The UE registers with a security credential, and while logging in, the edge server verifies the identity and password entered via a smart card. However, the central issue of this authentication approach was the misuse of a lost smart card. Adversaries can use one or more methods to work out the user's identity and password, e.g., brute force, guaranteeing login access. The need for authentication provisioning and trust evaluation is inevitable. In [11], an authentication mechanism based on a blockchain using asymmetric cryptography was developed. Elliptic curve cryptography is performed to encrypt data. SHA-256 hashing is used to secure the records in the blockchain. A UE registers and receives a digital signature, which is then used to verify the user's identity. The traditional SHA-256 hashing algorithm consumes significant computations and time, delaying task actions, especially when the MEC server is loaded with simultaneous task requests. As a result, it added complexity to the trust hashes validation. In [19], a multifaceted framework encompassing behaviour and price awareness was proposed. It considers resource challenges, employs advanced game-theoretic models, and showcases superior performance in edge-computing decision-making.

Trust-aware task offloading in MEC involves evaluating the trustworthiness of UEs and edge servers and making offloading decisions based on the trust level. However, there are several challenges related to achieving trust-aware task offloading in MEC environments, including:

- Trust evaluation. The trustworthiness of UE and MEC servers can be evaluated based on various factors, such as behaviour, usage patterns, and reputation. However, these factors may only sometimes provide a reliable measure of trust. Multiple factors, such as malicious attacks, false data injection, and privacy concerns, may affect their accuracy.
- Resource constraints. MEC servers have limited computational and storage resources, and their availability and capacity may vary over time. Thus, task offloading decisions need to consider the resource constraints of the MEC servers and balance the load among them.
- QoS requirements. Different tasks may have additional QoS requirements, such as latency, bandwidth, and energy consumption. Thus, task offloading decisions need to consider the QoS requirements of the tasks and choose the most suitable MEC server based on the available resources and trust level.
- Privacy and security. Trust-aware task offloading may involve sharing sensitive data and information between mobile devices and MEC servers. Thus, privacy and security concerns must be addressed to prevent unauthorised access, data breaches, and other malicious activities.

Addressing these challenges requires the development of efficient and effective trust-aware task offloading schemes that consider the dynamic and complex nature of MEC environments.

The proposed framework focuses on two critical aspects of MEC security: trust-aware authentication and task offloading. Trust-aware authentication ensures that only authorised UE can access the MEC resources. At the same time, task offloading enables the efficient and effective distribution of computation and storage resources between the MEC servers.

Our approach involves two stages. In the first stage, authentication, the UE identity uses unique biometric characteristics and a PUF to determine whether the user should be granted access to the MEC resources. The UE access requests should be verified and authorised on a case-by-case basis.

In the second stage, the edge servers' trust values are analysed using a fuzzy logic-based trust evaluation method to ensure task offloading efficiency in MEC and maintain a trustworthy relationship with the MEC servers.

2. Zero trust management model for MEC

This section describes the zero trust management architecture in MEC. The design of our proposed architecture can be seen in Fig. 2. It comprises a network model, UE registration and authentication, edge trust validation in blockchain and trust-aware task offloading.

2.1. Simplified reference architecture

The proposed model is constructed in three layers: UE, edge and cloud. The layers and their capabilities are discussed in [20]. The layer descriptions are:

- *UE layer.* UE integrate wireless or mobile technologies. This layer connects to the nearest access point, "eNodeB", and forms a unique identity during registration. The UE collects data or interacts with data from the edge layer.
- *Edge layer.* The edge layer includes the infrastructure incorporating connectivity, computing and storage in distributed hyperconverged systems. The edge layer securely connects and manages MEC servers, including the blockchain network.
- *Cloud layer.* The public, private, or hybrid cloud is the cloud layer. The cloud provides computing and storage for locally executed applications and services.

2.2. Blockchain technology in MEC

Blockchain, a fundamental element of electronic ledgers, facilitates transaction tracking, distribution, and synchronisation among network nodes [7]. This technology, often referred to as Distributed Ledger

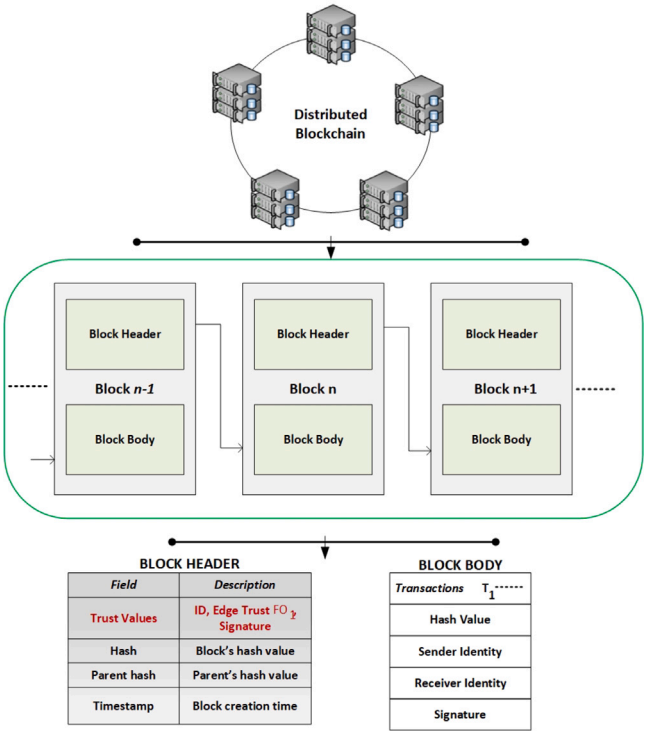


Fig. 1. Distributed blockchain.

Technology (DLT), operates on the principles of decentralisation, digital signatures, and encryption, specifically focusing on blockchain's immutable hash-based transaction recording [21].

In a decentralised system, individual nodes autonomously maintain and update a shared ledger designed to be immutable and append-only. This transparency addresses the limitations of a central ledger, with transactions validated through mechanisms like consensus algorithms [7]. Blockchain technology is integral to the described system, recording network activities using immutable cryptographic hashes, distributing transactions across interconnected ledgers, and ensuring the integrity of each block with real-time updates. Incorporating blockchain into the MEC framework minimises the risk of server compromise, fostering a reliable and trustworthy network ecosystem. The distributed blockchain in the edge layer is depicted in Fig. 1.

2.3. Zero trust orchestrator

The proposed ZTO is responsible for UE authentication, access control, policy enforcement, network segmentation, and monitoring tasks, as depicted in Fig. 3. It helps coordinate the authentication and authorisation processes, evaluate trust levels, and dynamically adjust access privileges based on contextual information [22].

The ZTO comprises a Policy Decision Point (PDP), Policy Enforcement Point (PEP), Resource Allocation Manager (RAM), and Inter/Intra MEC Federation Manager (MFM). The MFM is out of scope and left for future work. The PDP is logically split into the Policy Administration Point (PAP), Policy Information Point (PIP), and the policy engine, also known as the Access Control Policy (ACP). The policy engine is a decision-maker for access to MEC resources. The PAP executes the decision by signalling the PEP to shut down or establish the connections between an UE and a MEC resource.

The PEP is implemented as an edge authenticator between the UE requesting access and the resource accessed, e.g., an MEC application or data repository. It acts as an intermediary that enforces the policies and controls defined by the MEC operator. The PEP makes access

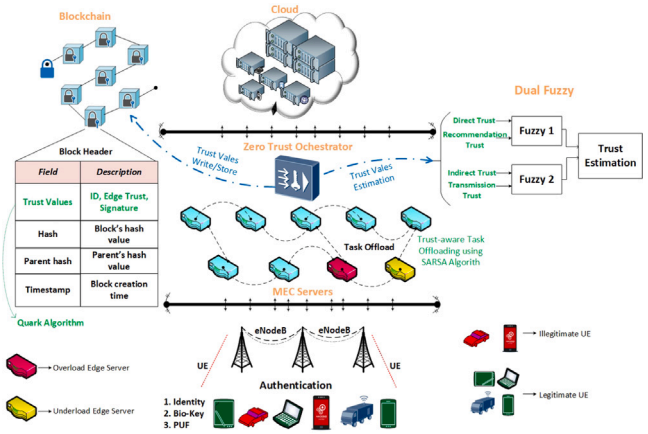


Fig. 2. The zero trust model.

control decisions and implements the defined policies in real time. It validates UE credentials, evaluates UE context and device information, and checks for compliance with security policies before granting access. The PEP may use various mechanisms, e.g., multifactor authentication, user and device profiling, and continuous monitoring, to determine the appropriate level of access.

The RAM manages and allocates computing resources in the MEC environment. Its primary role is to efficiently distribute resources to meet the requirements of applications and services at the network edge. It continuously monitors resource usage and availability, ensuring appropriate allocation based on QoS parameters. The RAM is crucial for optimising resource utilisation, ensuring efficient application execution, and maintaining desired QoS levels. It enables the effective utilisation of edge computing resources to support latency-sensitive applications and services.

2.4. System model

In our proposed framework, we focused on mobile networks as the scenario of interest. The mobile network represents a dynamic and intricate environment with stringent performance requirements. Our scenario comprises eNodeBs, RAN, and RSU e, each equipped with its individual MEC server m. Numerous service-specific applications s are available within the MEC servers, and each application includes a set of service instances i, as illustrated in Fig. 3. Within this context, mobile users denoted as UE, are present, and each user requests access to a specific service application running on the nearest MEC. The time is divided into discrete time slots denoted by t. At each time slot t, a mobile user u requests access to the service-specific application from a MEC server m (see Table 3).

2.5. Computation model

We consider that each mobile user has computational tasks that require strict adherence to latency requirements. As a result, mobile users require access to service-specific applications hosted on the MEC server to meet their demanding performance needs through computational offloading. The computational model can be described in three stages, drawing inspiration from previous studies [4,11,18].

Firstly, the mobile user initiates access by uploading a specific amount of data, which the MEC server must process via the RAN. The RAN forwards this request to the PEP module for UE registration and authentication. Secondly, the PEP module registers and authenticates the UE utilising the PRESENT protocol [20]. Finally, the processed results from the MEC server are transmitted back to the UE. Hence, the

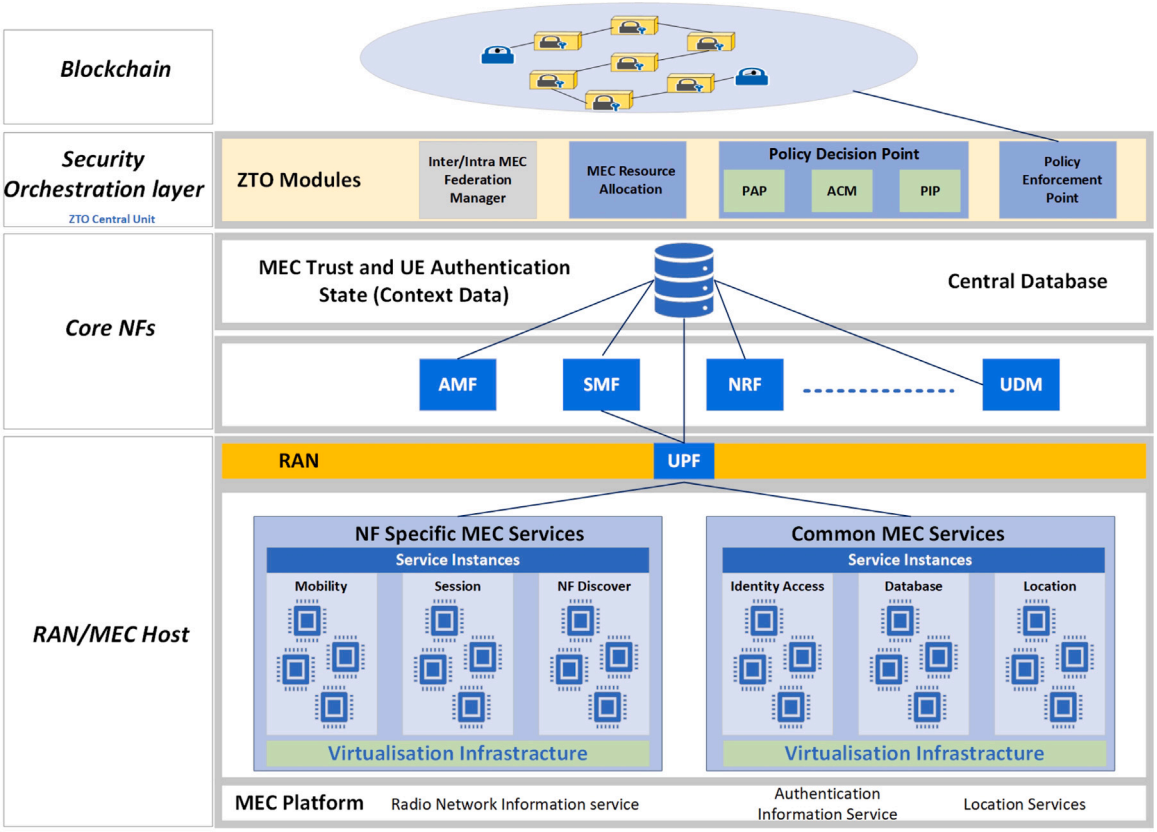


Fig. 3. Zero trust management reference architecture.

Table 3
Notation table.

Notation	Definition
e	eNodeBs/RAN/RSUs
u	User Equipment
s	Service-specific Application
i	service instance
m	MEC server
g^t_{um}	Channel gain between user u and MEC m
p_m	MEC server transmit power
σ^2	Noise power
I'_M	Accumulated interference from neighbouring MEC servers
b_m	Bandwidth allocated by the MEC server m
s_u	Service size requested by UE u
c^t_{us}	Requested s computation capacity by user u
RS^{max}_{sm}	Maximum computation capacity of s in MEC m
L^t_{sm}	Current load on the s in MEC server m

computational model encompasses the processing delay incurred in all steps.

When a mobile user u requests a service-specific application s from the MEC server m, the communication delay depends on factors such as the achieved data rate, wireless channel conditions, and the size of the requested service [11]. The channel gain between the mobile user u and MEC server m at time t is represented by g^t_{um}, and the signal-to-noise ratio can be defined as follows:

$$\gamma^t_{um} = \frac{p_m g^t_{um}}{I'_M + \sigma^2}, \tag{1}$$

where p_m represents the MEC server's transmit power, σ^2 represents the noise power, and I'_M represents the accumulated interference from neighbouring MEC servers [23,24]. The achieved data rate can be defined as follows:

$$R^t_{um} = b_m \lg(\gamma^t_{um} + 1), \tag{2}$$

where b_m is the bandwidth allocated by the MEC server m. The communication delay between the MEC server m and the mobile user u requesting a service-specific application at time t can be defined as:

$$d^t_{um} = \frac{s_u}{R^t_{um}}, \tag{3}$$

where s_u represents the size of the service requested by the mobile user u. The downloading delay of the processed service requests also relies on the size of the processed tasks and the download data rate of the mobile user u [10,25].

The service-specific application computation delay depends upon the resources allocated within an MEC server by the proposed RAM and the required computation capacity of the mobile user u at time t [9,10]. The computation delay between the service-specific application s in MEC server m and the mobile user u at time t can be defined as:

$$C^t_{u_{sm}} = \frac{c^t_{us} L^t_{sm}}{RS^{max}_{sm}}, \tag{4}$$

where c^t_{us} represents the requested computation capacity in [CPU cycles] for MEC application s by mobile user u at time t, RS^{max}_{sm} represents the maximum computation capacity allocated to an application s within a MEC server m, and L^t_{sm} represents the current computational load on the application s within a MEC server m, i.e., the number of mobile users accessing the same service-specific dedicated application s at time t.

2.6. MEC resources availability and allocation

MEC provides limited network resources that can be insufficient to deal with increased traffic due to UE mobility and the rising demand for providing low-latency applications. Therefore, optimising the MEC resource availability is essential. Our proposed framework leveraged ZTO functionalities to monitor resource utilisation and availability,

ensuring they are allocated appropriately according to the defined QoS parameters. We implemented practical algorithms to ensure MEC resource sharing, effective distribution of tasks, and MEC resource utilisation. The MEC resource availability and allocation function is represented as:

$$f_{RAM} = 1 - \frac{\sum_{s=1}^{n} RS_{sm}^{t}}{RS_{m}^{max}}, \tag{5}$$

where n represents the total number of MEC services, $\sum_{s=1}^{n} RS_{sm}^{t}$ represents the sum of the MEC resources, e.g., channel bandwidth, being used by connected UE U within MEC server m at time t, and RS_{m}^{max} represents the maximum MEC resources, i.e., the MEC server capacity m.

The optimisation goal ensures $\max f_{RAM}$ addresses two main constraints, C1 and C2. Constraint C1 is defined as resource allocation to service-specific application s in MEC server m is greater than or equal to the minimum resource required, i.e., minimum bandwidth, delay and jitter threshold requirements, to achieve an optimum QoS.

Constraint C2 is defined as the sum of the resources allocated to the applications running in MEC server m at time t cannot exceed the maximum system capacity of the MEC server.

$$\begin{aligned} &C1: RS_{s}^{m} \geq RS_{s}^{\theta}, \\ &C2: \sum_{s=1}^{n} RS_{sm}^{t} \leq RS_{m}^{max}. \end{aligned} \tag{6}$$

Our proposed framework aims to maximise the availability of resources on MEC servers to support the initiation and task offloading of new UE requests while fulfilling the dedicated service-specific QoS performance requirements. This is achieved by utilising the modularity and independently deployable functionalities the proposed ZTO framework offers.

3. Implementation of trust-aware authentication and task offloading in MEC

This section explains the working principles of our proposed trust management in MEC, as shown in Fig. 3.

3.1. UE registration and authentication

UE registration and authentication in MEC involves identifying and verifying the UEs allowed to access and utilise the computing resources at the network edge. We implemented the PRESENT algorithm that provides authentication and converts UE credentials into ciphertexts using the block cipher techniques [20].

The dynamic nature of UE implies that the set of UE connected to an MEC server is not fixed. Each UE has a different mobility profile [26]. Let U be the UE, and n be the total number of UE. $U_n = [U_1, U_2, U_3, \ldots, U_n]$ and each U has a unique identity ID, biometric characteristics BO and PUF entity embodied in a physical structure PF.

BO are physical traits of an individual that can be used to identify or verify a person's identity. They can be used to secure access to MEC resources and services and authenticate the identity of users and devices in the MEC network.

PF is a security feature used to uniquely identify a UE by taking advantage of the random variations in its physical characteristics. In practice, PUF works by using a set of challenge-response pairs, where a challenge input is provided to the PUF circuit, and the response is a unique output that is generated based on the physical characteristics of the UE. The answer is stored in a secure location and is used as a secret key for secure communication and authentication.

The UE register with the three security credentials. The BO is a binary value consisting of two random 4-bit binary numbers combined using the XOR Boolean operator. After registration of U_n, the values are authenticated every time a request is submitted. The UE authentication takes place in the ZTO based on the following steps, illustrated in Fig. 4

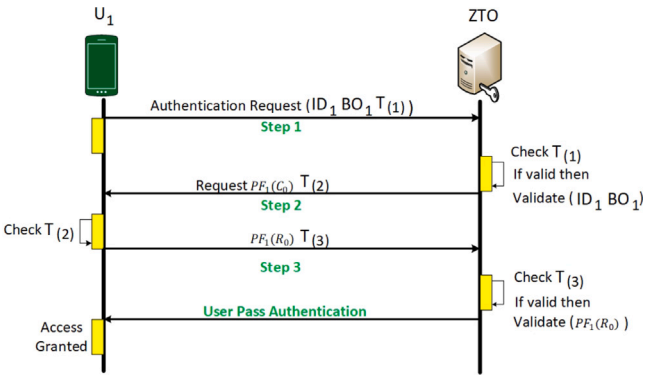


Fig. 4. UE registration and authentication.

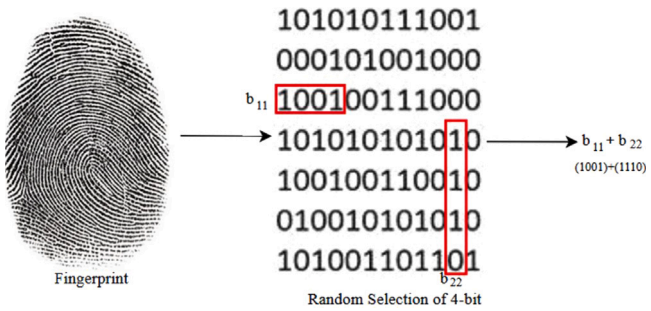


Fig. 5. Representation of fingerprint conversion process.

(1) Let D be a UE that submits an authentication request with $[ID_1, BO_1, T_{(1)}]$, where $T_{(1)}$ is the first timestamp created during the request transmission. First, the PDP module checks if the timestamp is valid or not. If $T_{(1)}$ is valid, then it validates $[ID_1, BO_1]$, where $BO_1 = (bo_{11} \oplus bo_{22})$ in which bo_{11} and bo_{22} are 4-bit binary numbers extracted from the fingerprint biometric. Representation of the fingerprint conversion process, exemplifying the transformation in the context of the proposed methodology, is depicted in Fig. 5.

(2) Then PDP module requests the UE with $PF_1(C_0)$ and timestamp $t_{(2)}$. The D_1 validates timestamp, and then it provides the corresponding $PF_1(R_0)$ for the $PF_1(C_0)$ to the AP. Here C_0 is the challenge in PUF, and R is the corresponding response to it.

(3) PDP module receives (C_0, R_0) from device D_1 and verifies $t_{(3)}$, if valid it then verifies (C_0, R_0). After verification completion, the next step occurs.

(4) The D_1 is successfully authenticated by the PEP module and marked as a legitimate UE, and the task request is forwarded to the edge layer.

According to the steps, the PEP module authenticates the UE. Only legitimate UE with successful authentication are granted access. UE registration and authentication are critical security measures in MEC, as they help to prevent unauthorised access and ensure that only trusted UE are allowed to access the MEC environment.

3.2. Edge trust validation in blockchain

Edge trust validation in the blockchain is a mechanism that ensures the trustworthiness of the edge servers within an MEC environment, which is critical for ensuring the security and reliability of the MEC servers.

In MEC, the servers are responsible for processing and storing data and providing computing resources to applications and services. It is

essential to ensure that these servers are trustworthy and have not been compromised by malicious actors.

Blockchain technology utilises a decentralised ledger to record transactions and changes made to the MEC environment. This ledger makes it possible to verify the integrity and authenticity of the data being processed and stored by the MEC servers.

The MEC servers can be registered as nodes in the blockchain network. Each node can be assigned a unique identifier, which can be used to track its activity and ensure that it behaves as expected. The blockchain network can also include mechanisms to monitor the health of the nodes, detect anomalies or attacks, and take appropriate actions to protect the integrity of the MEC environment.

The ZTO monitors the MEC servers and computes server trust using the dual fuzzy method [27]. Tasks are offloaded from a UE to the MEC server for processing based on a set of criteria evaluated using a fuzzy logic system. The fuzzy logic system considers the uncertainty and imprecision of the input criteria, such as task priority and computational complexity. It provides a degree of membership for each measure. This degree of membership is used to determine the suitability of a task or an MEC server for offloading.

Fuzzy Logic is a mathematical method that adeptly models and manages uncertainty in decision-making by employing membership functions to quantify the degree of an input's association with a specific set. Widely applicable, Fuzzy Logic finds use in control systems, image processing, natural language understanding, medical diagnostics, and artificial intelligence. Within Fuzzy Logic, the membership function plays a pivotal role as it establishes a mapping from an input value to a membership degree, typically falling within the range of 0 to 1. Here, a value of 0 signifies non-membership, while a value of 1 denotes full membership. The practical implementation of Fuzzy Logic hinges on using Fuzzy Rules, which articulate the nuanced relationship between input and output variables within a fuzzy context. This stands in contrast to Boolean systems, where the rigid delineation between true and false values is replaced with the ability to accommodate and process partial truths and falsities. The fuzzy architecture depicted in Fig. 6 comprises three essential modules:

- *Fuzzification Module.* This component converts crisp numerical system inputs into fuzzy sets.
- *Inference Engine.* Emulating the human reasoning process, the inference engine conducts fuzzy inference on the input data, applying IF-THEN rules.
- *Defuzzification Module.* Upon receiving the fuzzy set generated by the inference engine, the defuzzification module transforms it into a precise, crisp value.

In our work, we consider direct, recommendation, indirect, and transmission for computing trust. The dual fuzzy means that two fuzzy engines are used parallelly to estimate MEC servers' trust. The trust is computed as follows.

(1) Direct Trust DT. Refers to the level of trust assigned to an MEC server based on its past behaviour and performance, e.g., power, memory and storage capacity, and network connectivity. Direct trust is calculated based on the server's record of successfully executing tasks and the quality of the tasks it has executed.

(2) Recommendation Trust RT. Refers to the level of trust assigned to an MEC server based on the recommendations of neighbouring trusted servers in the MEC cluster and is used to determine the trustworthiness of a server when its direct trust cannot be accurately determined. For example, if Server A has a high level of trust in Server B, and Server B has a high level of trust in Server C. Server A may also have a high level of trust in Server C, even if it has limited direct knowledge about Server C. The RT_i of a server is used to make decisions about which servers are suitable for task offloading and which tasks can be safely executed.

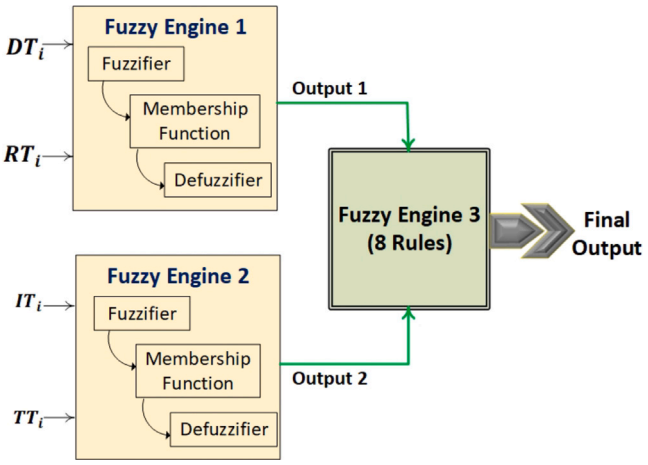


Fig. 6. Dual fuzzy trust model.

(3) Indirect Trust IT. A server IT is a measure of the overall trustworthiness of the MEC network and considers the trust relationships between UE and MEC servers in the cluster. In our work, the IT is the value given to the MEC server by UE.

$$IT = \sum_{1}^{n} UT_n, \tag{7}$$

where UT_n denotes User Trust of n UE communicating with the particular MEC server, the trust value from UE ranges between 0–1, where 0 is an untrustworthy server, and 1 is a highly trustworthy server.

(4) Transmission Trust TT. Refers to the level of trust assigned to the communication channel used to transmit data and tasks between servers in the MEC network and is used to ensure the confidentiality, integrity, and authenticity of the information being transmitted. The transmission trust denotes a trust value concerning successful transmission in a selected route. The TT is computed using,

$$TT = R + S, \tag{8}$$

where R represents the number of relay transmissions and refers to the use of intermediate communication devices to transmit data and tasks between servers in the MEC network. It is essential to evaluate the trustworthiness of relay devices and use secure encryption and authentication methods to protect the information being transmitted. The number of successful transmissions is represented by S.

The fuzzy engines take two trust values as input. The fuzzy interference entity is composed of four fuzzy rules. The proposed dual fuzzy-based trust model is illustrated in Fig. 6. The fuzzy engines contain three components: fuzzifier, membership function and defuzzifier [13,15]. The concept of crisp input is used to handle situations with no ambiguity or uncertainty in the data being used as input to the fuzzy logic system. While fuzzy logic allows for dealing with uncertain and imprecise information through fuzzy sets and membership degrees, crisp inputs are used when the input data is precise and deterministic. The fuzzifier is the initial block that receives crisp input and converts it into a fuzzy set for processing according to the membership function. The membership function is the set of rules defined using the input parameters. The results are analysed using the membership function, and fuzzy output is given to the defuzzifier to convert it into crisp output, which refers to the final output or decision produced by the fuzzy logic system, which is precise and unambiguous, as illustrated in Table 4.

Table 4
Fuzzy rules.

Fuzzy engine 1		
DT_i	RT_i	FO_1
H	H	H
H	L	M
L	H	M
L	L	L
Fuzzy engine 2		
IT_i	TT_i	FO_2
H	H	H
H	L	M
L	H	M
L	L	L
Fuzzy engine 3		
FO_1	FO_2	FO_3
H	H	H
H	M	H
H	L	M
M	H	H
M	M	L
M	L	L
L	H	M
L	M	L
L	L	L

The QUARK algorithm employs the sponge construction utilising a b-bit permutation P, as introduced in [28]. The structure of each QUARK instance is determined by the capacity c rate or block length r and digest length n. The designers predate the initial state for each instance of QUARK with b-bit values.

Three phases are applied to process a message m: initialisation, absorbing, and squeezing. In the initialisation phase, message m is padded to meet the rate r requirement. The padding involves appending a '1' bit, followed by the least number of '0' bits necessary to ensure the total length is divisible by the rate r.

In the absorbing phase, the r-bit message block is incorporated into the construction by XORing it with the last r bits of the internal state. The internal state is subsequently updated using the permutation P.

During the squeezing phase, the last r bits of the internal state are extracted, and the permutation P updates the internal state repeatedly until the total n bits are obtained as the final digest.

In fuzzy logic, rules are defined as statements using linguistic variables and fuzzy sets to represent imprecise and uncertain information. Each rule consists of an "IF-THEN" statement that relates the input variables to the output variable. The IF part specifies the conditions based on the values of the input variables, and the THEN part defines the output value or action based on the input conditions. The rules are defined as statements, e.g., 'IF INPUT 1 and INPUT 2 = 0, THEN OUTPUT = 0'. In this way, a set of rules is defined in fuzzy logic. The computed trust value is hashed and stored in the blockchain. The QUARK algorithm reduces the computations required for hashing, saving resource utilisation.

The QUARK algorithm utilises a permutation P, which is influenced by non-linear Boolean functions [29] represented as f, g, and h. A combination of Non-linear Feedback Shift Registers and one Linear Feedback Shift Register determines the internal states. The clock function plays a crucial role in processing and predicting net states. The output hash is obtained through the three functions f, g, and h. This hash value of the MEC servers is securely stored in the blockchain, ensuring the integrity and immutability of the data. Combining these cryptographic techniques and their integration with blockchain technology enhances the security and reliability of the QUARK algorithm in MEC environments. The QUARK pseudocode is presented in 1. We used the default $applyPermutation$, $padMessage$, $applyPermutation$, and $extractLast-rBits$ functions defined in the QUARK algorithm specification in [28].

Algorithm 1 QUARK Algorithm

```
function initializeInternalState
    internalState ← initialInternalState                ▷ b-bit value
end function
function absorbMessageBlocks(message)
    paddedMessage ← padMessage(message)  ▷ Pad the message to meet rate r requirement
    for i ← 0 to len(paddedMessage) by r do
        block ← paddedMessage[i : i + r]
        internalState ← internalState ⊕ block
        internalState ← applyPermutation(internalState, P)  ▷ Apply permutation P to update the internal state
    end for
end function
function squeezeDigest
    digest ← εε
    while length(digest) < n do
        last_r_bits ← extractLast_rBits(internalState)
        digest ← digest + last_r_bits
        internalState ← applyPermutation(internalState, P)  ▷ Continue applying permutation P to update the internal state
    end while
    finalDigest ← digest[ : n]        ▷ Ensure the digest is of length n
end function
function QUARKHash(message, c, r, n)
    initializeInternalState
    absorbMessageBlocks(message)

    return squeezeDigest
end function
```

Transactions are duplicated and distributed to linked ledgers in the blockchain as depicted in Fig. 1. Each block in the chain contains several transactions, and every time a new transaction arises, a record of that particular transaction is added to all of the blockchain ledgers. The blocks cannot be removed or tampered with after they have been added to a ledger. The hash value computed by the ZTO is submitted to the blockchain for storage. A dedicated block in blockchain technology is created and assigned to the MEC server. The transaction is validated to verify the trust value of the MEC server. After verification occurs, the new transaction information is added to a hashed block. Then, the block is verified, stored and propagated throughout the blockchain network.

As the number of MEC servers increases, more blocks are created and added to the blockchain. Incorporating blockchain technology within the MEC framework enables establishing a trustworthy network by mitigating the risk of compromised MEC servers. Implementing blockchain-based protocols within MEC is critical in ensuring the robustness and reliability of the MEC ecosystem.

3.3. Trust-aware task offloading

The ZTO uses the SARSA algorithm for trust-aware task offloading, combining reinforcement learning and trust evaluation, to make intelligent offloading decisions in MEC environments. The SARSA algorithm is a model-free, on-policy reinforcement learning algorithm that learns an optimal policy through environmental interactions [25].

The SARSA algorithm is used to learn the optimal offloading decisions based on the trustworthiness of the MEC servers. The algorithm learns from the experiences and feedback obtained during offloading to update its policy and make informed decisions.

To implement trust-aware task offloading using the SARSA algorithm, the steps are as follows:

Table 5
State-Action-Reward.

State i (Load, trust)	Action A	Reward R
$S_1 \rightarrow (l, T)$	A_1	R_1
$S_2 \rightarrow (l, T)$	A_2	R_2
$S_3 \rightarrow (l, T)$	A_3	R_3
⋮	⋮	⋮
⋮	⋮	⋮
$S_n \rightarrow (l, T)$	A_n	R_n

(1) **State representation.** Define the state representation, which includes the relevant information about the current state of the MEC environment, e.g., available resources, QoS requirements, and trust levels of the servers.
(2) **Action selection.** Based on the current state, select an action (task offloading decision) according to the current policy. The policy may be randomly initialised, and as the algorithm progresses, it will be updated to converge towards an optimal policy.
(3) **Task offloading and execution.** Offload the task to the selected MEC server. The task execution occurs at the MEC server, considering the available resources and QoS requirements.
(4) **Feedback and reward calculation.** Evaluate the outcome of the offloading decision and calculate the reward. The reward can be designed to reflect the quality of the offloading decision, such as minimising latency, energy consumption, or meeting QoS requirements. The reward is used to update the Q-values associated with the state–action pairs.
(5) **Q-value update.** Update the Q-value associated with the previous state–action pair using the SARSA update rule. The update rule adjusts the Q-value based on the current and expected future rewards. The Q-values represent the expected return for a particular action in a given state.
(6) **Repeat steps 2–5.** Repeat steps 2 to 5 for multiple iterations or until convergence. The SARSA algorithm gradually learns the optimal policy by updating the Q-values based on the experienced rewards.

By iteratively applying the SARSA algorithm, the ZTO can learn to make offloading decisions considering MEC server trustworthiness. The Q-values converge to reflect the expected return of the actions in a given state, enabling the system to make informed decisions that balance trust, resource constraints, and QoS requirements in MEC environments.

The pseudocode for the SARSA algorithm is illustrated in Algorithm 2.

Algorithm 2 Pseudo-code: SARSA Trust-Aware Task Offloading

1: Initialise the Q-table with arbitrary values for all state-action pairs
2: Set the initial state S_t
3: Choose an action A_t using an exploration policy that takes into account the trust values of available offloading servers from Table 5
4: Repeat the following steps until the terminal state is reached:
5: a. Take action A, observe the reward R and the next state S_{t+1}
6: b. Choose the next action A_{t+1} from S_{t+1} using the trust-aware exploration policy
7: c. Update the Q-value for the (S_t, A_t) pair using the formula represented in Equation (9)
8: d. Update the trust values of available offloading nodes based on the success or failure of offloading tasks
9: e. Update the state and action to be the next state and next action
10: Return the Q-table and updated trust values of available offloading servers
11: End

A trust-aware exploration policy is a decision-making strategy that always selects the action with the highest expected reward based on the current state. When choosing the following action, it considers the trust values of available offloading servers. The trust values are updated based on the success or failure of offloading tasks. The learning rate α and discount factor γ control the rate at which the Q-values are updated [30].

The three main elements in the SARSA algorithm are state S_t, action A_t and reward R. The exploration policy is mathematically represented in Eq. (9).

$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)] \quad (9)$$

Where S_{t+1} and A_{t+1} are the next state and the following action, respectively. Table 5 demonstrates the state–action-reward in this proposed SARSA, where each state depends on the load l and trust T.

The proposed algorithm follows a greedy policy by selecting the action with the maximum Q-value in the current state. The Q-value is the expected cumulative reward the algorithm will receive by taking a particular action in a given state followed by a specific policy. This algorithm selects an MEC server with high resource availability and offloads the task to balance the load at the edge layer.

4. Experimental evaluation

This section presents a practical implementation of our proposed architecture.

4.1. Simulation setup

The MEC environment was implemented using SUMO [31], Veins [32], JDK 1.8, and OMNeT++ [33]. RSU/eNodeB characteristics were configured, ensuring the framework's adaptability to diverse Radio Access Technologies (RATs), including Wi-Fi and 5G networks. Veins framework enhances realism in vehicular network simulations without compromising speed, while OMNeT++ and SUMO offer user-friendly GUI and IDE for simulation setup.

SUMO, an open-source traffic simulation suite, supports intermodal traffic modelling, automating tasks like network import, route calculations, and emission evaluation. The INET Framework for OMNeT++ provides communication network models. OMNeT++, chosen for its modular architecture, facilitates fast prototyping of MEC and supports well-known wireless propagation models.

For experimental evaluation, OMNeT++ handles high-level modules and network entity functionality. Key specifications are outlined in Table 6. Trust-based offloading and authentication processes were programmed within this setup, integrating OMNeT++, Veins, and SUMO to assess the proposed architecture's performance.

For the implementation, an ESXi server v6.7 hosted the Windows 10 virtual machine equipped with a quad-processor CPU and 16 GB RAM. The MEC service-specific application s modules were dynamically deployed as applications over the transport layer of the Virtualization Infrastructure. In the simulation, 100 mobile users were modelled as wireless UE, making requests from the corresponding eNodeB's MEC server at each time interval t.

The ZTO is deployed following a centralised approach where an individual and centralised controller manages its network, i.e., the RAN. The number of ZTO and their corresponding edge networks used in this evaluation scenario can be seen in Table 6. As Omnet++ [33] supports the emulation of API integration, we used the extended API package to integrate the RANs with the ZTO. The extended API module supports RANs to capture the network information, i.e., Received Signal Strength Indicator (RSSI) and network load. It forwards the network performance indicators by generating a *packet_in* message to its respective ZTO managing the network. The ZTO then extracts the information and the network performance indicators using the *get_protocol* method defined in [34] to have a global view of the underlying network.

Table 6
Simulation parameters.

Parameter	Setting
Network model	
Simulation Area	2.5 km ×2.5 km
Number of UEs	100
Number of eNodeB	2
Number of MEC servers	3
Number of ZTO	1
Speed of UEs	50 mps
Mobility model	TraCIMobility
Simulation run time	15 min
Authentication Algorithm	PRESENT
Blockchain hashing	QUARK
Task offloading Algorithm	SARSA
Queue Size	2 MB
Packet model	
Packet interval	5 s
Number of packets	49 – 99
Flow timeout	2 s
Service time	9.8 µs
Delay	1 µs
Communication model	
Data rate	300 Mbps
Link bandwidth	5 Mbps
Transmission range	80 m
Q-learning	
Learning rate (γ)	0.5
Discount factor (α)	0.5
Data size per task	500 MB
Maximum occurrence	300
The greedy factor range	0.4 - 0.9

The MEC server applications are modelled as a set of resources implemented as independent operating service instances chained by the RAM to complete a task offloading process. Resource allocation, e.g., bandwidth, is an optimisation problem that has been broadly discussed in the literature and is out of the scope of this research [27,30,35,36]. We assumed equal bandwidth allocation to the applications and services within an MEC server. The simulation parameters used in the performance evaluation are summarised in Table 6.

The model implemented in the OMNeT++ simulator is depicted in Fig. 2. Performance is determined based on the network devices, and the efficiency of the proposed work is identified.

4.2. Comparative analysis

We compared our proposal to a blockchain-based solution for MEC trust management presented in [11] and RTEM system [16]. A blockchain-based authentication mechanism utilising asymmetric cryptography is proposed. Data encryption is accomplished using elliptic curve cryptography, while the records within the blockchain are safeguarded through SHA-256 hashing. A UE receives a digital signature for identity verification during registration.

4.3. Performance metrics

The performance metrics are employed to compare the proposed solution with existing approaches. The simulation results concerning success, failure, task offloading delay and authentication time were analysed.

4.3.1. Success rate

The success rate refers to successfully transmitting packets from the UE to the edge layer through the RAN. The success rate should improve even as the load increases, represented by the number of UE requests. The load at the edge layer rises as additional requests

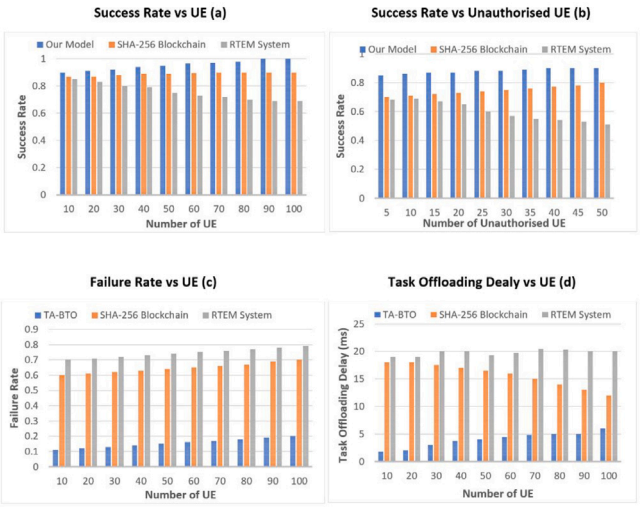


Fig. 7. Performance comparative analysis.

arrive for processing. Consequently, the traffic originating from UEs may contain malicious packets aimed at compromising the MEC servers and causing a decrease in network performance. It could be defined as the ratio of the number of tasks processed by the MEC server to the total number of requests received:

$$SR_m = \frac{N_m}{N_m^{total}}, \tag{10}$$

where SR_m represents the success acceptance rate of the MEC server m, N_m represents the number of tasks accepted by the MEC server m, and N_m^{total} represents the total number of tasks received by the MEC server m. The performance of success rate depends on the following factors:

- The processing of arriving packets without dropped or lost packets. The dropped or lost packets reduce the success rate.
- Efficient user request management enables higher loads, which increases the success rate.
- Unauthorised UE access attempts to MEC servers will utilise resources and reduce the success rate.

The MEC architecture aims to improve the success rate based on managing UE requests. The UE requests and the performance of the success rate are captured and depicted in Fig. 7(a).

As shown in Fig. 7(a), the proposed model achieves a 95% success rate compared with the 88% success rate in the earlier work. The average difference is a 7% improvement for our model when compared with the SHA-256 blockchain. The increase in requests also increases the success rate in our proposed model. The task offloading capability enables the edge layer to manage more requests than the SHA-256 blockchain.

The success rate progress is demonstrated by analysing the impact of an escalating number of unauthorised UE, as depicted in Fig. 7(b). Our model surpasses the SHA-256 blockchain and RTEM system in terms of success rate. This improvement can be attributed to our model incorporating authentication and trust evaluation. Unlike the existing approach that solely validates trust, our model effectively detects and disregards malicious UE. Consequently, the success rate diminishes in the SHA-256 blockchain as it relies exclusively on trust value validation.

The success rate achievement of our model and SHA-256 blockchain is 88% and 74%, respectively. The difference in success rate is 14% and is attributed to the absence of UE authentication in the SHA-256 blockchain.

4.3.2. Failure rate

The failure rate presents the rate of failed UE requests and can be defined as:

$$FR_m = \frac{F_m}{N_m^{total}}, \tag{11}$$

where FR_m represents the failure rate of the MEC server m, F_m represents the number of unsuccessful tasks by the MEC server m. The main factors for the failure rate are:

- The traffic load imbalance on MEC servers will overload particular edge nodes and cause packets to drop from the UEs.
- The access permission for malicious nodes without estimating trust values tends to increase the failure rate due to the arrival of unknown packets.

The goal is to minimise or eliminate the failure rate, ensuring efficient processing of tasks. In the given scenario, the failure rate rises as the number of unauthorised, i.e., malicious UE participating in the network increases. Fig. 7(c) illustrates the performance of the proposed model compared to the existing SHA-256 blockchain in terms of failure rate. Our model first authenticates UE and subsequently validates the trust values of MEC servers. As a result, our approach demonstrates improved performance with a lower failure rate compared to the existing SHA-256 blockchain model.

In our model, the average failure rate is recorded at 15%, which is significantly lower than the existing approach, with a failure rate of 64%. Our security provisioning system incorporates trust value computation and management, which considers unauthenticated UE attempting to use fake identities. Identifying and filtering out these unauthenticated users before they enter the network is essential to prevent an increase in the failure rate caused by submitting unknown packet structures.

4.3.3. Task offloading delay

Delay is a crucial parameter representing the time required to process incoming tasks. In a network model, minimising delay is essential to enhance overall network performance and ensure prompt task delivery. The comparison of delay parameters is depicted in Fig. 7(d). In our model, we observe changes in average delay and hit ratio as we vary the maximum tolerance of UE from 10 to 100. As the number of UE increases, the delay gradually rises. This can be attributed to the limited computing capacity of MEC servers. To address this, our model employs a strategy based on the SARSA algorithm to offload tasks and meet the demands of a reliable MEC architecture. The average delay is 4 ms in our model and 16 ms in the SHA-256 blockchain model.

4.3.4. UE authentication time

The depicted authentication time in Fig. 8 represents the duration the ZTO takes to authenticate the UE. Our observations indicate an average authentication time of 45 ms. Implementing a trust-aware framework showcases that our approach is a viable solution to enhance security and mitigate trust-related risks. However, there is an opportunity for further research on optimising algorithms tailored explicitly for MEC environments to improve reliability and security.

Security is a cornerstone of our proposed model, seamlessly integrating authentication, trust management, and trust-based offloading. The critical role of security is evident in Table 6, where we meticulously compare the security parameters of hashing algorithms. Notably, our findings demonstrate the superiority of the lightweight QUARK over the SHA-256 algorithm. This strategic use of a lightweight hashing algorithm ensures that the system can efficiently handle an influx of incoming user requests without compromising security (see Table 7).

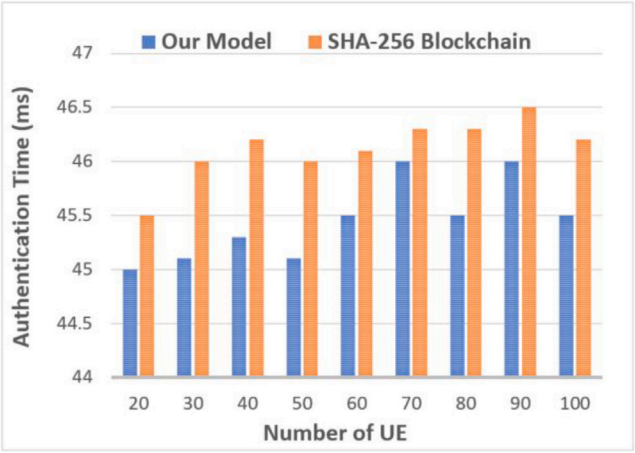


Fig. 8. UE authentication time.

Table 7
Hashing algorithms comparison.

Specification	SHA-256	QUARK
Output Size (bits)	256	128
Block Size (bits)	512	160
Number of Rounds	64	544
Collision (bits)	128	64

5. Conclusion and future work

This paper presents a layered architecture for MEC networks that utilises zero trust guiding principles to assess trust for MEC servers and enforce robust identity access for UE. The Policy Enforcement Point plays a crucial role in implementing zero trust principles by ensuring access to resources is granted based on the defined policies and continuously monitors to maintain secure and trusted MEC servers. Scalability is a crucial aspect in real-world deployments, and investigating how scalability will impact the proposed approach is left for future work. Utilising SDN controllers is an efficient and effective mechanism to implement a scalable solution that provides secure and reliable task-offloading in large network scenarios. Given the limited resources of MEC servers, efficient authentication algorithms must be developed to ensure smooth service continuity while performing UE identity verification. Drawing upon these insights, we believe our framework can be used as the basis for future work that explores scalability without compromising effectiveness.

CRediT authorship contribution statement

Belal Ali: Writing – original draft, Conceptualization. **Mark A. Gregory:** Writing – review & editing, Supervision, Resources, Methodology, Formal analysis. **Shuo Li:** Writing – review & editing, Supervision, Formal analysis. **Omar Amjad Dib:** Visualization, Project administration.

Declaration of competing interest

The authors 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

[1] ETSI, Multi-access Edge Computing (MEC); Framework and Reference Architecture, Tech. Rep. GS MEC 003, ETSI, 2020.

[2] B. Ali, M.A. Gregory, S. Li, Multi-access edge computing architecture, data security and privacy: A review, IEEE Access 9 (2021) 18706–18721.

[3] G. Mehta, G. Shrivastava, Comprehensive analysis of edge computing towards adaptive streaming for enhanced QoS, in: 2022 IEEE International Conference on Current Development in Engineering and Technology, CCET, IEEE, 2022, pp. 1–6.

[4] B. Ali, S. Hijjawi, L.H. Campbell, M.A. Gregory, S. Li, A maturity framework for zero-trust security in multiaccess edge computing, Secur. Commun. Netw. 2022 (2022).

[5] P. Phiayura, S. Teerakanok, A comprehensive framework for migrating to zero trust architecture, IEEE Access 11 (2023) 19487–19511.

[6] A. Manan, Z. Min, C. Mahmoudi, V. Formicola, Extending 5G services with zero trust security pillars: a modular approach, in: 2022 IEEE/ACS 19th International Conference on Computer Systems and Applications, AICCSA, IEEE, 2022, pp. 1–6.

[7] P. Bhattacharya, S. Tanwar, R. Shah, A. Ladha, Mobile edge computing-enabled blockchain framework—a survey, in: Proceedings of ICRIC 2019, Springer, 2020, pp. 797–809.

[8] U. Jayasinghe, G.M. Lee, Á. MacDermott, W.S. Rhee, TrustChain: A privacy preserving blockchain with edge computing, Wirel. Commun. Mob. Comput. 2019 (2019).

[9] L. Xiao, Y. Ding, D. Jiang, J. Huang, D. Wang, J. Li, H.V. Poor, A reinforcement learning and blockchain-based trust mechanism for edge networks, IEEE Trans. Commun. 68 (9) (2020) 5460–5470.

[10] D.C. Nguyen, V.-D. Nguyen, M. Ding, S. Chatzinotas, P.N. Pathirana, A. Seneviratne, O. Dobre, A.Y. Zomaya, Intelligent blockchain-based edge computing via deep reinforcement learning: Solutions and challenges, IEEE Netw. 36 (6) (2022) 12–19.

[11] S. Guo, X. Hu, S. Guo, X. Qiu, F. Qi, Blockchain meets edge computing: A distributed and trusted authentication system, IEEE Trans. Ind. Inform. 16 (3) (2019) 1972–1983.

[12] L.M. Dias, J.F. Ramalho, T. Silvério, L. Fu, R.A. Ferreira, P.S. André, Smart optical sensors for internet of things: Integration of temperature monitoring and customized security physical unclonable functions, IEEE Access 10 (2022) 24433–24443.

[13] G. Xia, C. Yu, J. Chen, A fuzzy-based co-incentive trust evaluation scheme for edge computing in CEEC environment, Appl. Sci. 12 (23) (2022) 12453.

[14] X. Deng, J. Liu, L. Wang, Z. Zhao, A trust evaluation system based on reputation data in mobile edge computing network, Peer-to-Peer Netw. Appl. 13 (5) (2020) 1744–1755.

[15] M. Poongodi, S. Bourouis, A.N. Ahmed, M. Vijayaragavan, K. Venkatesan, W. Alhakami, M. Hamdi, A novel secured multi-access edge computing based vanet with neuro fuzzy systems based blockchain framework, Comput. Commun. 192 (2022) 48–56.

[16] X. Deng, J. Liu, L. Wang, Z. Zhao, A trust evaluation system based on reputation data in mobile edge computing network, Peer-to-Peer Netw. Appl. 13 (2020) 1744–1755.

[17] A. Souri, Y. Zhao, M. Gao, A. Mohammadian, J. Shen, E. Al-Masri, A trust-aware and authentication-based collaborative method for resource management of cloud-edge computing in social internet of things, IEEE Trans. Comput. Soc. Syst. (2023).

[18] K. Mahmood, M.F. Ayub, S.Z. Hassan, Z. Ghaffar, Z. Lv, S.A. Chaudhry, A seamless anonymous authentication protocol for mobile edge computing infrastructure, Comput. Commun. 186 (2022) 12–21.

[19] G. Mitsis, E.E. Tsiropoulou, S. Papavassiliou, Price and risk awareness for data offloading decision-making in edge computing systems, IEEE Syst. J. 16 (4) (2022) 6546–6557.

[20] B. Ali, M.A. Gregory, S. Li, Uplifting healthcare cyber resilience with a multi-access edge computing zero-trust security model, in: 2021 31st International Telecommunication Networks and Applications Conference, ITNAC, IEEE, 2021, pp. 192–197.

[21] T. Kwantwi, G. Sun, N.A.E. Kuadey, G. Maale, G. Liu, Blockchain-based computing resource trading in autonomous multi-access edge network slicing: A dueling double deep Q-learning approach, IEEE Trans. Netw. Serv. Manag. (2023).

[22] B. Ali, M.A. Gregory, S. Li, O.A. Dib, Zero trust security framework for 5G MEC applications: Evaluating UE dynamic network behaviour, in: 2023 33rd International Telecommunication Networks and Applications Conference, 2023, pp. 140–144.

[23] D. Wang, H. Qin, B. Song, X. Du, M. Guizani, Resource allocation in information-centric wireless networking with D2D-enabled MEC: A deep reinforcement learning approach, IEEE Access 7 (2019) 114935–114944.

[24] Z. Li, X. Wang, X. Yue, D. Zou, J. Li, L. Si, Based on temporal and spatial traffic characteristics for energy-efficient Wi-Fi network, Math. Probl. Eng. 2021 (2021) 1–9.

[25] D.H. Abdulazeez, S.K. Askar, Offloading mechanisms based on reinforcement learning and deep learning algorithms in the fog computing environment: A comprehensive review, IEEE Access (2023).

[26] A.S. Patil, R. Hamza, A. Hassan, N. Jiang, H. Yan, J. Li, Efficient privacy-preserving authentication protocol using PUFs with blockchain smart contracts, Comput. Secur. 97 (2020) 101958.

[27] Y. Shi, J. Chu, C. Ji, J. Li, S. Ning, A fuzzy-based mobile edge architecture for latency-sensitive and heavy-task applications, Symmetry 14 (8) (2022) 1667.

[28] J.-P. Aumasson, L. Henzen, W. Meier, M. Naya-Plasencia, Quark: A lightweight hash, J. Cryptol. 26 (2013) 313–339.

[29] X. Lu, B. Li, M. Liu, D. Lin, Improved conditional differential attacks on lightweight hash family QUARK, Cybersecurity 5 (1) (2022) 1–16.

[30] M.D. Hossain, T. Sultana, M.A. Hossain, M.I. Hossain, L.N. Huynh, J. Park, E.-N. Huh, Fuzzy decision-based efficient task offloading management scheme in multi-tier MEC-enabled networks, Sensors 21 (4) (2021) 1484.

[31] M. Behrisch, L. Bieker, J. Erdmann, D. Krajzewicz, SUMO - simulation of urban mobility: An overview, in: SIMUL 2011, the Third International Conference on Advances in System Simulation, ThinkMind, Barcelona, Spain, 2011, pp. 63–68, URL http://elib.dlr.de/71460/,

[32] C. Sommer, D. Eckhoff, A. Brummer, D.S. Buse, F. Hagenauer, S. Joerer, M. Segata, Veins – the open source vehicular network simulation framework, in: Recent Advances in Network Simulation, Springer, 2019.

[33] A. Varga, OMNeT++, in: Modeling and Tools for Network Simulation, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 35–59.

[34] A. Varga, A. Sekercioglu, Parallel simulation made easy with omnet++, in: 15th European Simulation Symposium, Vol. 527, 2003, p. 528.

[35] W. Tan, K. Ding, X. Zhang, Z. Liang, J. Liu, Minimizing terminal energy consumption of task offloading via resource allocation in mobile edge computing, in: 2022 IEEE 25th International Conference on Computer Supported Cooperative Work in Design, CSCWD, IEEE, 2022, pp. 683–688.

[36] W. Kong, X. Li, L. Hou, J. Yuan, Y. Gao, S. Yu, A reliable and efficient task offloading strategy based on multifeedback trust mechanism for IoT edge computing, IEEE Internet Things J. 9 (15) (2022) 13927–13941.

Belal Ali is a Ph.D. candidate in the School of Engineering RMIT University, Melbourne, Australia. He has more than 17 years of working experience in enterprise network design, architecture and cybersecurity solutions. He received his Bachelor of Electronics and Telecommunications Engineering from Applied Science University in 2004. Research interests include SDx, MEC and associated cybersecurity solutions.

Mark A. Gregory (SM'99) is a Fellow of the Institute of Engineers Australia and a Senior Member of the IEEE. Mark A. Gregory received a Ph.D. from RMIT University, Melbourne, Australia, in 2008, where he is an Associate Professor in the School of Engineering. 2009 he received an Australian Learning and Teaching Council Citation for outstanding teaching and learning contributions. He is the Managing Editor of an international journal (IJICTA) and the General Co-Chair of an IEEE technically co-sponsored conference (ITNAC). Research interests include telecommunications, network design, security, public policy and technical risk.

Shuo Li is a Lecturer in the School of Engineering, RMIT University, Australia. She received a Bachelor's degree from the City University of Hong Kong, Hong Kong SAR, in 2009 and a Ph.D. from the same University in 2014. Her research interests include analysing and designing telecommunications, optical, and core networks.

Omar Amjad Dib is a third-year Computer Engineering student at Kuwait College of Science and Technology. Omar is carrying out innovative research projects in cybersecurity and blockchain.