Journal Pre-proof

DecChain: A decentralized security approach in Edge Computing based on Blockchain

Ernest Bonnah, Ju Shiguang

PII: S0167-739X(19)32393-3
DOI: https://doi.org/10.1016/j.future.2020.07.009
Reference: FUTURE 5731

To appear in: *Future Generation Computer Systems*

Received date : 9 September 2019
Revised date : 17 May 2020
Accepted date : 4 July 2020

Please cite this article as: E. Bonnah and J. Shiguang, DecChain: A decentralized security approach in Edge Computing based on Blockchain, *Future Generation Computer Systems* (2020), doi: https://doi.org/10.1016/j.future.2020.07.009.

DecChain: A decentralized security approach in Edge Computing based on Blockchain

Ernest Bonnah *Ju Shiguang**

School of Computer Science and Telecommunication Engineering, Jiangsu University, Zhenjiang, Jiangsu Province, P.R. China.

**Corresponding Author: jushig@ujs.edu.cn*

This work was supported by the National Key R&D Program, China (No. 2016YFD0702001) and Modern Agriculture Projects of Jiangsu Province (No. BE201735B).

Abstract

Security in edge computing paradigms has become a major concern in recent times due to the integral role it plays in the framework of edge computing. Privacy-preserving and data security challenges are among the many concerns impeding the goal of making data storage available and processing at the edge of the network quite difficult to implement. Authenticated users must be the only ones with access to their respective stored data which are protected against any form of intruder manipulation. Most authentication schemes proposed and implemented in edge computing and other paradigms use a trusted entity to initialize the authentication process between edge servers and prospective users. Servers and users are expected to register with the trusted party first before they are able to subsequently authenticate one another. The presence of the trusted party presents scalability issues as well as the threat of having a single point of failure which may threaten the availability of the entire network.

In this paper, we present a fully decentralized approach to solving this problem by eliminating the public trusted entity within the network framework termed DecChain. In DecChain we employ some notable principles of permissioned blockchain technology in the rollout and authentication of elements within the network. Authenticated users within our proposed framework would not have to sign in to every service provider to be authenticated to access a service or resource. Security experiments and the deployment of our scheme are carried out to evaluate the performance of DecChain. The results show our scheme is secured and achieves the intended purpose efficiently.

Index Terms: Blockchain, Edge Computing, Security, Scalability, Authentication, Decentralized

1.0 Introduction

The emergence of Cloud Computing technology and its accompanying services were widely appreciated since it was seen as a panacea to solving all the challenges posed by the overwhelming demand of pervasive intelligent devices and new network applications. Despite the huge benefits present in cloud computing such as convenience, rapid elasticity, pay-per-use, ubiquity, etc. [1], cloud computing also presented its version of challenges as network applications evolved with growth in end-user devices. Generally, public cloud vendors made available large data centers in various parts of the world that had enough computing resources to meet the demands of the users. However, the centralization of these resources created a seeming vast separation between end-users and their respective cloud. This, in turn, increases the average latency and jitter [2].

Again, the rapid development of pervasive intelligent devices and ubiquitous networks gave way to a new variety of network applications and services. Cloud computing has been hugely challenged by the

requirements of services such as the Internet of Things, Internet of Vehicles, Internet of Everything, smart city, smart grids, social networks, etc. [3]. In the face of the centralized computing paradigm in the design of cloud computing, meeting the associated services of these proliferating new network applications seems a daunting task.

Latency and jitter have become a major issue because of the dynamic nature of end-user requirements which is a result of the development of ubiquitous network technologies[3]. Users have evolved over the years from data consumers to data producers with an increasing demand for big data processing capabilities such as data acquisition, pattern recognition and data mining [4]. These evolving demands of users coupled with other limitations of end-user devices and real-time applications, low latency is a crucial demand in any modern application. Delay sensitive - applications like GPS navigation, object recognition, language translation, real-time machine learning, healthcare monitoring networks, vehicular networks, etc. make it highly impossible to tolerate any form of delay in the network architectural framework. In summary, the limitations of traditional cloud computing due to the centralized model of operation can be summarized as follows [4]:

1. Linear growth in computing capabilities of cloud computing cannot meet the multi-sources data processing requirement at the edge of the network.
2. The network bandwidth and the transmission speed have come to a bottleneck because of the large scale of user access, while long-distance transmission between user and cloud center will lead to the high service latency and waste of computing resources.
3. Private user data in edge devices are likely to be leaked during the outsourcing process.

In addition to the above limitations enumerated, the distance between mobile nodes and the cloud center, makes cloud services unable to access contextual information such as user location, local network conditions or even sensitive information about users' mobility behavior [1].

Edge computing has the potential in providing the solution to the many challenges of traditional cloud computing. Combined with cloud computing, edge computing can efficiently handle the edge big data processing problems as well as storage resources. Edge computing leverages the concepts and principles of cloud computing and takes the concept of content delivery networks (CDNs) a step further. While CDNs are all about caching web content, edge computing seeks to include the accompanying computation as was being done in cloud computing [5].

There are various forms of edge paradigms such as fog computing [6-8], mobile edge computing (MEC) [9, 10], mobile cloud computing (MCC) [11], dew computing [12] among others. Table 1 summarizes the distinctions between these edge computing paradigms. Despite the differences among the edge computing paradigms, the underlining principle of all these types of edge computing is the same – to bring traditional cloud computing services to the edge of the network. Thus, storage and computation resources in cloud infrastructure are logically or physically deployed close to the end-user devices, where the data originates, intending to improve response time and accelerate processing. The deployment of a replica of the core of the network at the edge of the said network will go a long way to offload the computation and communication burden of the network core. Additionally, there is an improvement in the quality of service relative to delay-sensitive services as well as an enhanced opportunity to improve user privacy and data security.

Despite the improvements offered by the various edge paradigms in providing enormous solutions to the limitations and bottlenecks of traditional cloud computing, there are fundamental security challenges with this emerging approach. This paper seeks to discuss a common security challenge that confronts the paradigms mentioned in this paper – Privacy-Preserving and data security. This will be done with focus on

an enhanced decentralized form of authentication that resonates with the principle behind the deployment of these schemes.

Table 1. Differences in Edge Computing Paradigms

	DEW	FOG	MEC	MCC
Ownership	Private entities, individuals	Private entities, individuals	Telecommunication companies	Private entities, individuals
Purpose	Make services available to users using on-premise computers without collaborating with cloud computing	To meet the demands of delay sensitive services, mobile support, geographical distribution of the IoT	To move cloud computer-like service such as storage and computation from the core of the network to the edge to reduce latency	To delegate storage and computation services to devices at the edge of the network to improve service delivery
Deployed Hardware	On-premise computers	Routers, switches, access points, gateways, heterogeneous servers, etc.	Radio access points, base stations, heterogeneous servers, etc.	Radio access points, base stations, servers, user devices
Deployment	User devices	Near-edge, edge	Network edge	Network edge, devices
Target Users	Common internet user (including mobile users)	Mainly mobile users	Mainly mobile users	Mainly mobile users

The major contributions of the paper are:

1. We discuss the limitations of conventional method of authentication in both cloud and edge computing environment.
2. We evaluate the prospects of blockchain technology as well as the challenges expected with the integration of blockchain into edge computing environment.
3. Based on our findings, we propose a comprehensive decentralized scheme based on the principles of permissioned blockchain technology.

The rest of this paper is organized as follows: Section 2 discusses the authentication in edge paradigms, in Section 3, we discuss the principles and strengths of Blockchain. Section 4 introduces DecChain, our scheme designed based on blockchain. Informal cryptanalysis of the proposed scheme is carried out in Section 5. Cryptanalysis of our scheme using AVISPA is performed and discussed in Section 6 with a further analysis based on a simulated deployment on NS2 done in Section 7. Performance and comparative analysis between our scheme and others were done in Section 8. Section 9 concludes the paper and outlines future works to be done on our scheme.

2.0 Challenges in Edge Paradigms

Mechanisms to preserve privacy and improve upon data security within the framework of cloud computing will not be applicable in an edge computing environment. This is because, edge computing support services such as location-awareness, mobility support, heterogeneity distributed architecture, etc. It is therefore important to discuss security in the exclusive view of edge computing infrastructure as we seek to propose ways of improving this service.

The decentralization of the network, as well as the movement of the services to the edge of the network, comes with huge benefits as has already been espoused. Nonetheless this approach to improve service delivery also has its security challenges that ought to be addressed. Every potential security threat in every tier or layer of the network, therefore, has to be addressed in the design of any edge infrastructure to protect the data of the end-user. In [4], Zhang et al. outlined all the potential threats in edge networks from the core infrastructure, edge servers, the edge networks to the mobile edge devices. Roman et al. [1] did a categorization of all the threats to the network of an edge paradigm while analyzing the potential threats to all the services that these emerging networks offer.

There are multiple elements in the rollout of any of the forms of edge computing at each of the layers. These elements may be either in the same domain or not. However, there has to be an attempt for collaboration or coexistence promoting heterogeneity in the network which is characteristic of edge computing. At the heart of all these potential threats and attacks is the need to guarantee the identity of every entity as well as to ensure the right entity has access to the right information. Without an efficient authorization scheme, internal and external attackers can take advantage of sensitive resources and information of users and manipulate them to their benefit [3].

Furthermore, one other issue that presents a huge security challenge to edge computing is the outsourcing feature as part of the design. This makes access control an inevitable necessity within any edge computing virtualization framework [1, 4]. The possibility of multiple trust domains in the technology presents an opportunity for intruders to access, misuse and abuse resources in the absence of a unique identifier to every entity in the network as well as an efficient authentication mechanism between entities [1].

Privacy continues to be one of the most critical security requirements of edge computing. This is because within the framework of the network, there is the possibility of having authorized entities who seek to gain access to sensitive information of other, authorized entities for their use. These malicious attackers can be either a service provider or an edge user. In an implementation environment with multiple trust domains, it is highly possible such attacks can be rampant without the right security schemes in place. An efficient security system should be able to maintain the anonymity of its users' while ensuring the authorized users use the assigned resources as expected [4, 13].

2.1 Authentication Concepts of Edge Computing

A scheme that seeks to address the security challenges of edge computing as enumerated above has to be premised on an enhanced authentication scheme which provides the platform for solving all the challenges above. Most proposed authentication schemes designed for edge computing environments have been applicable only in single-domain [4]. These schemes are not applicable for cross-domain schemes which seem to be the future of this emerging technology. Edge computing has evolved to embrace distributed interactive computing systems with multiple trust domains where there is a coexistence of multiple functional actors, services, and infrastructure [4]. Hence authentication and authorization schemes should be comprehensive to first provide an identity for every element of interest and then authenticate each other across different trust domains bearing in mind the mobility of these edge mobile devices.

Again, authentication schemes should have mechanisms to monitor the judicious use of allocated network resources [1]. This helps to identify authorized malicious attackers within the network framework looking for the least opportunity to manipulate user profile information. An authentication scheme should not only look at trying to keep external adversaries out but also internal adversaries who present an even greater threat since they are already authenticated and granted authorization to some resources in the network already. All these attempts to ensure authorized users are restricted in perpetuating any attack on the network should be done without compromising user anonymity requirements.

2.2 Overview of Current Authentication Model in Edge Computing

Authentication mechanisms have evolved with steady progress in cloud computing. Conventional methods of authentication focused on providing a strong means of authenticating users to enable them to remotely access their data in on-demand mode. The authentication requirements within any network setup have evolved over the years as users may request access and share each other's authorized data fields to achieve productive benefits [14]. A basic approach to authenticate each other using a combination of a chosen identity (username) and a password [15]. In recent times, most authentication schemes have in addition to the two parameters introduced biometric pattern input as a means of making their scheme more secured and inaccessible to malicious attackers [13, 16-18]. Single sign-on authentication has recently been a more advanced way of authentication elements in both cloud and edge computing [19]. In network systems using this means of authentication which mostly requires a trusted 3rd party in the authentication process, users can access multiple edge services using one secret key or password. OpenID is a classic example of single sign-on authentication which is made up of three elements playing different roles namely service providers (SPs), identity providers (IdP) and users. Users register with the IdP to obtain their OpenID identifiers. Users submit their OpenID identifiers to the SPs (these service providers have adopted OpenID) they would want to access their service via a secured channel. SPs forward the requests to the IdP to confirm the identifier of the users. Once the OpenID identifiers of users are confirmed, the IdP sends the verification of the respective SPs for onward authentication and access to be granted to the users.

OpenID allows you to use an existing account to sign in to multiple web servers without the need to create new passwords [20]. Created in the summer of 2005, this approach has been adopted by reputable organizations including Google, Facebook, Yahoo, AOL, Novell, etc. The major drawback of the single sign-on authentication is the fact that the trusted third party has to be involved in every user authentication session which does not scale. In [19], Armando et al. also identified one other drawback based on the assumption that the IdP cannot be compromised and the fact that SPs trust the IdP to generate authentication assertions. Another drawback with OpenID is the use of SSL connections in every interaction within the framework of OpenID. The integration of SSL in the implementation of this authentication scheme is to make it suitable and fit for the corporate environment. However, SSL is based on traditional public-key cryptosystem thus imposes heavy computation cost on any device which uses it [18]. This scheme will therefore not be suitable for the kind of devices that are usually deployed at the network at the edge of the network.

Elliptic Curve Cryptography (ECC) and Bilinear Pairing Cryptography (BPC) has been used extensively to resolve the many drawbacks faced in the attempt to secure edge devices using conventional PKC. Traditional PKC such as RSA is designed based on keys sizes between 1,024 to 4,096 bits. The larger the key size, the longer it takes to generate and the more secure it is. Elliptic Curve cryptosystem (ECC) provides the smallest key size per equivalent strength of any variant traditional public-key cryptosystem, including the discrete logarithm problem in a multiplicative group [17]. To achieve the security level

provide by RSA – 1024 bit modulus, ECC uses 160-bit. Similarly, ECC needs 224 bits modulus to achieve comparable security provided by RSA_2048 bits. ECC derives its security robustness from the hardness the discrete logarithm problem based on an operation on points over an elliptic curve [21].

Hong et al. [14] proposed a Shared Authority based on the Privacy-Preserving Authentication Protocol scheme which aimed to enhance data accessing and sharing between two users whiles protecting their privacy. They employed attribute-based access control and proxy re-encryption mechanisms jointly applied for authentication and authorization. Tsai and Lo in [18] based on bilinear pairing cryptosystem and dynamic nonce generation proposed an authentication scheme for data security and privacy preservation in distributed mobile cloud computing environment. This scheme enables users to access the services of multiple service providers using just one private key. Although the scheme supported mutual authentication, key exchange, user anonymity, and user untraceability, the scheme was found to have a lot of fundamental flaws such as service provider impersonation attack, violation of user anonymity, known session-specific temporary information attack and biometric misuse. In [17], an attempt was made by Amin et al. to propose a scheme that provided solutions to the flaws in Tsai-Lo's scheme. Based on elliptic curve cryptography, the enhanced user authentication and session key agreement scheme allowed users to access services of different cloud service providers without the use of a private key. Furthermore, Odelu et al. [13] proposed a scheme that provided session key security and strong credentials' privacy. The scheme also provided an enhanced secured system that is more robust against well-known attacks such as impersonation attacks and ephemeral secrets leakage attack identified in the scheme proposed by Tsai-Lo. Various schemes [22-24] have also been proposed to achieve authentication in various edge computing environments to achieve authentication between users and service providers without the interference of the trusted third party while also achieving privacy and user anonymity.

One thing common to all these schemes is the inclusion of a trusted third party (TTP) to register and issue parameters to be used in the subsequent authentication process. Efforts have been made in recent research works to exclude the TTP from partaking in any of the authentication steps involving service providers and users. Nonetheless, in rare cases, the TTP is called upon to partake in any attempt to revoke the access of any malicious user. A typical authentication infrastructure for authentication in recently proposed schemes involving a TTP is seen in fig. 1. Service providers and users will first have to register with the TTP to be issued private, public keys or any related parameter for subsequent processes.

2.3 Limitations of this approach

There are a few drawbacks to this means of authentication in the edge computing architectural framework [25, 26]:

1. Edge computing thrives on a decentralized approach to make available their resources and services to edge devices. The inclusion of a TTP in the framework presents another centralized element which in principle is not in line with the idea of edge computing. Including a centralized entity introduces a single point of failure which could affect the availability of the network. Once the TTP is compromised, the entire network authentication process will be affected. This means all information of subscribed users could be susceptible to malicious attackers.

2. Again, a user once registered by the TTP will have to authenticate to each service provider. Authenticating independently to each machine increases the overhead of the system and doesn't support scalability.

The inclusion of a TTP in the setup is always premised on the assumption that the communication channel between the edge servers (service providers), edge devices (users) and the TTP is secured. In the real world, this assumption may not always hold. Just like information exchanged between service providers and users are susceptible to manipulation, the secret and critical information exchanged between the TTP and any of the aforementioned elements can also be manipulated to compromise the entire security of the network. Hence this assumption must be dealt with to make the implementation of any authentication scheme more practical in the real world. One other major security challenge with edge computing is securing sensitive information against internal adversaries i.e. user privacy leakage [3]. Most security schemes have been designed to protect the network from external attackers. Nonetheless when the attackers have privileges and can abuse these privileges and tamper with data and services, then it becomes difficult to deal with.

Ensuring users do not misuse assigned resources to sabotage the network can be a difficult target to achieve. A malicious user within a virtual environment based on his privileges can hijack and exploit resources available to the virtual machines. Malicious virtual machines can equally execute malicious programs that do not target the edge data center within which they are deployed but other local environment. Such activities include cracking passwords, host botnet servers, exploiting vulnerable IoT devices, etc. [1].

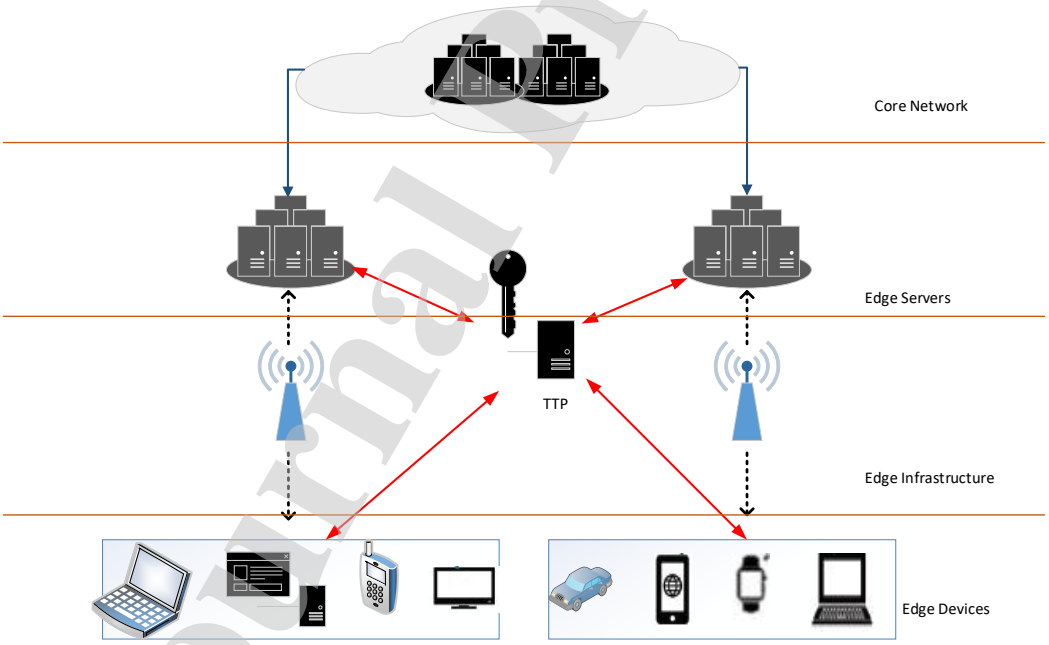

Fig. 1 Authentication model involving a TTP in recent authentication schemes. ⟷ Secured wired communication channel ⟷ Secured Communication channel; -------> Insecure wireless communication channel

3.0 Blockchain: Principles and Strengths

The blockchain (BC) technology provides a platform for the transfer of value between entities without the involvement of any trusted third party. The noninvolvement of any intermediary between transacting entities means that processes leading to the completion of the transaction will be faster compared to the opposite. The transactions between entities are approved through the use of a distributed consensus protocol. The type of consensus protocol depends on the type of blockchain network and the attack vector that the network operator adopts [27]. That notwithstanding, the principle behind any type of consensus protocol is to avoid any form of collusion between nodes to alter any data. Transactions approved within a preapproved time frame are compiled into blocks and distributed among all entities.

BC also uses asymmetric cryptographic key principles to secure data blocks. The use of asymmetric cryptography within the blockchain setup ensures authentication, integrity, and nonrepudiation while bringing authoritativeness in all transactions between elements of the network [27]. Every entity gets to have a pair of private and public keys. An entity is identified within the network by other entities through the public key. Transactions are digitally signed and advertised to other entities by the user applying the private key. The digital signature ensures that only the owner of the private key could have sent the transaction. The use of the data signatures and encryption mechanisms eliminate the possibility of man-in-the-middle, replay and other attacks [28]. A combination of the private and public keys brings an exclusiveness and peculiarity to every single transaction.

Blockchain technology is distinctive because it is characterized by some fundamental attributes and principles that make it quite adaptable to our system. These attributes include:

- Peer-to-peer: The absence of a TTP allows all nodes to connect in a manner that rights and privileges are the same among all nodes. Network control and responsibilities are distributed among peers thereby enhancing network security [29].
- Trustless: Trust in traditional networks is very critical. Maintaining trust in these centralized clouds and edge networks have been an arduous task [4]. Nonetheless, in blockchain, nodes do not have to trust each other before they can transact. Nodes rely on the efficiency of the cryptographic principles to guarantee the security of their transaction.
- Transparency: In blockchain technology, every transaction can be checked, audited and traced from the system's initial transaction [30]. All users have access to the same ledger thereby making it extremely difficult for an attacker to have access to manipulate every node's ledger.
- Transaction Verifiability: Every node within a blockchain network partakes in the process to verify the authenticity and integrity of a transaction either as a miner or as an observer. This enforces trust within the network architecture. Once a transaction is approved by the majority of the nodes, it can then be added unto a new or an already existing block.
- Tamper-proof: Blocks in the BC are designed such that to successfully alter the content of a block, the attacker has to also alter all the subsequent blocks. Such an act also needs the consensus of the majority of the nodes in the network. Thus, an element of the security inherent in the BC is the difficulty in modifying blocks without rendering previous blocks invalid due to the changes in hash values.

Nodes within a Blockchain network can be assigned different roles such as [31]:

- Light node: This node stores a portion of information recorded in the blockchain network.
- Full node: This node keeps a record of all the information transpiring on the blockchain network.
- Mining or forging node: This node processes transactions, compile these approved transactions into blocks, adds the block to a blockchain and subsequently broadcasts the block to the entire network.

3.1 How Blockchain works

A blockchain network can either be permissionless (public) or permissioned (private). In a permissionless BC, there is no control over who can join the network. All nodes in a permissionless BC participate in broad consensus leading to the verification of a block. Due to the nature of the permissionless or public BC network, consensus protocols have to be stricter to avert any attempt of collusion between nodes. A classic example of a permissionless BC network is Bitcoin which operates using the Proof-of-Work consensus method.

Permissioned networks are controlled and regulated. The architecture is not a typical peer-to-peer network since some elements within the network have more control than others. This scenario allows a limited number of users to have access to partake in the consensus process while the others participate as observers. Every node in the network is identifiable within this setup. Due to the regulated nature of the network, the consensus algorithms used are simpler and less intense compared to the public network. A smart contract is an example of a permissioned network while variants of Byzantine Fault Tolerance (BFT) are usually used as a consensus algorithm. A smart contract is a set of computer protocols or programs that automatically executes contracts considering set of predefined conditions [30]. Smart contracts can be used to define the logic for the application for the exchange of cryptocurrency for every transaction exchanged.

With the types of architecture discussed, we can now look at how blockchain fundamentally works.

1. Using their pair of cryptographic keys, two users can initiate a transaction. The transaction is then broadcast to all neighboring nodes to confirm the validity of the received transaction. The validity of the transaction is based on the public addressable key and its accompanying signature (private key).
2. The neighbors confirm the validity of the received transaction and forward to other nodes close to them. The transaction will be received by all nodes after some time.
3. Once the transaction meets the consensus criteria, it is added to other validated transactions within the network and subsequently ordered and packaged into a timestamped block through the process of mining.
4. During the process of mining, the participating nodes (miners) go through a rigorous, computationally intensive task based on the consensus protocol (e.g. Proof-of-Work) used. Mining is an integral component of the process as the new block is generated [30].
5. The miner that completes the Proof-of-work then gets the opportunity to broadcast the generated block to all nodes as well as the opportunity to add the new block to the chain once verified by all the other nodes.
6. Every block is made up five of components: index, timestamp, data, the hash and the hash of the previous block [32]. The index provides a number for the block. The time of the block creation can be known through the timestamp. The data collected in a block can be different data types depending on the blockchain applications. The hash function converts a block and its contents to a unique fixed-length output which is exclusive only to that block [29]. Every existing block has to reference the hash of the preceding block. This presents an ordered arrangement of the blocks creating a chain of blocks. Changing the hash of any block invalidates the subsequent blocks since the hash values of the previous blocks will have to change. This single feature enhances the security of the block protecting it against any mutability and tampering.

The workflow of blockchain as described above are summarized in figure 2.

3.2 Blockchain – Edge Paradigm Computing Integration

In recent years, the blockchain and its integration in the broad Internet of Things (IoT) have been extensively studied with many reviews done to discuss both the positives and the challenges [26, 27, 29, 33]. However, not much work has been done narrowing the discussion down to the area of edge paradigm. Edge computing was introduced to address the many bottlenecks of traditional cloud computing. Nonetheless, the various variants of edge computing have been confronted with fundamental challenges that negatively impact the implementation of the network.

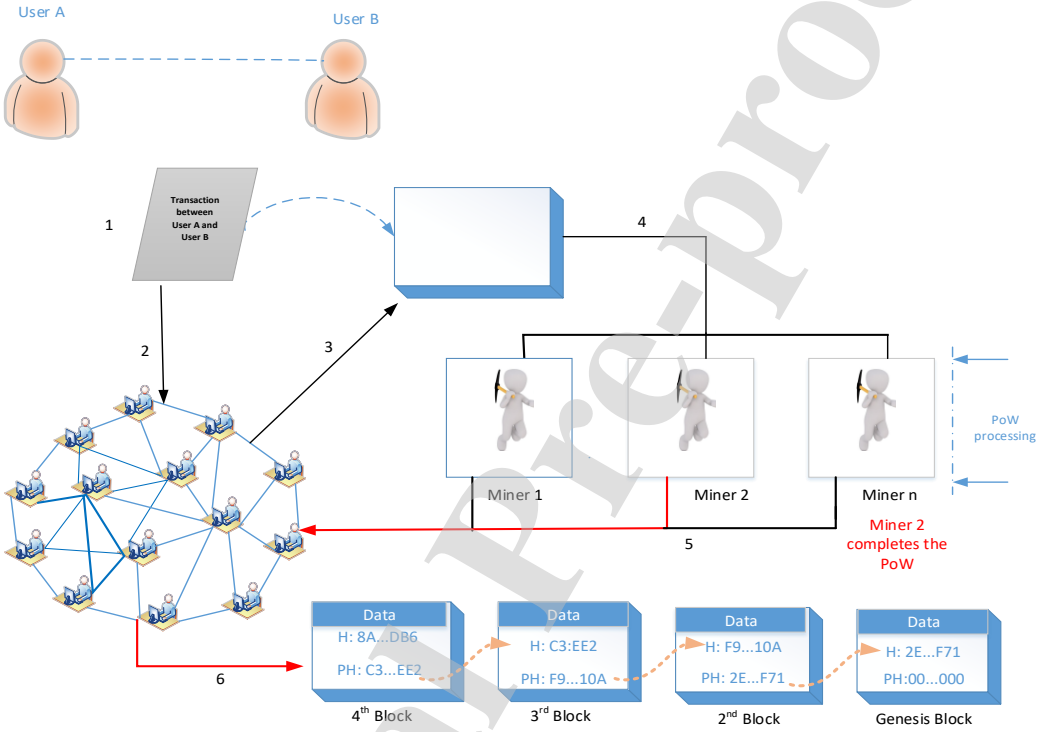

Fig. 2. Summary of the workflow in Blockchain

Blockchain technology due to its inherent characteristics as discussed above provides the platform to deal with most of these challenges of edge computing. Blockchain allows designing a comprehensive robust peer-to-peer network while eliminating any single point of failure in the setup. Inherent in the system are underlying principles of transparency, auditability, immutability and variability which have the potential to enhance service delivery and security in edge computing.

3.3 Challenges with Blockchain - Edge Paradigm Computing Integration

Despite the prospects of BC, coupled with the extent it can be leveraged to improve services in edge computing, there are other BC principles that may not enhance the security and service delivery in edge computing if readily implemented. Among the many integration challenges, the notable challenges relative to our scheme is discussed below:

1. The blockchain technology places a high demand on storage and computational resources of any device it's implemented on [30]. Most edge devices are also noted to be low on computational resources. Devices at the edge are expected to keep a copy of a distributed ledger of all transactions in the network for validation purposes. An increase in the number of blockchain nodes will cause storage problems for these low resources edge devices. It is therefore important the design and

integration of blockchain into any of the edge computing paradigms be done without adversely affecting the traffic flow and overhead in the network.

2. Further, on computation limitation, mining in a blockchain is very computation expensive despite the fact the degree of intensity may differ between network type i.e. permissioned or permissionless [26, 30]. Involving all nodes in the consensus process may not be ideal for the network since the adverse effect started above applies here too.
3. The implementation of privacy relative to edge computing and its devices makes it a very complex situation. Every node must have access to the details of every transaction to validate it. If this principle is to be replicated with the integration into an edge computing environment, encryption mechanisms which are not resource-intensive should be employed to preserve transaction confidentiality between nodes. Traceability in the blockchain is plausible since an interested party can observe patterns and create connections between addresses in a bid to identify the users behind the transactions [27]. Such potential occurrences should be mitigated in any integration of blockchain into edge computing.

4.0 DecChain (DC) Design Principles

In this section, we introduce the design principles of our scheme to deal with the challenges of the blockchain – edge computing integrated platform. DecChain is a progressively decentralized blockchain-edge computing platform built with the foremost aim of enhancing security and service delivery in an edge computing environment. DecChain protocol proposes a novel privacy preservation and authentication scheme fit for edge computing environments without the involvement of any trusted 3rd party in either the registering or any subsequent process in the mutual authentication in the network. Our scheme was built bearing in mind the limitations of edge devices as well as the integration challenges discussed above. In this paper, which is the first of our work into this area, we lay the foundation in our scheme dealing primarily with authentication and privacy in edge computing environments. Thus, for now, we will not introduce principles of smart contracts since that will be subsequently introduced in future works to cater for resource allocation and use.

The principles applied in the design of DecChain which formed the basis of our contribution are discussed below:

1. DecChain was designed based on the principles of permissioned blockchain since every entity in the scheme should be identifiable either by a chosen identity or a public key.
2. Typical of a permissioned blockchain, all entities will not partake in the consensus process. Only edge servers (service providers) which are assumed to be computationally resourced will partake in the process of adding transactions to the network.
3. Edge devices will be treated as transactions during the registration and setup phase as they interact with their respective service providers to be added to the DecChain network. The respective service provider will have the responsibility of broadcasting the identity and public key of the edge device to all other service providers in the network.
4. Once an edge device has been authenticated and validated by all the other providers in the network, the service provider will then send to the device all public parameters, the identities and public keys of the other service providers.
5. Transactions between an edge device and a service provider will be treated as a sub-transaction will be stored only between the two devices involved. Thus, every edge device will be deployed as a light node. Validated nodes only keep records of sub-transactions relative to itself.

6. Every service provider, on the other hand, will keep records of all sub-transactions between itself and all edge devices connected to it. Again, service providers will keep a distributed ledger of public keys of all edge devices and other service providers in the network. Service providers will also keep records of all transactions exchanged between themselves. Thus service providers will be deployed as full nodes as well as miners.
7. Every sub-transaction between a node and a service provider will have a transaction id which is the hash function of the content of the sub-transaction into a fixed length and exclusive to only that sub-transaction. This will be referenced by the subsequent transaction to help mitigate any form of intrusion attack.
8. The scheme is designed in such a way to maintain user anonymity and untraceability despite the broadcasting of the chosen identity and public key using nonces and data encryption.

4.1 DecChain Architecture

DecChain scheme involves two elements namely, the edge servers (service providers)and mobile users Similar to a blockchain setup, the decision to allow particular user access to a particular server depends on consensus validation of a majority of all service providers in the network. Once a service provider is enabled as a DecChain element, a layer is created within its setup to accommodate the activities relative to this architecture. The DecChain layer created above the cloud platform as seen in figure 4, will accommodate the distributed ledger that will have all the user identities and public keys once validated, and the sub-transactions and other related information related to any discourse between a user's edge device and the service provider.

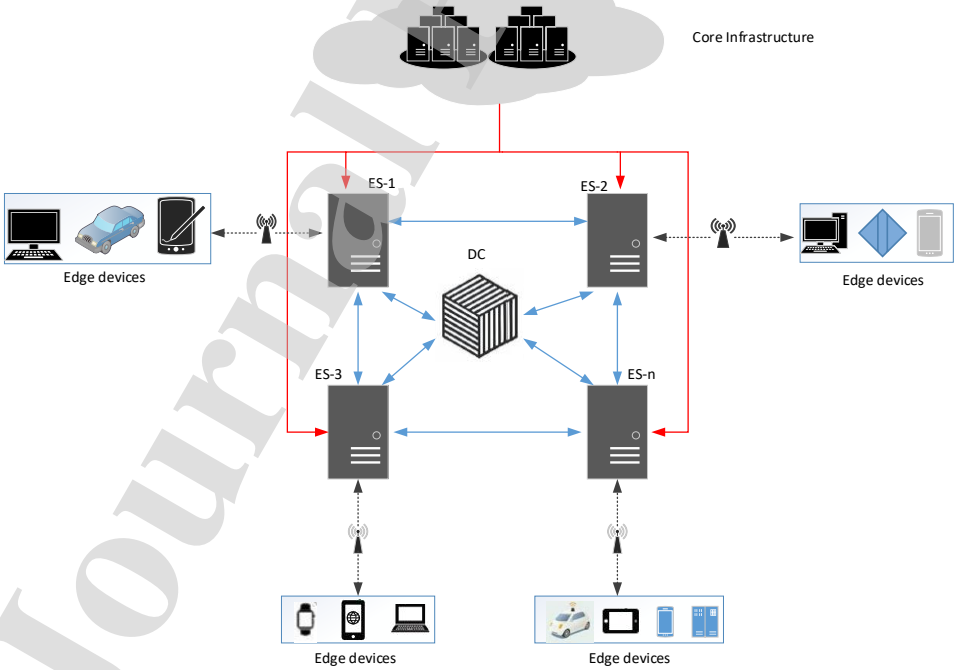

Fig. 3: DecChain Architecture. ES: Edge Server; DC: DecChain Platform: ◄——► Wired connection between cloud and edge servers ⟷: Wired connection between DecChain enabled edge servers; ⋯⋯► Wireless connection between DecChain enabled edge devices and Servers

4.2 Node Registration, Login, and Authentication

Every node upon installing the DC client will have the option to be configured as either a service provider or an edge device as seen in figure 5. Once a node meets the requirements to be configured as a service provider, it will become a genesis block to initiate its trail of nodes joining in the form of transactions. The genesis nodes will all be connected as seen in figure 3. Nodes that are configured as edge devices will be added to the trail of nodes (as a transaction) of the service provider it wants to access.

Let P be a generator of the additive cyclic group G_1 while G_2 be a cyclic multiplicative group, where p is the prime order of G_1 and G_2 and $g = e(P, P) \in G_2$. Let the bilinear map $e: G_1 \times G_1 \rightarrow G_2$ satisfy the following properties [13, 17, 34]:

1. **Bilinearity**: For all values of $M, N \in G_1$ and for all random numbers $a, b \in Z_p$, then, $e(aM, bN) = e(M, N)^{ab}$.
2. **Non – degeneracy**: There exist $M, N \in G_1$, such that $e(U, V) \neq 1$, where 1 is the identity element of G_2.
3. **Computability**: There exists an efficient algorithm to compute $e(M, N)$ for all values of $M, N \in G_1$.

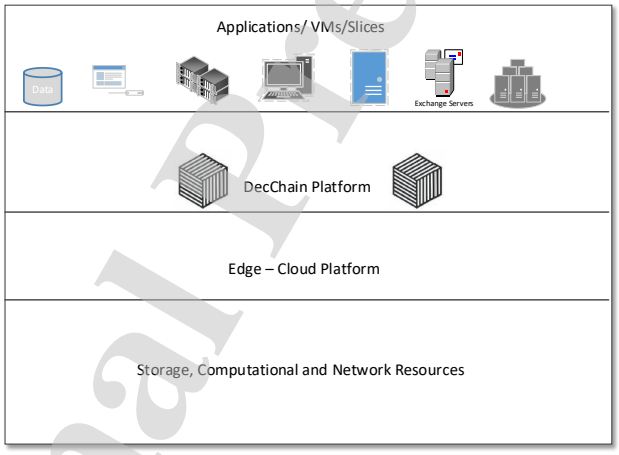

Fig. 4: DecChain enabled Service provider architecture

Table 2: Notations Used in our Scheme

Symbol	Description	Symbol	Description
U_i	$i-th$ User	PK_i	Public key of user U_i
SP_j	$j-th$ Edge Server (Service Provider)	PK_j	Public key of SP_j
S_i	Secret key of U_i	SK_j	Secret key of SP_j
ID_i	Chosen ID of user U_i	PW_i	Password of U_i
f_i	Biometric of U_i	$H()$	Hash function
P	Generator of the group G	p	Large prime number
$a, b, c \in Z_p$	Random numbers	α	Biometric key
β	Public reproduction parameter	γ	Error tolerance of fuzzy extractor
E	Encryption		

Within DC framework, three hash functions are computed: $H_i: \{0,1\}^* \rightarrow \{0,1\}^n, i = 1, 2, 3$. To safely secure information stored on mobile devices within our scheme, we further incorporated fuzzy extractor bio-cryptosystem [35, 36]. The role of the fuzzy extractor bio-cryptosystem is explained in Appendix A. The public parameters to be generated and stored on the service providers for onward forwarding to all nodes that seek to access their services are $\{H_1, H_2, H_3, P, p, e, G_1, G_2, g, \ Gen(.), Rep(.), \gamma\}$.

Step 1: U_i installs DC client, chooses an identity ID_i and forwards the id to the service provider SP_j.
Step 2: SP_j receives ID_i and broadcasts the identity of U_i to all the other service providers for validation. Once the majority of the nodes validate the request, SP_j forwards the public parameters of the network to the user U_i.
Step 3: Upon receiving the public parameters, user U_i chooses $a \in Z_p$ and computes its secret key S_i as:

$$S_i = \ a \cdot g^a \ mod \ p$$

The public key PK_i is computed as:

$$PK_i = S_i P.$$

U_i then forwards the computed public key to SP_j.
Step 4: SP_j then adds the identity of U_i and its public key to its registry and forwards the same information to all other service providers to be validated for onward addition to the trail of already validated users in the distributed ledger. SP_j then forwards to U_i all the identities and public keys of all the service providers within the network $[(SP_1, PK_1), (SP_2, PK_2), (SP_3, PK_3), \dots, (SP_j, PK_j)]$.
Step 5: Upon receiving the identities of the service providers and their respective public keys, U_i then computes $(\alpha_i, \beta_i) = Gen\ (f_i)$. U_i then encrypts and saves the following parameters securely as

$$D_i = E_{H_2(ID_i \| PW_i)}[(SP_1, PK_1), (SP_2, PK_2), (SP_3, PK_3), \dots, (SP_j, PK_j)]$$

$$d_i = E_{H_1(\alpha_i)}(ID_i \parallel PW_i \parallel S_i)$$

The encryption of the identities and public keys are computed using the identity of the user and password known only to the user as encryption keys. Thus, the details of service providers and their public keys cannot be retrieved without $\{ID_i, PW_i, f_i\}$. Malicious attackers in the event of loss and theft of mobile devices cannot retrieve these details without the user information.

4.3 Exchange of Information between Nodes and Service Providers

Once a node is validated and added to the other validated nodes connected to the service provider, these nodes can then generate signatures and further exchange transactions with its service provider or other service providers through the service provider it is connected to. Validated users do not also have to go through the same process of registration again once they connect to the network through a different service provider.

4.3.1 Digital Signature Generation

At the center of our proposed scheme is the generation of digital signature to augment the authentication and integrity of both user identity and data. The generation of the signature was based on the elliptic curve cryptosystem with the computation involving the content of messages transmitted between nodes. The processes involved are generation and verification.

The generation phase which leads to signing takes as input the following details of the sender, U_i: ID_i, PK_i and m.

U_i selects any nonce $\{a, b, c\} \in Z_p$ and computes the following:

$$f = (H_1(m).ID_i.P) \in G$$

$$d = (f.PK_i\,.a) \in Z_p$$

The generated signature on message m is therefore, $Sig_i = (f, d)$.

The verification phase occurs on the recipient's device. Upon receiving the file, recipient U_j computes f. Based on bilinearity property, the validity of the signature holds if $e(ad, bG) = e(f.PK_i, G)^{ab}$. Transaction file is accepted if the statement holds and vice versa.

4.3.2 Transaction Creation and Exchange

1. When a node or user wants to exchange a transaction with a service provider it is connected to, it creates a transaction file F_t, as seen in figure 6. U_i then selects a random number $b \in Z_p$ and computes transaction to be exchanged as:

$$H_2(\mathrm{b} \,\|\, \mathrm{PK}_i \,\|\, Sig_i \,\|\, E_{PK_j}(\mathrm{F}_t) \,\|\, PK_j \,\|\, \mathrm{H})$$

2. Once the service provider receives the transaction from U_i, the previous hash value, H is compared to the hash value of the last transaction between the two elements. The file is dropped if the previous hash value is different from the hash value of the last correspondence exchanged between them.
3. The source details are then confirmed by checking if the public key corresponds to both the signature and the identity provided. The transaction will be dropped if any of these checks do not correspond.
4. The service provider can then check the destination to confirm if it is the intended receiver of the encrypted transaction or just a forwarder. If the service provider is the recipient of the transaction, it then confirms the authenticity of the signature. If the signature is valid, it decrypts the transaction which has been encrypted using the public key of the recipient of the file otherwise file is rejected.

When a validated user wants to send a transaction to a different edge server (service provider), it does so through the server it is directly connected to. In figure 7, user U_i intends to send a transaction to the service provider SP_4.

5. When a service provider SP_1 takes delivery of the transaction and realizes it is not the intended recipient after going through steps 1 – 3, it repackages the transaction by dropping the source and hash values. SP_1 then selects a random number, computes and signs it with the private key:

$$H_1(\mathrm{b} \,\|\, \mathrm{Sig}_{SP_1} \,\|\, E_{PK_4}(\mathrm{F}_t) \,\|\, PK_4 \,\|\, \mathrm{T}_i)$$

   where T_i is the transaction id. SP_1 then forwards the repackaged transaction to all its neighboring edge servers.
6. When a service provider receives the transaction, it first confirms the authenticity of the signature of the sender. The transaction is dropped if the signature is invalid otherwise, the server confirms if it is the intended recipient. If it is the not intended recipient, it forwards it to its neighboring nodes. Within a short period, every service provider will have taken delivery of the transaction.
7. Service providers will discard any transaction with a T_i of an already forwarded transaction. This will prevent servers or service providers from flooding the network with already received transactions.

Once the intended recipient receives the transaction, F_t is decrypted and the validity of the user is confirmed by checking the authenticity of the digital signature. The transaction is dropped if there is any discrepancy, otherwise, the information in the transaction is accessed. F_t also contains the authentication duration, within which the authenticated user can access the resources on the service provider until the access is revoked and a new authentication is required.

8. When the recipient is done accessing whatever information the sender wanted to communicate in the transaction, it adds the transaction to a block.
9. An acknowledgment is computed by and sent to the sender along with the means the transaction was delivered.

$$H_2(\mathrm{b} \,||\, \mathrm{E}_{PK_i}(Sig_{SK_4}(ACK)) \,||\, PK_i)$$

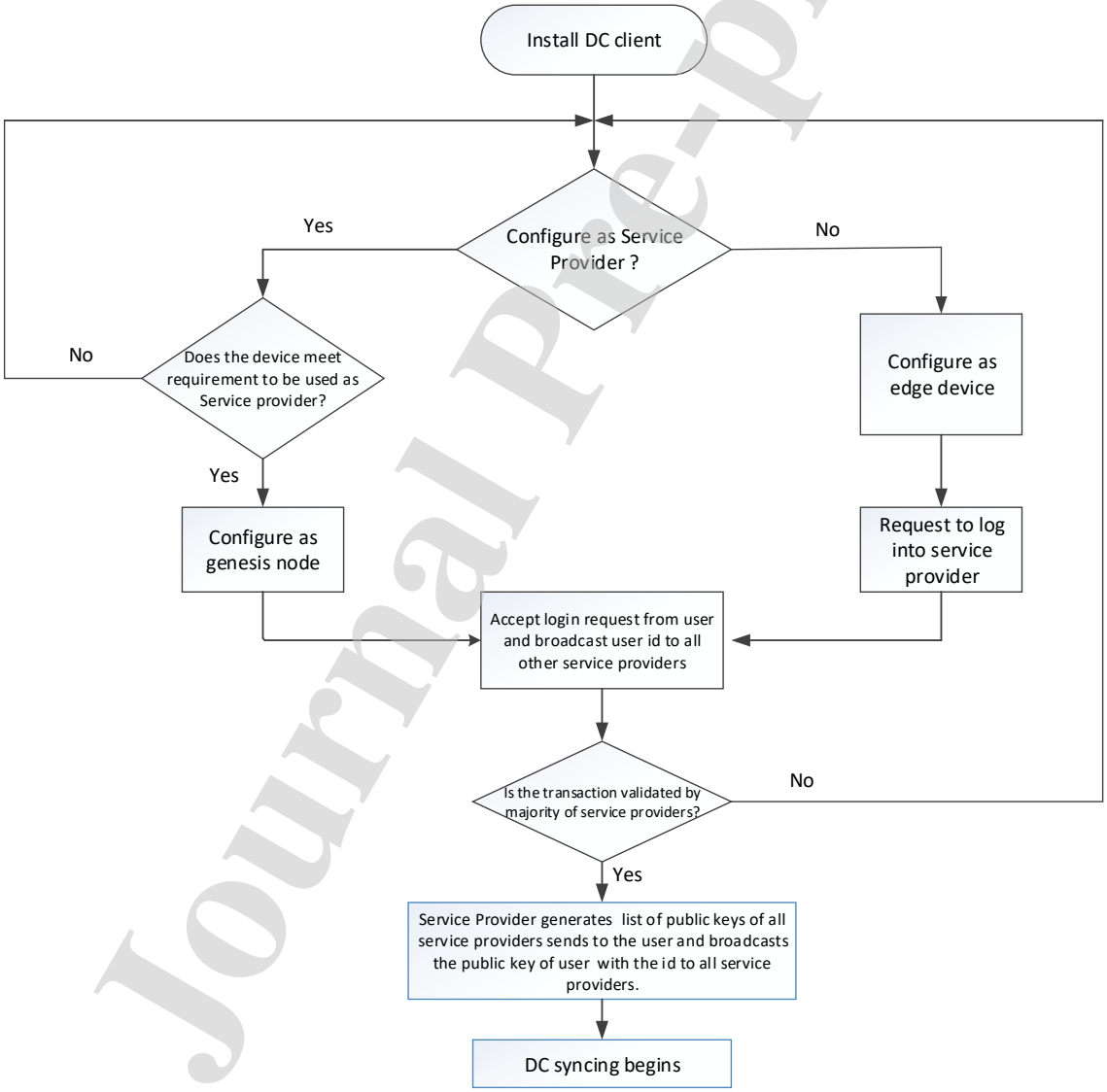

Fig. 5: Flow chart diagram for node initialization in DecChain

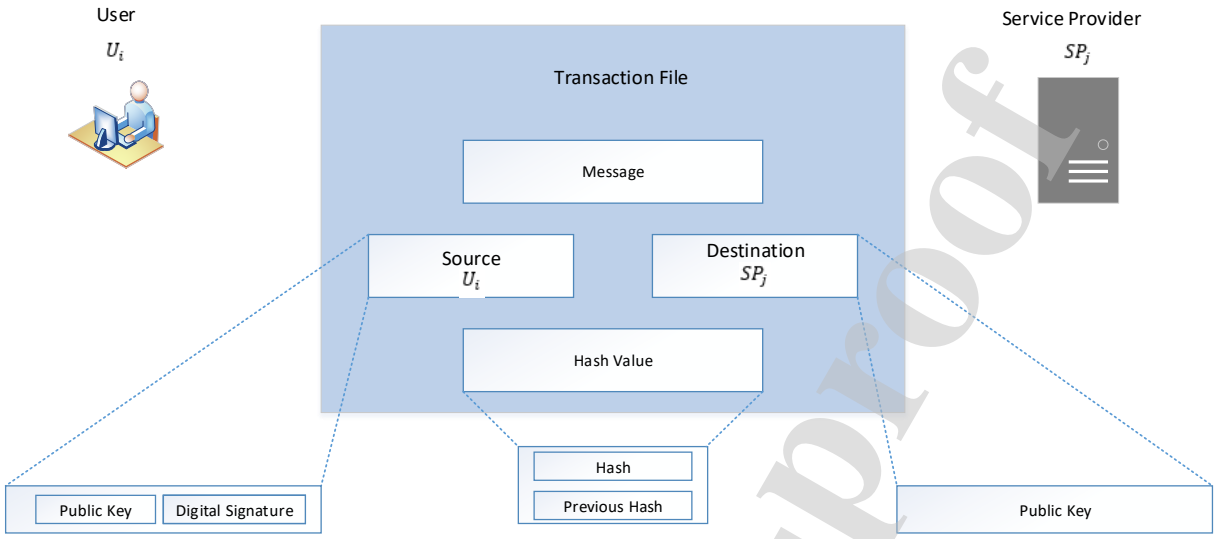

Fig. 6: Transaction file to be exchanged between User U_i and Service Provider SP_j.

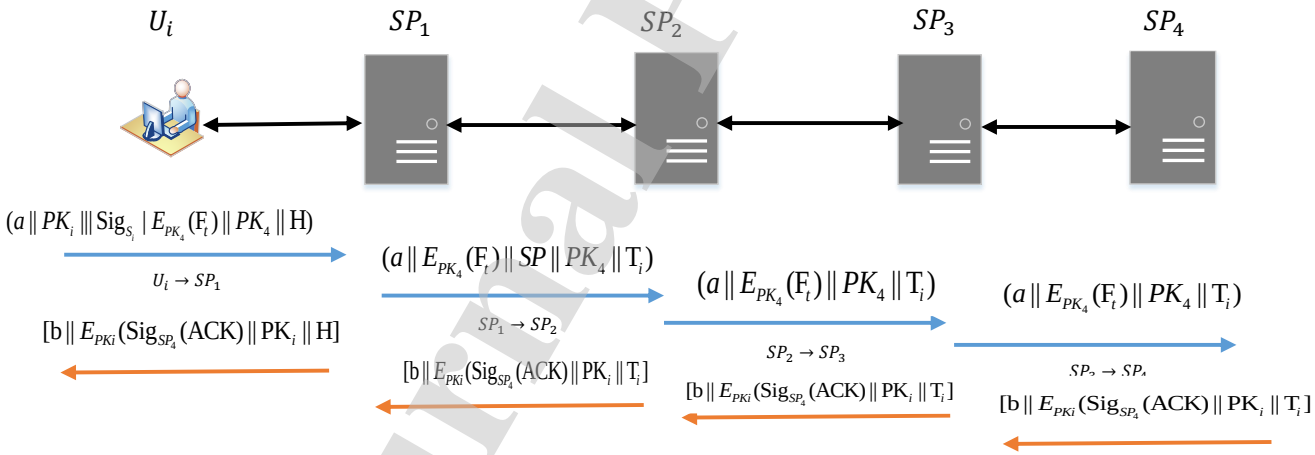

Fig. 7. Summary of the process of exchange of transaction between user U_i and service provider SP_4 across 3 service providers, namely SP_1, SP_2 and SP_3

5.0 Cryptanalysis of DecChain Scheme

In this section, we present an analysis on our scheme and its ability to withstand known attacks that may emanate from both internal or external attackers. The discussions were made based on the following assumptions [17]:

1. A polynomial adversary $\mathcal{A}$ has control over transactions transmitted over unreliable communication links and can, therefore, delete, modify, re-route, replay captured transactiond.

2. It is assumed cryptographically that the secret or private key of any entity in the network and any random nonce have higher entropy. Thus, these two parameters can not be easily guessed by $\mathcal{A}$ within the polynomial time.
3. $\mathcal{A}$ knows all the public parameters communicated over the network.
4. Adversary $\mathcal{A}$ can alter the hash functions on a transaction exchanged between a node and a service provider as well on a block added to the blockchain.
5. All encrypted transactions exchanged on the network cannot be readily decrypted by $\mathcal{A}$ in polynomial time. However $\mathcal{A}$ is allowed to capture any information leakage.

5.1 Privacy of User Credentials and Impersonation Attack

The scheme is designed to provide maximum protection for the user's credentials in the case of loss or theft. Users are also protected from any form of impersonation attack from any malicious attacker within our scheme. Assume the adversary $\mathcal{A}$ manages to intercepts all transmitted transactions exchanged between entities, it will be difficult for A to access credentials of the U_i such as the identity ID_i, password PW_i, and the secret key S_i since they are secured by the encryption of the hashed value of the biometric details, $E_{H_1(\alpha_i)}(ID_i \parallel PW_i \parallel S_i)$. }. f_i is needed as an input together with β in the deterministic reproductive function $Rep(.)$ to generate a biometric key α before a possible retrieval of the $\{S_i, PW_i, ID_i\}$ for onward retrieval of the identities and public keys of the service providers. It will be therefore highly difficult for $\mathcal{A}$ to impersonate U_i to initiate any kind of attack since you will need the user credentials as well the intended recipient details and all this information are secured within our scheme.

5.2 User Untraceability and Anonymity

The introduction of random nonces makes every transaction exchanged between users fresh in each session. Thus, it is therefore impossible to link transactions from session to session. Thus the identity of a user can not be traced based on intercepted transactions by adversary $\mathcal{A}$. In the unlikely event that any polynomial-time adversary is able to access a transmitted transaction despite the presence of a random nonce, the identity of a user or service provider will be not readily accessible to the adversary since the identities of every entity is highly secured. Computationally it will be difficult for $\mathcal{A}$ to reveal or trace the identity of any given user within any given polynomial-time. Thus, our scheme is secured against any attempt to reveal the identity of a user or use intercepted sessions as a means of tracing a particular user's transactions.

5.3 Service Provider Impersonation

The DecChain scheme is designed in such a way to make impersonation highly difficult from both sides of the network. As much as possible the identities of servers are not transmitted without valid encryption. Assuming $\mathcal{A}$ captures transactions transmitted between a server and any entity, it becomes difficult to impersonate a server due to the presence of a digital signature which can be computed with the private key of the service provider. Any received transaction without a valid signature will be discarded. Furthermore, the referencing of hash functions in interactions between service providers and nodes connected to it makes it difficult for any adversary to initiate any discussion with these nodes.

5.4 Replay and man-in-the-middle attacks

In the event $\mathcal{A}$ captures any transmitted transaction between U_i and SP_j or between two service providers and intends to replay them, the network requires a reference to the hash value of the previous transaction between the entities. Hence any attempt a transaction without a correct hash value referenced will be identified. For any exchange between service providers, the presence of the index T_i, allows the servers to know when a transaction is replayed. Hence our proposed scheme is secured against replay and man-in-the-middle attacks.

6.0. Further Cryptanalysis using AVISPA

AVISPA is a suite of applications for analyzing models of security protocols. These protocols to be analyzed are written in High Level Protocol Specification Language (HLPSL) [37]. An HLPSL specification is translated into Intermediate Format (IF) through a translator called hlpsl2if. AVISPA has four backends namely On-the-fly Model-Checker (OFMC), CL-based Attack Searcher (CL-AtSe), SAT-based Model-Checker (SATMC) and Tree Automata-based Protocol Analyzer (TA4SP). The backends can read the HLPSL specification in the Intermediate Format (IF).

We chose to implement our scheme under the OFMC backend of AVISPA. This is because OFMC is efficient in detecting attacks as well as verification of protocols [38]. The discussion of the simulation results is done taking into consideration the goals for the simulation as earlier listed. The implementation of the cryptanalysis on the various elements in the network is explained in Appendix B.

Our scheme was modeled on AVISPA using Dolev-Yao (dy) channel. This allows for the inclusion of an intruder, i, to test the vulnerability of the scheme as transaction traverses the network. The intruder i can manipulate any transmitted transaction and retransmit the transaction to any other user [37]. The simulation sought to achieve the following aims:

1. Dolev-Yao model check
2. Replay attack check
3. Verify the executability check in non-trivial HLPSL specification

- Dolev-Yao model check: The chosen backend checks the possibility of intruder i to initiate a man-in-the-middle attack. Based on the results, the intruder failed to initiate such an attack in both phases of the scheme thereby confirming the robustness of our scheme against man-in-the-middle attacks.
- Replay attack check: The intruder i was given knowledge of the transactions exchanged between the various parties in both phases. Irrespective of the knowledge i had about the networks, it failed to initiate any replay attack in both phases of the simulation.
  Executability check: The executability check checks to see if all the goals declared in the simulation are achieved. From figures 8 and 9, our proposed scheme is safe and therefore passes this test.

The results obtained using the AVISPA tool indicates our proposed schemes is safe against active and passive attacks. The simulation results using the OFMC and CL-AtSe back ends of AVISPA are given in Figures 8 and 9.

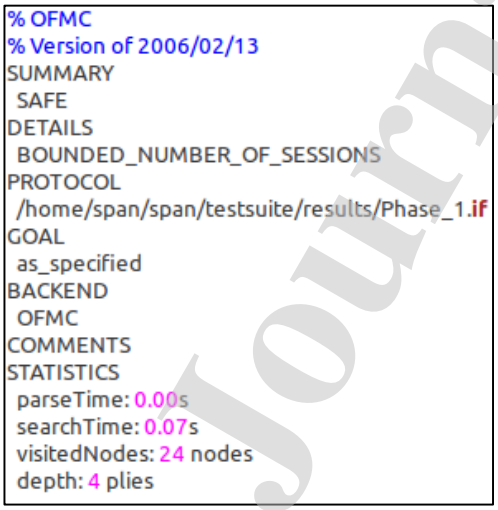

Fig. 8. Simulation results for Phase 1

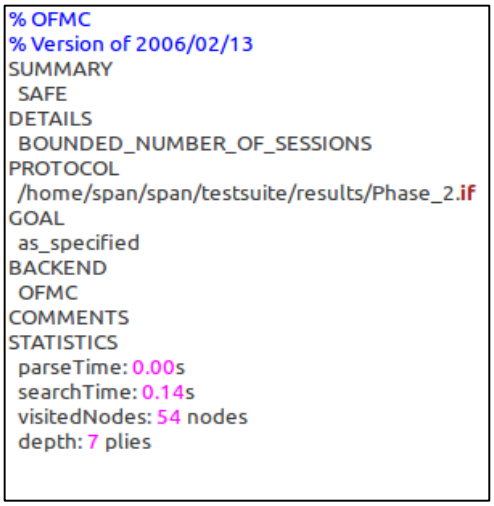

Fig. 9. Simulation results for Phase 2

7.0. Practical simulation using NS2 simulator

We simulated our proposed scheme on an NS2 2.35 simulator to check the viability and practicality of our scheme. NS2 is an event-driven simulation tool that is widely used to check the corresponding behaviors of simulated network protocols [39]. The parameters used in the simulation are listed in table 3. Using three different scenarios across two different phases to access the practicality of our scheme, our focus was to access the impact of our scheme on computational resources of the deployed services providers with an even distribution of mobile users across all scenarios. Not much attention was given to the impact on the computional resources on mobile user devices because the values of the metrics analyzed during all the phases of the simulation were virtually the same.

Table 3: NS2 Simulation Parameters

Parameter	Description	
OS Platform	Ubuntu 18.04.4 LTS	
Phases of simulation	Phase 1	Phase 2
Number of Service Providers SP_j	10 (for scenarios 1, 2, 3)	4 (for scenarios 1, 2,3)
Number of users U_i	Scenario 1: 10 Scenario 2: 20 Scenario 3: 30	Scenario 1: 10 Scenario 2: 15 Scenario 3: 25
Mobility of users	5 metres	
Initial energy e_s of each service provider SP_j	500 Joules	
Initial energy e_u of each user U_i	200 Joules	
Simulation time T_s	1500 seconds	

As done by Odelu et al. [13], we assumed the length of every hash function to be 160 bits long. The randomly generated nonces were assumed to be 128 bits while the transaction id T_i, all digital signatures, and the authentication duration, T_{auth} were all assumed to be 32 bits long. The simulation was done in two phases for all three scenarios with the number of mobile users evenly distributed among the service providers. The first phase of the simulation involved mobile users computing a transaction file F_t and sending it directly to its connected service provider. The service provider then sends an acknowledgment to the sender of the F_t. The second phase of the simulation involved the exchange of transactions between a mobile node and another service provider across two other service providers as seen in figure 7.

In the first phase of simulation, two transactions were exchanged between U_is and SP_js. U_i initiated the transaction with $M_1 = F_t$ to SP_j. SP_j then responded with $M_2 = Ack$. Based on the parameters used in the composition on the transactions, M_1 and M_2 were assumed to be 544 bits and 352 bits respectively. Unlike the first phase of the simulation, the second phase involved the exchange of transactions between $U_i s$ and another SP_j other than the one they are directly connected to. Two transactions were generated again by U_is and SP_4 as $M_1 = F_t$ and $M_2 = Ack$ respectively. Service providers SP_1, SP_2 and SP_3 were configured to be forwarding nodes to forward the transactions to either an U_i or SP_4 as they are exchanged between the entities.

The parameters analyzed for the simulations are load (bps), throughput (bps) and energy consumption (mW). All these parameters were analyzed on all the deployed elements in the simulation i.e. mobile nodes and service providers. The analysis was done based on the average calculation involving Total Packets Sent

(S_p), Total Packets Received (R_p), Bit size of packet (P_s), Initial Energy in SP_j (E_s), Initial Energy in U_i (E_u), Remaining energy (E_r) and Simulation Time (T_s).

7.1. Impact Analysis on Load

The load on the mobile nodes and servers (service providers) as the executed our proposed scheme in the various phases were analyzed. This was done based on the calculation of the average load on both the nodes and servers as:

$$Load\ (bps) = \frac{(S_p + R_p) \times P_s}{T_s}$$

In the first phase of the simulation which was done in three different scenarios, the average values of load on the service providers were 74.48 bps, 152.37 bps and 211.81 bps for scenarios 1, 2 and 3 respectively. The gradual increase in the load on the service providers across the various scenarios is due to the increase in mobile users on the service providers. In the second phase which involved three additional service providers, the average load on the service providers was 43.48 bps, 66.92 bps and 101.79 bps for scenarios 1, 2 and 3 respectively. The involvement of the other providers to act as forwarding nodes between U_i and SP_4 saw a slight increase in the average load on the service providers. The additional increase in the load can, therefore, be attributed to the load on the forwarding service providers which is very low relative to the capacity of the individual service providers.

7.2. Impact Analysis of Throughput

The transmitted bit per unit time defined as throughput. This simulation measured the bits transmitted with the simulation time. The throughput was calculated as

$$Throughput\ (bps) = \frac{R_p \times P_s}{T_s}$$

The average throughput values in the first phase were 104.12 bps, 187.98 bps and 274.56 bps for scenarios 1, 2 and 3 respectively. The increase in throughput values is as a result of the increase in the number of mobile users on the deployed service providers. Thus, with an increase in the number of mobile users, will come with an increase in the number of transmitted transactions between the edge servers and mobile users. In phase two, the average throughput values of the service providers were 51.19 bps, 76.04 bps, and 93.39 bps.

7.3. Impact Analysis of Energy Consumption

The impact of our proposed scheme on the energy of the various nodes is important considering the resource limitation of some edge devices. The computation of energy computation was calculated

$$Energy\ consumption\ (mW) = \frac{E_i - E_r}{T_s}$$

In the first phase, the average energy consumed by the service providers was 4.34mW, 10.27mW and 14.88mW for scenarios 1, 2 and 3 respectively. The energy consumed increased with an increase in nodes across the service providers. In phase 2, the average energy consumed was 3.19mW, 5.57mW, and 9.06mW for scenarios 1, 2 and 3 respectively.

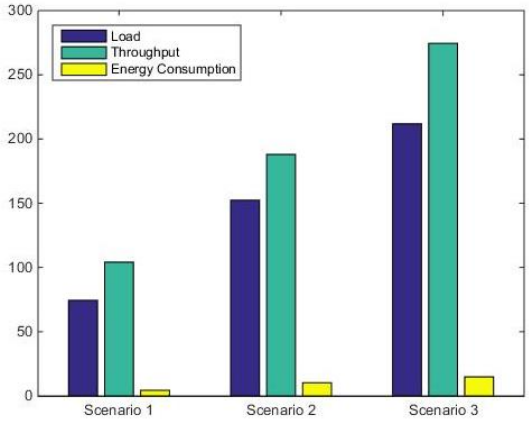

Fig. 16. Phase 1 simulation results for the DecChain on NS2

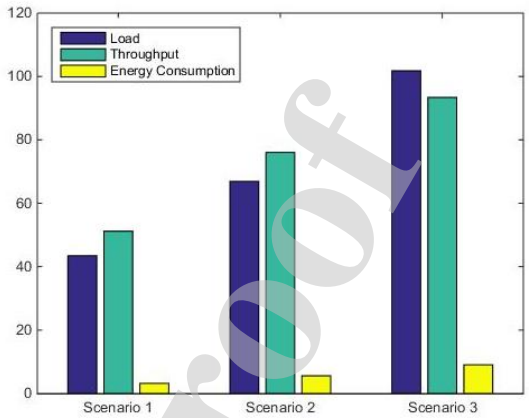

Fig. 17. Phase 2 simulation results for the DecChain on NS2

8.0. Performance Analysis

The performance of our scheme relative to the cost of computational resources on the devices involved in the execution of our proposed scheme was analyzed and compared to a similar scheme proposed by Odele et al. [13] and Amin et al. [17]. The computation of the result of the execution timings used in the comparison was as used in [40]. Based on PBC library [41], the execution timings of both phases of our scheme are also of the group order 160 bits long and base field of order 512 bits long. The execution time required to perform an elliptic curve multiplication on G_1 was denoted as T_{mul} while the execution time for an asymmetric bilinear pairing operation was denoted as T_{par} just as in [13]. To present a comparison based on equal parameters, we ignored light computation such as cryptographic hash function and bitwise XOR operation.

Table 4. Cryptographic operations computation on two platforms

	iPhone 6S	i5 – 4200M
T_{mul}	7.7 ms	0.8 ms
T_{par}	52.1 ms	1.4 ms

The computation of the computational costs of our scheme for both the mobile user U and the service provider SP were done using the cost on iPhone 6s and i5 – 4200M respectively. The cost of computation on U and SP based on our scheme is expressed as $3T_{mul} \approx 23.1\,ms$ and $3T_{mul} + T_{par} \approx 3.8\,ms$ respestively. The computation cost using the scheme proposed by Odelu et al. [13] for both U and SP expressed as $5Tmul \approx 38.5\,ms$ and $4T_{mul} + 2T_{par} \approx 6.0\,ms$ respectively. In the scheme proposed by Amin et al. [17], the computation cost based on their scheme is expressed for U and SP as $4T_{mul} \approx 30.8\,ms$ and $3T_{mul} + 3T_{par} \approx 6.6\,ms$ respectively. The reduction in computation cost in our scheme can be largely attributed to the absence of a TTP as used in the schemes in [13, 17]. Aside from that, our scheme presents a scheme that offers similar security features at a lower cost to the elements involved in the execution of the scheme.

Table 5. Comparison of Computational Cost

Scheme	Amin et al. [17]	Odelu et al. [13]	Ours
U	$4T_{mul}$ $\approx 30.8\,ms$	$5T_{mul}$ $\approx 38.5\,ms$	$3T_{mul}$ $\approx 23.1\,ms$
SP	$3T_{mul} + 3T_{par}$	$4T_{mul} + 2T_{par}$	$3T_{mul} + T_{par}$

	$\approx 6.6ms$	$\approx 6.0\ ms$	$\approx 3.8\ ms$

We also analyzed and compared the security features of our scheme with a similar scheme proposed by Odelu et al. [13]. The scheme proposed in [13] involved three entities namely the Smart card generator (SCG), Service Provider, S_j and mobile user, U_i. Our proposed scheme due to the integration of the blockchain did not involve any trusted third party. Service providers and mobile users generated their secret and public keys. Aside from this distinction and some similar security features and functionalities in both schemes, our scheme provided further features that were integrated to ensure any form of manipulation on transactions exchanged on the network.

Table 6. Comparison of Security Functionalities

Functionality	Amin et al. [17]	Odelu et al. [13]	Ours
Inclusion of the trusted third party	Yes	Yes	No
Provision of user traceability and anonymity	Yes	Yes	Yes
Provision of user credentials' privacy	Yes	Yes	Yes
Provision of secured mutual authentication	Yes	Yes	Yes
Prevention of a man-in-the-middle attack	No	No	Yes
Prevention of replay attack	No	No	Yes
Prevention of user impersonation attack	Yes	Yes	Yes
Provision of data security	No	No	Yes

9.0 Conclusion

The conventional ways of authorization in edge computing platforms have been fronted by many challenges notably scalability and introduction of a single point of failure. The integration of blockchain into this environment sought to eliminate any challenge in the implementation of a purely decentralized and distributed platform. Nonetheless, the incorporation of blockchain into edge computing environments also introduces structural challenges due to the diverse principles of operation and implementation. After identifying and discussing these challenges, we proposed a scheme that leverages the strengths of both operating principles We introduced a comprehensive decentralized approach for enhancing security and privacy applicable in the various paradigms of edge computing. We simulated our scheme on AVISPA to test the robustness of our scheme against man-in-the-middle and replay attacks. The simulation also tested the security of cryptographic primitives used in our scheme. The results indicated our safe against the listed attacks to be initiated by any malicious intruder. We also undertook a practical simulation to verify the efficiency of our scheme relative to cost on computational resources. Again, the results from the simulation confirmed the efficiency of our scheme on computational resources. We compared the performance and security functionalities of our scheme to a similar scheme proposed by Odelu et al. [13]. Our scheme had lower computation cost and additional security features than the scheme compared to.

Acknowledgments
The authors will like to thank all the reviewers and Editors for their constructive review work done on this paper.

Author Contributions
Ernest Bonnah and Ju Shiguang have written this paper and have done extensive research to support our idea. Ernest Bonnah and Ju Shiguang reviewed the work together as well.

Funding
This work was supported by the National Key R&D Program, China (No. 2016YFD0702001) and Modern Agriculture Projects of Jiangsu Province (No. BE201735B).

Conflict of interest

The authors declare that there is no conflict of interest regarding the publication of this manuscript.

References

[1] R. Roman, J. Lopez, and M. Mambo, "Mobile edge computing, Fog et al.: A survey and analysis of security threats and challenges," *Future Generation Computer Systems,* vol. 78, p. S0167739X16305635, 2016.

[2] M. Satyanarayanan, "A Brief History of Cloud Offload:A Personal Journey from Odyssey Through Cyber Foraging to Cloudlets," *Acm Sigmobile Mobile Computing & Communications Review,* vol. 18, pp. 19-23, 2015.

[3] Y. Zhou, Z. Di, and N. Xiong, "Post-Cloud Computing Paradigms: A Survey and Comparison," *Tsinghua Science & Technology,* vol. 22, pp. 714-732, 2017.

[4] J. Zhang, C. Bing, Y. Zhao, C. Xiang, and H. Feng, "Data Security and Privacy-Preserving in Edge Computing Paradigm: Survey and Open Issues," *IEEE Access,* vol. PP, pp. 1-1, 2018.

[5] M. Satyanarayanan, "The Emergence of Edge Computing," *Computer,* vol. 50, pp. 30-39, 2017.

[6] L. M. Vaquero and L. Rodero-Merino, "Finding your way in the fog: Towards a comprehensive definition of fog computing," *Acm Sigcomm Computer Communication Review,* vol. 44, pp. 27-32, 2014.

[7] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proc. 1st Edition of the MCC Workshop on Mobile Cloud Computing*, Helsinki, Finland, 2012, pp. 13-15.

[8] S. Khan, S. Parkinson, and Y. Qin, "Fog computing security: a review of current applications and security solutions," *Journal of Cloud Computing,* vol. 6, p. 19, 2017.

[9] M. T. Beck and M. Maier, "Mobile Edge Computing: Challenges for Future Virtual Network Embedding Algorithms," *Proceedings of the 8th International Conference on Advanced Engineering Computing and Applications in Sciences (ADVCOMP),* pp. 65-70, 2014.

[10] ETSI, "Mobile-Edge Computing – Introductory Technical White Paper," https://portal.etsi.org/Portals/0/TBpages/MEC/Docs/Mobile-edge_Computing_-_Introductory_Technical_White_Paper_V1%2018-09-14.pdfSeptember 2014.

[11] Y. Wang, I. R. Chen, and D. C. Wang, "A Survey of Mobile Cloud Computing Applications: Perspectives and Challenges," *Wireless Personal Communications,* vol. 80, pp. 1607-1623, 2015.

[12] P. P. Ray, "An Introduction to Dew Computing: Definition, Concept and Implications," *IEEE Access,* vol. PP, pp. 1-1, 2018.

[13] V. Odelu, A. K. Das, S. Kumari, X. Huang, and M. Wazid, "Provably secure authenticated key agreement scheme for distributed mobile cloud computing services," *Future Generation Computer Systems,* vol. 68, pp. 74-88, 2017.

[14] L. Hong, H. Ning, Q. Xiong, and L. T. Yang, "Shared Authority Based Privacy-Preserving Authentication Protocol in Cloud Computing," *IEEE Transactions on Parallel & Distributed Systems,* vol. 26, pp. 241-251, 2014.

[15] A. Z. Ourad, B. Belgacem, and K. Salah, "Using Blockchain for IOT Access Control and Authentication Management," in *International Conference on Internet of Things*, 2018.

[16] A. Bhargav-Spantzel, A. C. Squicciarini, S. K. Modi, M. Young, E. Bertino, and S. J. Elliott, "Privacy preserving multi-factor authentication with biometrics," in *Workshop on Digital Identity Management*, 2006.

[17] R. Amin, S. H. Islam, G. P. Biswas, D. Giri, M. K. Khan, and N. Kumar, "A more secure and privacy-aware anonymous user authentication scheme for distributed mobile cloud computing environments," *Security & Communication Networks,* vol. 9, pp. 4650-4666, 2016.

[18] J. L. Tsai and N. W. Lo, "A Privacy-Aware Authentication Scheme for Distributed Mobile Cloud Computing Services," *IEEE Systems Journal,* vol. 9, pp. 805-815, 2017.

[19] A. Armando, R. Carbone, L. Compagna, J. Cuéllar, G. Pellegrino, and A. Sorniotti, "An authentication flaw in browser-based Single Sign-On protocols: Impact and remediations," *Computers & Security,* vol. 33, pp. 41-58, 2013.

[20] OpenID and Foundation. (2018, 25/5/2019). *What is OpenID?*

[21] N. Gura, A. Patel, A. Wander, H. Eberle, and S. C. Shantz, "Comparing elliptic curve cryptography and RSA on 8-bit CPUs," *Cryptographic Hardware & Embedded Systems,* vol. 3156, pp. 119-132, 2004.

[22] N. W. Lo and J. L. Tsai, "An Efficient Conditional Privacy-Preserving Authentication Scheme for Vehicular Sensor Networks Without Pairings," vol. 17, pp. 1-10, 2015.

[23] D. He, N. Kumar, M. K. Khan, L. Wang, and S. Jian, "Efficient Privacy-Aware Authentication Scheme for Mobile Cloud Computing Services," *IEEE Systems Journal,* vol. PP, pp. 1-11, 2017.

[24] K. Mahmood, S. A. Chaudhry, H. Naqvi, S. Kumari, L. Xiong, and A. K. Sangaiah, "An elliptic curve cryptography based lightweight authentication scheme for smart grid communication," *Future Generation Computer Systems,* 2017.

[25] N. Alexopoulos, J. Daubert, M. Mühlhäuser, and S. M. Habib, "Beyond the Hype: On Using Blockchains in Trust Management for Authentication," in *IEEE TrustCom 2017*, 2017.

[26] A. Dorri, S. S. Kanhere, and R. Jurdak, "Blockchain in Internet of Things: Challenges and Solutions," 2016.

[27] K. Christidis and M. Devetsikiotis, "Blockchains and Smart Contracts for the Internet of Things," *IEEE Access,* vol. 4, pp. 2292-2303, 2016.

[28] N. Kshetri, "Can Blockchain Strengthen the Internet of Things?," *IT Professional,* vol. 19, pp. 68-72, 2017.

[29] M. Maroufi, R. Abdolee, and B. M. Tazekand, "On the Convergence of Blockchain and Internet of Things (IoT) Technologies," 2019.

[30] A. Reyna, C. Martín, J. Chen, E. Soler, and M. Diaz, "On blockchain and its integration with IoT. Challenges and opportunities," *Future Generation Computer Systems,* 2018.

[31] A. T. Norman, *Blockchain Technology Explained: The Ultimate Beginners Guide About Blockchain Wallet, Mining, Bitcoin, Ethereum, Litecoin, Zcash, Monero, Ripple, Dash, IOTA and Smart Contracts*: CreateSpace Independent Publishing Platform 2017.

[32] A. Rosic. (2017, June 8). *Blockchain Tutorial | How To Become A Blockchain Developer*. Available: https://blockgeeks.com/guides/blockchain-developer/

[33] A. Panarello, N. Tapas, G. Merlino, F. Longo, and A. Puliafito, "Blockchain and IoT Integration: A Systematic Survey," *Sensors,* vol. 18, pp. 2575-, 2018.

[34] S. H. Islam and G. P. Biswas, "Provably Secure and Pairing-Based Strong Designated Verifier Signature Scheme with Message Recovery," *Arabian Journal for Science & Engineering,* vol. 40, pp. 1069-1080, 2015.

[35] X. Huang, X. Yang, A. Chonka, J. Zhou, and R. H. Deng, "A Generic Framework for Three-Factor Authentication: Preserving Security and Privacy in Distributed Systems," *IEEE Transactions on Parallel & Distributed Systems,* vol. 22, pp. 1390-1397, 2011.

[36] Q. Jiang, F. Wei, S. Fu, J. Ma, G. Li, and A. Alelaiwi, "Robust extended chaotic maps-based three-factor authentication scheme preserving biometric template privacy," *Nonlinear Dynamics,* vol. 83, pp. 2085-2101, 2016.

[37] The AVISPA Team, "HLPSL Tutorial - A Beginner's Guide to Modelling and Analysing Internet Security Protocol," http://www.avispa-project.org/2006.

[38] AVISPA. (July). *Automated Validation of Internet Security Protocols and Applications*. Available: http://www.avispa-project.org/

[39] T. Issariyakul and E. Hossain, *Introduction to network simulator NS2*: Springer US, 2009.
[40] H. J. Jo, J. H. Paik, and D. H. Lee, "Efficient Privacy-Preserving Authentication in Wireless Mobile Networks," *IEEE Transactions on Mobile Computing,* vol. 13, pp. 1469-1481, 2014.
[41] B. Lynn, *PBC Library Manual 0.5.14*. https://crypto.stanford.edu/pbc/manual.pdf, 2006.

Appendix A: Integration of fuzzy extractor bio-cryptosystem into DecChain

The principle of the fuzzy extractor bio-cryptosystem is for the user U_i to generate a pair of input (α, β) using the fuzzy extractor probabilistic generation function $Gen\ (.)$. β can be retrieved if the biometric key α is close to the biometric pattern earlier provided. Thus, the biometric key has to be above a certain threshold γ used in the fuzzy extractor deterministic reproduction function $Rep\ (.)$. For a user to log in and retrieve the details of a service provider, U_i inputs firsts inputs his biometric detail f_i^* into the device to retrieve the username and password. If $d(f_i, f_i^*) < \gamma$, then the device goes ahead to compute $\alpha_i = Rep\ (f_i^*, \beta_i)$to be subsequently inputted to decrypt d_i otherwise login request into the device is rejected. Upon decrypting d_i, the user retrieves ID_i and PW_i, inputs these details into the device to retrieve the details of service providers from the memory of their device.

Appendix B: Cryptanalysis of our scheme using AVISPA

The simulation was implemented in two phases. The first phase involved two basic roles namely $user\_1$ played by U and $service\_provider$ played by SP. The details with regards to the declarations of the basic user roles can be seen in figures 8 and 9. U initiates the transaction with T. SP receives T and authenticates user U using the command $witness(U, SP, sp\_u\_t, T')$ based on the content of T. SP then acknowledges the receipt of T by computing acknowledgment A to U. U receives T and authenticates SP through the command $request(U, SP, u\_sp\_ack, A)$ based on the contents of A.

In addition to declaring basic roles, HLPSL requires the declaration three other aspects of the simulation namely $session, goal$ and $environment$ as in figure 10. The individual roles declared are combined and instantiated to run in defined sessions in this section of the code. Again, the goal of the simulation is also declared under $goal$. The simulation had two secrets, $sec1$ and $sec2$ which are to be kept between U and SP without i gaining access to it. Again, both U and SP authenticate each other based on the cryptographic primitives within the respective files they receive. Any access to the secrets by i as well as the inability of the basic roles to successfully authenticate each other reveals the flaws in our scheme.

In addition to these two declared basic roles, three additional service providers were declared in phase two of the simulation making the basic declared roles five. Likewise, in phase two, two secrets were defined namely $sec1$ and $sec2$ as seen in figure 10. The secrets defined to be kept between U and SP_4 without either i, SP_1, SP_2 *or* SP_3 gaining access to the secured information. Again, the intended parties i.e. U and SP_4 authenticated each other based on the cryptographic primitives within the transaction and acknowledgment received at both ends.

```
role user_1 (U, SP: agent, Pu, Ps: public_key, Snd, Rcv: channel (dy))
played_by U def=

local State : nat,
Na, Nb, H: text,
T, A, Sigu, M, Si: message,
H1, H2: hash_func

init
State :=0
transition

1. State  = 0 ⋀ Rcv(start) =|> State':= 2
              ⋀ Na' := new()
              ⋀H' := H1(M)
              ⋀T' :=H1(Na'.Pu.{{Sigu.M}_Ps}.Ps.H')
              ⋀Snd(T')
              ⋀ secret({M},sec1,{U, SP})

2. State  = 2 ⋀ Rcv(A') =|> State':= 4
              ⋀ request(U,SP,u_sp_ack,A')

end role
```

Fig. B1. Role specification for U in HLPSL in phase 1

```
role service_provider(U, SP: agent, Pu, Ps: public_key, Snd, Rcv: channel (dy))
played_by SP def=

local State : nat,
Na, Nb, H: text,
T, A, Sigs, M, Ack: message,
H1, H2: hash_func

init
State := 1
transition

1. State  = 1 ⋀ Rcv(T') =|> State':= 3
              ⋀ witness(U,SP, sp_u_t, T')
              ⋀ Nb' := new()
              ⋀H' :=H2(Ack)
              ⋀A' :=H2(Nb'.Ps.{{Sigs.Ack}_Pu}.Pu.H')
              ⋀Snd(A')
              ⋀ secret(Ack,sec2,{U, SP})

end role
```

Fig. B2. Role specification for SP in HLPSL in phase 1

```
role session(U, SP: agent, Pu, Ps: public_key) def=
local SA, RA, SB, RB: channel (dy)
composition
user_1(U,SP,Pu,Ps,SA,RA)
/\service_provider (SP,U,Pu,Ps,SB,RB)

end role

role environment() def=
const u, sp: agent,
pu, ps, ki: public_key,
h1,h2: hash_func,
sec1, sec2,
u_sp_ack,
sp_u_t : protocol_id
intruder_knowledge = {u, sp,  pu, ps, ki, inv(ki)}
composition
session(u,sp,pu,ps)
/\session(u,i,pu,ki)
end role

goal
secrecy_of sec1
secrecy_of sec2
authentication_on u_sp_ack
authentication_on sp_u_t

end goal
```

Fig. B3. Role specification of *session*, *environment* and *goal* in HLPSL in phase 1

```
role user_1 (U, SP1, SP2, SP3, SP4: agent, Pu, P4: public_key, Snd, Rcv: channel (dy))
played_by U def=

local State : nat,
Na, Nb, H: text,
T, A, Sigu, M: message,
H1: hash_func

init
State :=0
transition

1.  State  = 0 /\ Rcv(start) =|> State':= 2
               /\ Na' := new()
               /\H' := H1(M)
               /\T' :=H1(Na'.Pu.{{Sigu.M}_P4}.P4.H)
               /\Snd(T')
               /\ secret({M},sec1,{U, SP2})

2.  State  = 2 /\ Rcv(A') =|> State':= 4
               /\ request(U,SP4,u_sp4_ack,A')
end role
```

Fig. B4. Role specification for U in HLPSL in phase 2

```
role service_provider1(U, SP1, SP2, SP3, SP4: agent, Pu,P4: public_key, Snd, Rcv: channel (dy))
played_by SP1 def=

local State : nat,
Na, Nb, H: text,
T, A, Sigs, M, Ack: message,
H1: hash_func

init
State := 1
transition

1.  State  = 1 /\ Rcv(T') =|> State':= 3

2.  State = 3 /\ Rcv(A') =|> State':= 5

end role
```

Fig. B5. Role specification for SP_1 in HLPSL in phase 2

```
role service_provider2(U, SP1, SP2, SP3, SP4: agent, Pu,P4: public_key, Snd, Rcv: channel (dy))
played_by SP2 def=

local State : nat,
Na, Nb, H: text,
T, A, Sigs, M, Ack: message,
H1: hash_func

init
State := 1
transition

1. State  = 1 /\ Rcv(T') =|> State':= 3

2. State = 3 /\ Rcv(A') =|> State':= 5

end role
```

Fig. B6. Role specification for SP_2 in HLPSL in phase 2

```
role service_provider3(U, SP1, SP2, SP3, SP4: agent, Pu,P4: public_key, Snd, Rcv: channel (dy))
played_by SP3 def=

local State : nat,
Na, Nb, H: text,
T, A, Sigs, M, Ack: message,
H1: hash_func

init
State := 1
transition

1. State  = 1 /\ Rcv(T') =|> State':= 3

2. State = 3  /\ Rcv(A') =|> State':= 5

end role
```

Fig. B7. Role specification for SP_3 in HLPSL in phase 2

```
role service_provider4(U, SP1, SP2, SP3, SP4: agent, Pu, P4: public_key, Snd, Rcv: channel (dy))
played_by SP4 def=

local State : nat,
Na, Nb, H: text,
T, A, Sigs, M, Ack: message,
H1: hash_func

init
State := 1
transition

1. State  = 4 /\ Rcv(T') =|> State':= 5
              /\ request(U,SP4, sp4_u_t, M)
              /\Nb' := new()
              /\H' :=H1(Ack)
              /\A' :=H1(Nb'.Pu.{{Sigs.Ack}_Pu}.Pu.H')
              /\Snd(A')
              /\ secret({Ack},sec2,{U, SP4})

end role
```

Fig. B8. Role specification for SP_4 in HLPSL in phase 2

```
role session(U, SP1, SP2,SP3,SP4: agent, Pu, P4: public_key)
 def=
local SA, RA, SB, RB, SC, RC, SD, RD, SE, RE: channel (dy)
composition
user_1(U,SP1,SP2,SP3,SP4,Pu,P4,SA,RA)
/\service_provider1 (SP1,U,SP2,SP3,SP4,Pu,P4,SB,RB)
/\service_provider2 (SP2,SP1,U,SP3,SP4,P4,Pu,SC,RC)
/\service_provider3 (SP3,SP1,SP2,U,SP4,P4,Pu,SD,RD)
/\service_provider4 (SP4,SP1,SP2,SP3,U,P4,Pu,SE,RE)
end role

role environment() def=
const u, sp1, sp2,sp3,sp4: agent,
pu, p4, ki: public_key,
h1: hash_func,
sec1, sec2,
u_sp4_ack,
sp4_u_t : protocol_id
intruder_knowledge = {h1,u, sp1, sp2, sp3, sp4, pu, p4, ki, inv(ki)}
composition
session(u,sp1,sp2,sp3,sp4,pu,p4)
/\session(i,sp4,sp1,sp2,sp3,p4,ki)
end role
```

Fig. B9. Role specification for *session* and *environment* in HLPSL in phase 2

```
goal
secrecy_of sec1
secrecy_of sec2
authentication_on u_sp4_ack
authentication_on sp4_u_t

end goal

environment()
```

Fig. B10. Role specification for *goal* in HLPSL in phase 2

.

Ernest Bonnah

Prof. Ju Shinguang

Highlights

1. Inclusion of any trusted third party into any authentication scheme introduces issues relative to scalablity as well as a single point of failure.
2. Blockchain can be incorporated in an edge computing environment to eliminate these challenges.
3. Integrating blockchain principles into edge computing also presents challenges which ought to be resolved.
4. Simulation results show a successful integration of the blockchain principles into edge computing can achieve a comprehensive authentication scheme.

Ernest Bonnah had both his BSc. and MSc degrees in from Kwame Nkrumah University of Science and Technology in Ghana and is currently pursuing a Ph.D in Jiangsu University in the School of Computer Science and Telecommunication Engineering. His research interests include Security in Edge Computing, Wireless Sensor Networks and Computer Networks.

Prof. **Shiguang Ju** received his M.S. degree in 1988, from Nanjing University of Science and Technology (China) and a Ph.D degree in 1996 from CINVESTAV of National Polytechnic Institute (Mexico). He is professor of School of Computer Science and Communication Engineering of Jiangsu University, China. His current research interests include wireless networks, information security and big data. He is the Corresponding Author of this paper. Email: jushig@ujs.edu.cn

Author Statement

We appreciate the detailed criticisms and suggestions made by the reviewers. We believe these suggestion and corrections are geared improving the quality of our work. We have therefore acknowledged every concern raised and have made the needed suggested changes.

Once again, thank you for the detailed work done in reviewing our work.

Conflict of Interest

The authors declare there is a no conflict of interest.