Blockchain and Edge Computing based Architecture for Participatory Smart City Applications

Zaheer Khan
[*]*University of the West of England, Coldharbour Lane, BS16 1QY Bristol, United Kingdom*
Zaheer2.Khan@uwe.ac.uk

Abdul Ghafoor Abbasi
[+]*RISE Acreo, AB Isafjordsgatan 22, 164 40, Kista, Sweden*
[^]*SEECS, National University of Sciences and Technology, Islamabad, Pakistan*
abdul.ghafoor@ri.se

Zeeshan Pervez
[±]*University of the West of Scotland, High Street, PA1 2BE, Paisley United Kingdom*
zeeshan.pervez@uws.ac.uk

* Contact author

Abstract

Smart cities aim to provide smart governance with the emphasis on gaining high transparency and trust in public services and enabling citizen participation in decision making processes. This means on the one hand data generated from urban transactions need to be open and trustworthy. On the other hand, security and privacy of public data needs to be handled at different administrative and geographical levels. In this paper, we investigate the pivotal role of blockchain in providing privacy, self-verification, authentication, and authorisation of participatory transactions in open governance. We also investigate that to what extent edge computing can contribute towards management of permissioned sharing at specific administrative levels and enhances privacy and provides an economic approach for resource utilisation in a distributed environment. We introduce a novel architecture that is based on distributed hybrid ledger and edge computing model. The architecture provides refined and secure management of data generated and processed in different geographical and administrative units of a city. We implemented a proof of concept of the architecture and applied it on a carefully designed use case, citizen participation in administrative decisions through consensus. This use case highlights the need to keep and process citizen participation data at local level by deploying district chaincodes and only share consensus results through permissioned chaincodes. The results reveal that proposed architecture is scalable and provide secure and privacy protected environment for citizen participatory applications. Our performance test results are promising and show that under control conditions, the average registration time for a citizen transaction is about 42ms, while the validation and result compilation of 100 concurrent citizens' transactions took about 2.4s.

Keywords: *Open governance; Edge Computing; Blockchain; Distributed Ledger; Smart Cities; Citizen Participation; Security; Privacy; Trust;*

1. Introduction

Smart cities are becoming prevalent in urban areas to deal with different societal challenges such as sustainable transport, energy, greenhouse gas emissions and monitoring, public health & quality of life, economic & job growth, etc., [1], [2]. The ubiquitous nature of smart IT-based solutions provides new services to citizens. As a result, it has become a source of new information for city administrations for smart open governance, based on which city development plans are proposed and strategic decisions are made. Such a smart open governance approach aims to engage with citizens through innovative approaches and deliver highly transparent and trustable public services [3]. The involvement of citizen in city planning through consultation meetings, opinions, polls, and call-for-comments on online proposals is referred as citizen engagement or participation e.g. smarticipate project [27]. This enables city administrators to get to know actual needs of the local community, co-create and transform the city infrastructure and services around citizens' needs and requirements.

Berntzen [28] highlighted importance of citizen's role in the participatory process. Citizens' competence, local knowledge, and awareness of issues can produce better plans and services, and their capabilities as data providers/contributors (e.g. crowdsourcing, citizen science) can facilitate building liveable environments. In the new changing landscape of integrated and participatory urban governance, there is need to provide more sustainable, open and transparent IT solutions which can promote and achieve greater degree of citizen power in participatory decision making (e.g. Arnstein's participation ladder [29]).

Existing citizen participation solutions are centralised and focus on specific thematic applications. However, the collection and storage of data across functional urban areas (i.e., districts) can have high carbon footprint due to transfer and processing of the citizen participation data at a data centre or cloud. In addition, the openness of public services and associated data can be vulnerable to security and privacy threats. Therefore, there is a dire need to redesign and configure IT infrastructure in smart cities, which can handle data processing, transfer, and storage efficiently i.e., securing citizen participation information at district/unit levels and passing on the processed information to city administrators, which informs the city planning. Smart city IT infrastructure should secure and ensure privacy of the data and services managed in and across various geographical administrative boundaries. For example, delegation of power in cities from central administration to town councils or districts is a well-known phenomenon. This means in a District some of the services are managed by the town councils or local district administration, in other cases major or critical services are provided and decisions are made by the main City Administration. Therefore, the local data is processed locally and ensures the privacy of the users while the refined and resultant data is managed by the City administration. In our solution, we considered this limitation and divided the access rights of the citizens and administration geographically.

In the context of smart cities there is a lot of on-going research on internet-of-things (IoT) and edge-computing [1] and [7], to distribute computing model at edge of the IT infrastructure in order to gain functional and geographical scalability [7], [8], [9], [10], [11],

[12], [13], and [14]. However, most existing solutions focus on IoT and/or sensors data processing through edge computing enabled deployment models. These solutions focus upon designing efficient IoT stream processing and actuating platforms – thus most of the effort goes in the reliability of sensing platform, processing of sensory data, and responding on processed sensor data through actuators and provisioned services. Our aim is to adopt edge computing as an economic and environmentally sustainable approach for managing resources in a distributed data processing environment.

Citizen participation opens significant research challenges like trust, transparency, auditability, traceability, security, and privacy of citizen data and its management, are mostly ignored by most of the existing work e.g. [30] and [31]. Those who do only provide mundane solutions by either tapping into existing social interaction platform tailored for limited citizen involvement e.g., voting and comments, or building citizen involvement services which cater for special need e.g., initiate new proposals and exchange ideas [27].

In this respect, blockchain technology [4] has emerged as a great enabler, providing distributed trust and openness in transaction based systems. Beyond the most widely used and known use-case of blockchain in the form of cryptocurrencies, blockchain technology can be developed to provide verification of data (transparency), proof of data origin (traceability), restrain data modification (auditability), security services (authentication and authorization) and ensures the privacy of personal data. Though there is a lot of debate on blockchain standards [5], basics of blockchain remain same across a variety of existing blockchain solutions [4].

Despite blockchain's favourable characteristics for distributed applications, it has not been fully investigated in smart cities participatory applications development and deployment context. European Parliamentary Research Service's Scientific Foresight Unit advocates blockchain will have disruptive effect and impact on lives e.g. public services such as record keeping without the need of a third party making the processes efficient and effective [34]. Recently PricewaterhouseCoopers (PwC) published a report on potential use and benefits of blockchain in making cities smarter [33]. For instance, Dubai is aiming to take their governmental transactions on blockchain by 2020, Estonia has developed a blockchain solution to a host of government services on its own blockchain solution called keyless signature interface (KSI) [36], and Delaware is implementing a real-time stock ownership tracking system on a blockchain. In [39], Blockchain is proposed for vehicular network and intelligent transportation in smart cities. With greater opportunities, there are research challenges such as protecting identities or keeping them anonymous, right to forget [34], law compliance through smart contracts for automated verification [35].

Considering the role and impact citizen participation has in transforming conventional cities to smart cities [32] and realising smart open governance, in this paper we propose a novel blockchain and edge computing based architecture to engage citizens in urban planning. Our architecture utilises blockchain technology to realise an open and auditable governance framework for smart urban planning. Edge computing is utilised to realise a green smart IT infrastructure by segregating city and district levels planning and decision making, whilst creating tightly coupled evidence-based communication mechanism. In order to make our work clear, the conceptual model of the proposed architecture is presented in Figure. 1. In

Figure 1, we introduce District Admin as compute node that provides secure and trusted data processing capacity and data management capability for citizens.

Recently, edge computing is proposed as economic approach for resource management in mobile blockchain applications [40]. Our objective is to innovate citizen participation with blockchain and edge computing for transparent, auditable, traceable, secure and privacy-aware participatory applications in smart cities. The proposed architecture has made following contribution:

- we propose a novel architecture that is inspired by edge based model to handle efficient citizen participation data collection, processing, storage and permissioned sharing of data across geographical administrative boundaries;
- the architecture uses blockchain based solution for transaction traceability and auditing but also to provide data security and privacy protection through verifiable identities, encryption, authentication, and authorisation;
- we use a smart citizen participation use case to develop a proof of concept demonstrating how blockchain and edge computing can work together for developing and deploying participatory applications in a smart city environment.

It is worth mentioning that our approach uses blockchain and edge computing differently than conventional approaches [4] and [7], i.e. we eliminate the need for mobile edge nodes and hence avoid mining overhead by deploying multiple edge node (i.e. district nodes) in a city to provide a trusted platform. Furthermore, our approach uses a different technique to perform a trusted transaction. This approach does not rely on the mining process to append a new block (transactions) to a chain since our transactions are inherently trusted and only authorized citizen can perform a transaction. In our approach addition of a new block is event driven i.e., when citizen participation time is elapsed for a specific call for comments/participation/proposal, all individual transactions are evaluated and verified by the corresponding node before appending to the chain.

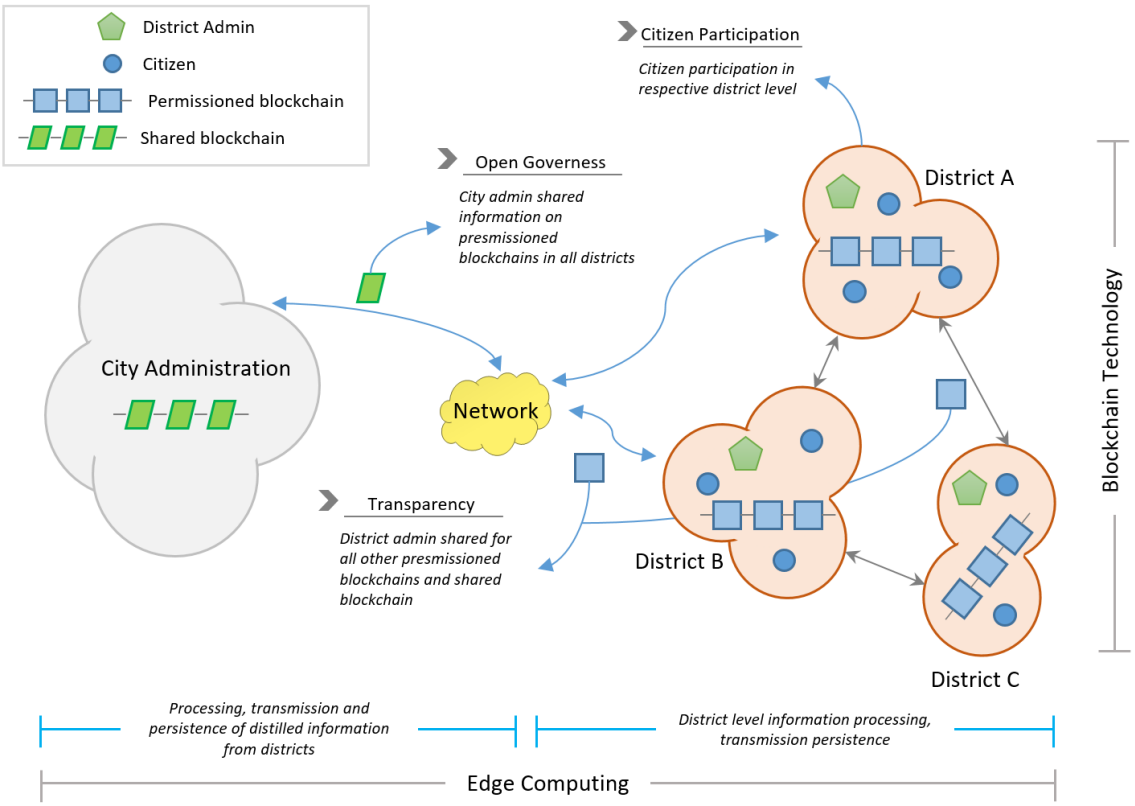

Figure 1: *Edge Computing and Blockchain for Smart Cities.*

The rest of this paper is organised as: Section 2 discusses the related work. Section 3 presents the proposed architecture of edge computing and blockchain enabled participatory smart city applications. System implementation details are discussed in Section 4. Section 5 presents the evaluation of the proposed architecture on a realistic scenario of citizen engagement in the smart city. The paper is concluded in Section 6 along with future directions.

2. Related work

This section covers two complementary aspects of our proposed system, edge computing, and blockchain technology. Edge computing enables computation and network intensive application to process and manage data at the edge of the network. Blockchain provides open and transparent data management platforms on which applications requiring auditability and transparency can be built. In the following, we present state-of-the-art in both of these areas with particular focus on edge based smart cities and data management through blockchain technology.

2.1. Edge and smart cities:

In the last couple of years, edge computing [6], [7] has been increasingly gaining attention from smart city applications [7], [8], [9], [10]. This can be attributed to edge computing characteristics of functional and geographical scalability in urban environments and efficient utilisation of compute, network and storage resources. However, a lot of research is taking place to deal with computation capacity, security, and placement of edge devices for specific smart city applications such as vehicular network [11].

In [12], authors present a service-oriented middleware approach called SmartCityWare, to integrate cloud-of-things and fog computing for different smart city applications. The SmartCityWare provides a virtual environment to develop and operate smart city applications by creating a set of services and using a multi-agent runtime environment. Authors claim that service-oriented approach can provide flexibility and extensibility; however, increasing the number of sensors or actuators (cloud-of-things) and mesh of agents for managing resources, job scheduling and monitoring across fog will require highly reliable data management and secure mechanism.

In [13], authors introduce iSapeins, and IoT-based platform which utilises software agents and virtual objects (VO), deployed on distributed compute nodes (Raspberry Pi 2 model B board), for implementing application and services for the smart city. All software agents and VO are managed by an iSapeins server in an external data centre. These agents running on compute nodes provide processing capacity and capability at the edge nodes and aggregated data is stored on a centralised database, resulting in efficient processing, storage, and network bandwidth usage. The smart street case in the city of Cosenza (Italy) demonstrates decentralised urban intelligence services to urban stakeholders.

In [14], authors argue that due to vendor lock-in in an edge infrastructure there is less flexibility of deploying third-party services. They introduce the concept of participatory edge computing system running on home gateways. Such a system can serve as an open edge environment to deploy services in city neighbourhoods. Dedicated contributed nodes use cloudy software distribution and personalised services deployed using Docker containers.

Similarly, there are other examples such as managing smart city applications in 5G edge network [15], managing IoT at the edge [16], etc. However, most of the existing literature focuses on fog and edge computing for sensors or IoT based data collection and processing.

2.2. Blockchain for distributed data management:

Aniello et. al., present implementation details of layered blockchain system that solve the problem of data integrity for distributed databases redo log [17]. In conventional database management system redo logs are used for change management and auditability; however, unauthorised modification to redo logs can lead to inconsistent data in a distributed database. The concept of layered blockchain was originally proposed in [18] called 2LBC, which utilised two blockchains to ensure data integrity of redo logs. The first layer of blockchain is a permissioned blockchain, and the second layer is public permission less blockchain. The first layer utilises a fast consensus algorithm (leader rotation), for each rotation, a leader is selected using a fair selection policy. In a distributed setting, each federated domain contributes a miner (leader) to the algorithm, collectively all miners maintain a consistent replica of the ledger and the database (i.e., redo logs). The second layer utilises proof-of-work consensus algorithm to ensure the integrity of the first layer blockchain. Periodically hash from the first layer is sent to the second layer via Anchoring Manager, this periodic push of first layer blocks to the second layer realises immutability.

BBDS is a blockchain based medical record sharing system [19]. The system utilises permissioned blockchain, which only allows invited users to access the shared medical records. BBDS is a use-case of blockchain technology for confidential data sharing. BBDS allows users to access the data on the blockchain once their identities and cryptographic

keys are verified. Medical records are accessed through a shared pool on a blockchain network. MeDShare [20] is another medical big data sharing system, which employs blockchain and smart contracts to monitor the data access and modification. Every data interaction is recorded in blockchain for traceability and auditability. The implementation of MeDShare can provide data provenance and auditing to cloud service providers and data guardians while sharing medical data with research and medical institutions with minimal risk to data privacy. MedRec [21] presents another use-case of blockchain for healthcare sector as a whole. The system provides a comprehensive, immutable and easy access to medical records across providers and treatment sites. The system utilises proof-of-work based blockchain technology to manage authentication, confidentiality, and accountability for medical sharing.

Data provenance in cloud based data management system poses significant challenges towards data integrity and trust on public cloud service providers. ProvChain [22] is a blockchain enabled data provenance framework, which provides tamper-proof records for transparent data accountability in public clouds. ProvChain considers individual files as data units, and records user operations on individual data units. The provenance data is embedded in blockchain transactions, which can be retrieved for data provenance checks.

PriWatt [23] is a token based blockchain enabled energy trading platform for a smart grid. Besides blockchain, it utilises multi-signatures and anonymous encrypted message streams to enable peers in anonymously negotiating energy prices and securely performing transactions (energy trading). PriWatt utilises a peer-to-peer system to replicate energy trading data among peer; in addition to this it makes use of the proof-of-work to overcome Byzantine failures and to restrain from double-spending attacks – both are critical for any electronic payment system like energy trading as demonstrated by the PriWatt.

In [24], authors propose a theoretical security framework that aims to integrate blockchain with smart city devices to provide a secure communication platform. However, this work is at very preliminary concept stage; no practical aspects related to design and implementation are covered in [24]. Our previous work [25] covered detailed security and privacy issues for smart city data and applications and provided SSServProv framework that provides end-to-end security and privacy to city participatory smart city applications. However, due to use of conventional distributed approaches, SSServProv framework lacks economically and environmentally sustainable data management and processing. In [26] we introduced the concept of Verifiable Identity Block (VeidBlock) that utilises blockchain concept to generate verifiable identifiers (ids).

Our proof of concept covered authentication and self-verifiable identities using distributed ledgers. These identities are cryptographically protected and can only be registered in the distributed ledger if the trusted node vetted its information and issues digital certificate. In this paper, we extended the VeidBlock concept to provide authorisation and privacy protection using a customized and hybrid blockchain model that can be deployed in an edge computing environment. We use the concept of chaincode which is governed by the smart contract that defines specific constraints on transaction, access and data exchange. For the sake of brevity, we used smart contract and chaincode interchangeably because chaincodes are same as a smart contract since it is responsible to manage transactions stored in the form of chain in physical storage and applies rules which are programmed in it to perform application specific transactions. To the best of our knowledge, this work is novel and has not been reported in the literature.

In the following Table 1, we present a detailed comparative analysis of the existing work, which either directly addresses the societal challenges (e.g. transport, energy, sustainability, economic aspects, etc) in smart cities by means of edge computing, blockchain and Internet-of-things, or provides an overarching framework to address these issues. Citizen participation and open governance play a pivotal role in realising smart cities which are transparent in their day-to-day activities to respond to the societal challenges and citizens engagement for effective city enhancement projects. To ensure the underlying infrastructure and services which provide citizen engagement and open governance are secure, scalable and support automation of workflow, we evaluated the existing work for security, trust and usage of DLT and edge computing. From the analysis it is evident that existing work significantly lack the support for citizen participation and open governance. Though a reasonable progress has been made in the adoption of DLT and edge computing; however, realisation of open governance and citizen participation remains the open research challenge. The proposed architecture of blockchain and edge computing (Section 3) and its realization as a proof of concept (Section 4) present the novel contribution of this work for participatory smart city applications.

Table 1: Comparative analysis (**Key:** ++ : means discussed in detail with a proposed solution; + : means very briefly talked about it e.g. survey of others work; -- : not covered; - : mentioned but not discussed in detail)

Papers	**Citizen Participation**	**Open Governance**	**Framework Administrative Delegation**	**Scalability of Infrastructure**	**Security / Privacy**	**Trust, Confidence, Data Reliability**	**DLT (smart contracts, Legal aspects, etc)**	**Edge / Fog**
[47]	--	--	--	+	+	+	+	+
[48]	--	--	-	-	--	--	++	++
[49]	--	--	-	+	+	+	++	++
[50]	+	--	-	+	++	++	++	++
[51]	--	--	-	+	++	++	++	++
[52]	--	--	--	--	--	--	++	++
[53]	--	--	--	--	++	+	+	--
[54]	--	--	+	--	-	--	--	++
[55]	--	--	--	+	+	-	-	+
[56]	--	--	--	+	+	+	+	+
[57]	--	-	-	+	+	+	++	++
[58]	--	--	--	--	+	+	++	++
[59]	--	--	--	+	++	++	++	++

3. Blockchain and Edge based Architecture

We first briefly elaborate a use case that highlights the problem of keeping and processing citizen participation data locally i.e. at district level and only consensus results are shared widely. Then we explain architectural details of our proposed solution. Then we derive testing scenarios to evaluate system performance and security provision.

3.1. Use case:

Suppose city Pesh is a large city with a population of 500,000. City Pesh is divided into 10 geographical wards (or districts) where each ward councillors look after local neighbourhood level issues, plan local actions and invest public funds in much needed initiatives. The annual budget for the city Pesh is set by the central city administration. As part of the smart governance initiative, City Pesh would like to provide transparent services to their citizens. Among those services include participatory budget planning, transparent and up-to-date status of public funds spending. City administration, ward councillors and citizens can identify priorities for different proposals, view funds spending on different activities by different city wards.

Each ward in City Pesh is interested to collect and process data generated by local residents and hence would like to employ suitable provenance and privacy protection mechanisms. City administrators (e.g. Mayor) is more interested in various integrated reports at city scale such as total number of unique participants in the city or comparative analysis between different wards, total number of local ward-level initiatives, number of citizens benefited from public funding, etc. Whilst it is necessary to be able to identify individuals for auditing purposes, the security and privacy of citizens personal information is also important due to data protection regulations.

3.2. Architecture Details:

The proposed solution comprises of various components shown in Figure 2. We divided the architecture into two major units: i) *City Admin*, and ii) *District Admin*. *City Admin* is a compute node that manages core activities which are applied across the whole city e.g. managing identities of citizens. In contrast, *District Admin* is a compute node that manages core activities within specific districts or wards e.g. citizens interactions on local proposals. There are one *City Admin* and many *District Admins* each representing to city district or ward. Some components are installed in *City Admin* and some are deployed in *District Admin*. In each city administration setup, all citizens are registered with *City Admin* through *Citizen Registration Manager Service*, which uses *Identity Provider* (IP) component. This is a traditional registration authority which plays a pivotal role in our framework. It is designed to use for (i) users or citizens registration, (ii) managing identities information, (iii) validating identities to be acceptable in the domain to interact with the ledger, and (iv) providing local authentication services to form a verifiable identity, VeidBlock [26].

In this solution, City Admin manages a *Certification Manager* (CM) component. This component is tightly coupled with IP and therefore each citizen (or any user in City Admin as well as District Admin) registered in the IP possesses security credentials such as private key and X.509 certificate which are issued by the CM. All these credentials are being managed by the *Citizen Facing Apps* on behalf of citizens and can be exported to other devices. It is worth mentioning here that the CM also issues X.509 certificates to all the

applications and services provided through *Decentralised App* and *Citizen Facing Apps* components. The purpose of issuing these credentials is to perform required cryptographic operations on personal information, business data and blocks (i.e. chaincodes) before publishing in the *Distributed Ledger* (or Blockchain) or sharing with external entities. This feature provides trust on the participating citizens and Apps because only registered citizens are eligible to acquire certificate from CM.

In order to perform a transaction in the above setup, it is important to ensure the authenticity of the source and destination entities. To achieve this feature, in our system each citizen creates a VeidBlock [26] by using the IP component and then interacts with distributed Ledger to publish VeidBlock in the relevant chaincode(s). These VeidBlocks are cryptographically encapsulated using public-key cryptography and hence can only be opened by the authorized user. Furthermore, VeidBlock does not encapsulate citizen's personal data but it can be used to validate its existence and authenticity by verifying its contents. As shown in Figure 2, the District Admin also deployed an Identity Authentication Service which uses verification of VeidBlock to authenticate the users for benefiting services deployed in the District Admin.

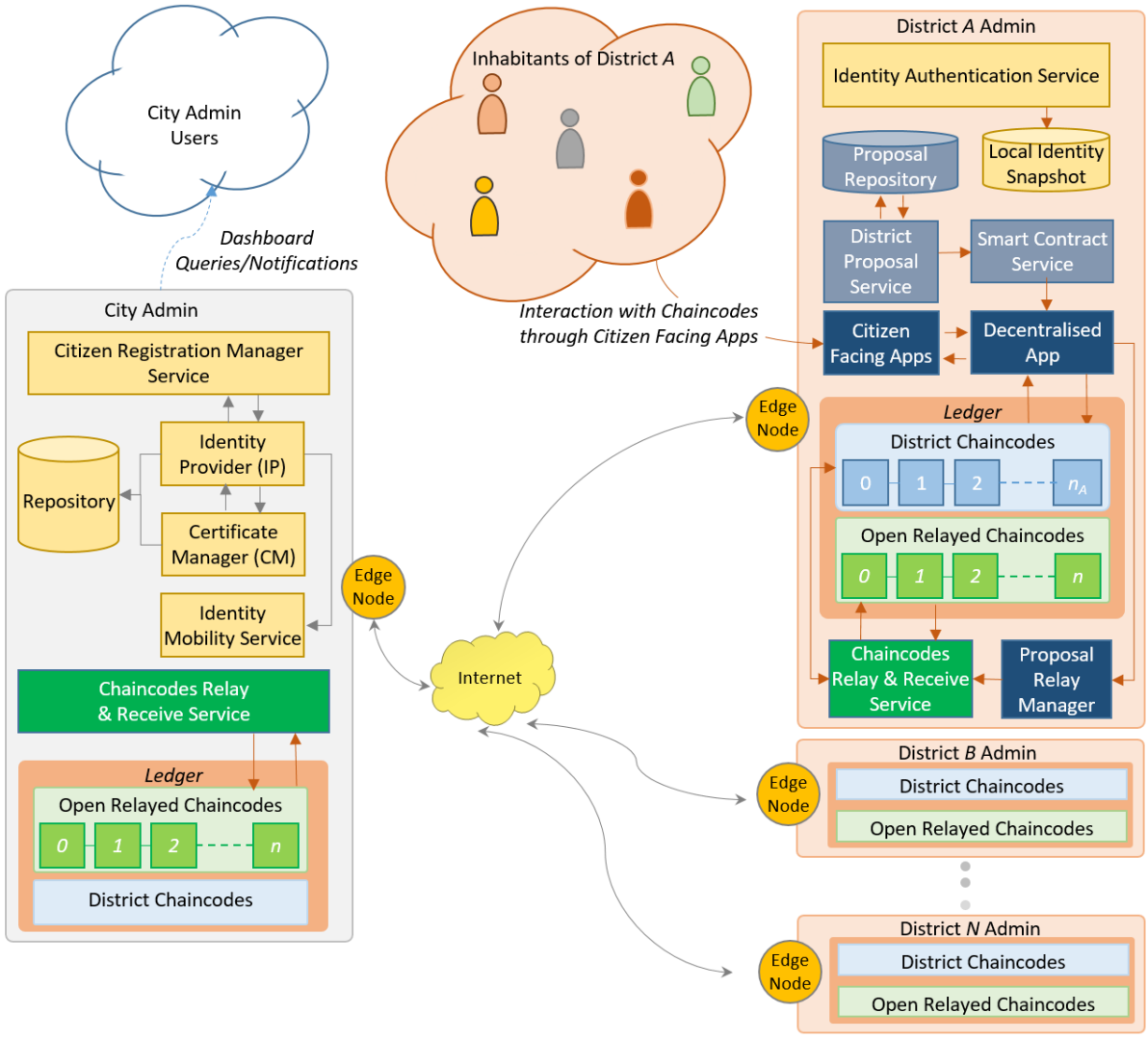

Figure 2: *System Architecture: citizens' interaction through district nodes and permissioned chaincodes.*

The *Distributed Ledger* component implements blockchain services to store and share identity and other smart city related data in a consistent and secure manner. Most of currently available distributed ledgers are by nature connected in a peer-to-peer fashion to maintain a consistent state of every block. In our proposed solution, they are also connected in peer-to-peer fashion; however, considering the sensitivity of the stored blocks, we extended the concepts of standard blockchain and introduced relay permission options through smart contracts. Therefore, distributed ledgers at individual *Districts Admin* level are not in a consistent state, across various District Admin nodes - the scope of a distributed ledger at *District Admin* level is local as it is not shared across other District Admins nodes. Only those blocks will be consistent which are open, authorized or permissioned by the owner to relay these blocks on other blockchain network (i.e., districts and city) otherwise it will stay in the local domain. Since all blocks are chained with each other therefore to make it verifiable and validated, the header of each block is consistent on all the distributed ledgers. This will be discussed in detail in section 4.2 - proposal chaincode creation.

The customization in its relaying behaviour is based on the requirements of the citizen's data, administrative boundaries and privacy regulation. In our proposed distributed ledger two types of the chaincodes are developed as depicted in Figure 3: (i) *District Chaincode* (DC) and (ii) *Open Relayed chaincode* (ORC). The local blocks are maintained by the district level ledger in DCs and all the recorded transactions will remain in the district. These chaincodes will not be synchronized with the peer ledgers through ORC. The residents of a specific district can only interact with DCs of their own district. Each district level proposal or activity is recorded in the DCs and the residents interact with the proposal or activity through *Citizen Facing Apps* to provide opinions, initiate new proposals, comments or participating in a poll, etc. Any proposal, voted by the residents is recorded in DCs.

In conventional blockchain systems participating nodes compete to append a new block to the blockchain and get rewarded for their participation i.e., solving a cryptographic puzzle. The proposed architecture is based on citizen participation, new blocks are added to the ORC when a *District Admin* concludes the engagement activity i.e., reporting results of a citizen participation on a request for comments, call for participation, or opinion poll etc. Since, in each district a single copy of ORC is maintained, newly added blocks are synced seamlessly on their respective chains by using peer-to-peer feature of the blockchain.

In the proposed architecture each transaction is digitally signed by the citizen. *District Admin* verifies the signatures and ensure only validated transactions are added to *District Chaincodes*.

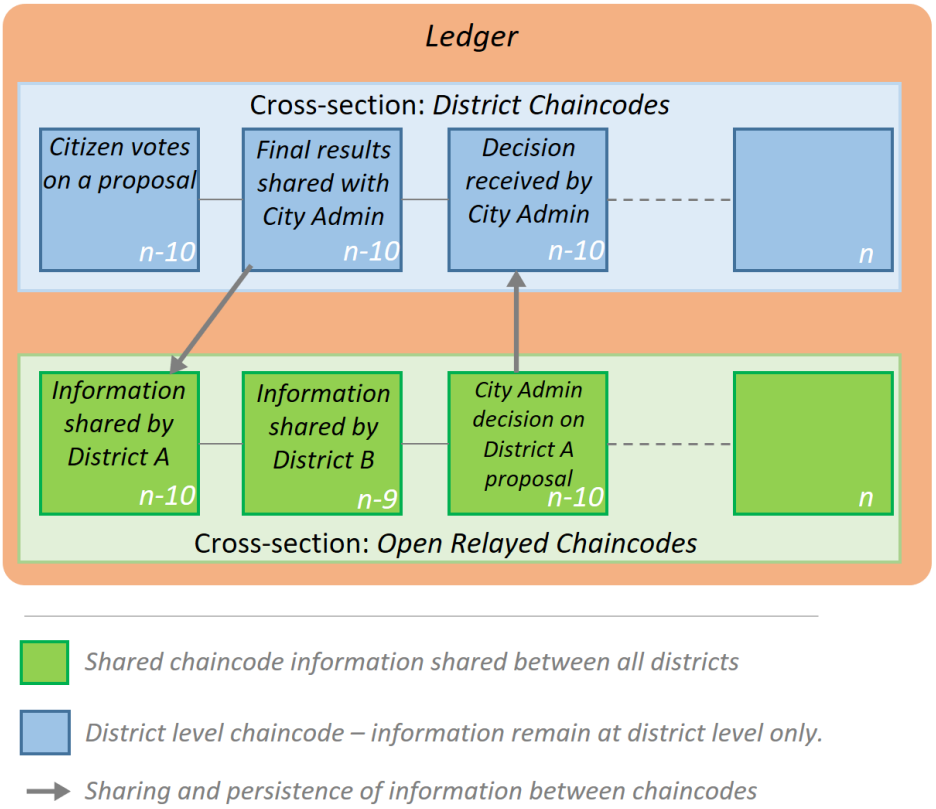

Figure 3: *Ledger: secure data sharing through District and Open Relayed Chaincode*

The second type of chaincode is distributed and open in nature and known as ORC. It is initiated by the *City Admin* and relayed to the peer ledgers across District Admin nodes. The openness, access rules, cryptographic functions and behaviour of such chaincodes are governed by specific constraints on transaction, access and data exchange. To make it easier to understand we can refer to these constraints as smart contract. It is important to mention here that in our system, these constraints define basic attributes which can be extended with more fine-grained elements and rules for automated distributed transactions and data exchange between chaincodes. The metadata and description in JSON format of our constraints is presented below:

Smart Contract or Constraints Template:

- **SCOPE** *{OPEN or LOCAL}* : Scope defines the access level of the transactions stored in the chain.
  - *LOCAL: If scope is LOCAL then only local users and administrators can append and access transaction in this chain*
  - *OPEN: If scope is OPEN then any authenticated users and administrators can append and access transaction in this chain.*
- **SECURITY_LEVEL** *{NONE, DIGITAL_SIGNATURE, ENVELOPED, DIGITAL_SIGNATURE_ ENVELOPED}* : Security level defines the end-to-end payload protection level.
  - *NONE: Clear payload,*
  - *DIGITAL_SIGNATURE: In this option, payload is digitally signed and encapsulated in PKCS7 cryptographic format by the clientAPI,*

- *ENVELOPED: In this option, the payload is enveloped in PKCS7 enveloped cryptographic format by the clientAPI,*
- *DIGITAL_SIGNATURE_ENVELOPED: In this option the payload is digitally signed and enveloped in PKCS7 Signed and Enveloped cryptographic format by the clientAPI.*

- **Start DateTime**: Chaincode available for appending and viewing transactions
- **End DateTime**: End date time defines the closing time of the chaincode, after this time no one can append transaction in the chaincode but can view it.

Information related to districts remains in DCs and can only be recorded in the ORC through *Chaincodes Relay & Receive Service* (CRRS). CRRS follows the rules and procedures defined in the associated smart contract. The main purpose of ORC is to realise open and transparent governance between a city and its districts i.e., *City Admin* and *Districts Admin* nodes. In our scenario, only users in City and District admins can interact with the ORC through CRRS e.g. any proposal or activity approved at the district level is recorded in the ORC by using the CRRS. *City Admin* can fetch the specific block (proposal or activities) from the ORC and make appropriate decision e.g. new policy or funding approval. The action taken by local authority or city administration will also be recorded in ORC e.g. funds released for a specific district is also recorded in ORC to maintain financial transparency. CRRS of a district receiving the funds also copy the blocks (blocks which contain the release of funds) in its own DCs. Information in ORC has a limited access since these are cryptographically enveloped for authorized users such as City and District Admins, unless requested by legal entities to comply with rule of law.

The novel combination of *ORC*, *District Chaincodes* and *City* and *District Admin* nodes have realised an edge-computing model. For the sake of brevity, the only one edge node is considered in each district; however, the proposed architecture can support multiple edge nodes at a district level. With multiple edge nodes, subsystems (identify authentication service, constraints or smart contract service, proposal repository etc.) of the architecture can be replicated and/or delegated to support horizontal and vertical scaling.

4. System Implementation - Proof of concept

In the following text, we will use District Admin interchangeability for both district administrator and software interface (app, web etc.) used by the district administrator. This distinction between them will be context based. Our proposed solution follows the process depicted in Figure 4. All users and system components are registered with *Identity Provider* so that necessary certificates can be issued and kept in a shared certificate chaincode. The certificate chaincode is included in ORC so that all District Admins should be able to access and verify users' credentials. *City Admin* creates *Request4Proposal* that is used to collect compiled results from *District Admin* when citizen participation on a *PlanningProposal* is completed (*PlanningProposal* and *Request4Proposal* are discussed below).

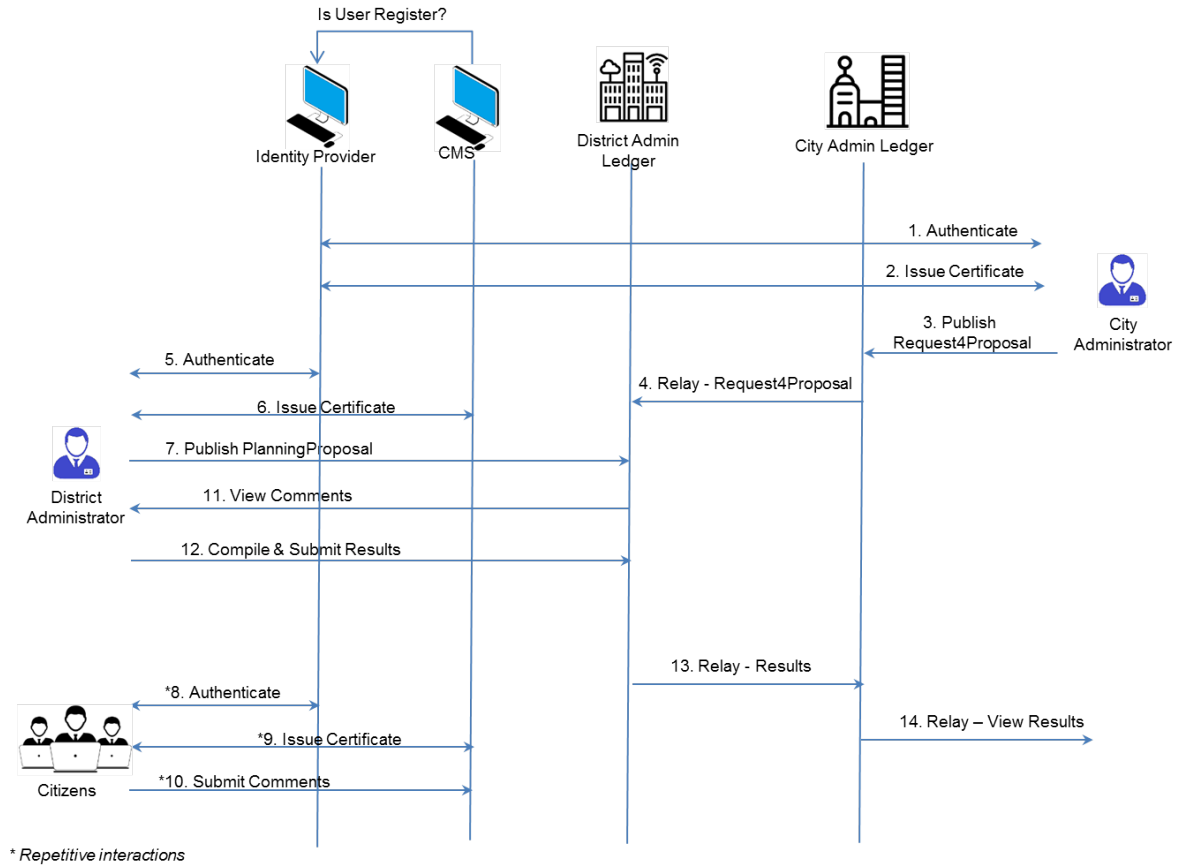

Figure 4: *Working Implementation: sequence of steps for blockchain based user (city and district administrators, and citizen) authentication, proposal and transaction submissions, and processing of results through Edge computing model.*

All computing modules, involved in the solution, are implemented by following the concepts of RESTful micro-services [41] and used java based dropwizard framework [42]. RESTful API exposes various endpoints for serving client requests like registration, certificate issuance, registration of a chaincode and handling transactions. Various endpoints are also implemented for processing all the transactions and generating results for relaying to City Admin. Since current modules are implemented as a proof-of-concept, therefore all services are currently deployed on a same network and are accessible to the clients using standard HTTPS protocol. The efficient communication between the micro-services, developed to manage the chaincodes, is achieved by integrating Apache Kafka framework (a messaging service) [43] while security features are implemented by using bouncy-castle library [44]. In Kafka, we define various topics when we create a new chaincode and all citizens subscribe with that topic are authorized to send and receive information about transactions. All transactions are stored in local storage on edge nodes in the form of chaincodes as depicted in Figure 2. Section 4.2 describes the chaincode creation workflow. In this way, we implemented the concept of Open Relayed Chaincode and District Chaincode to restrict their accessibility.

To be more specific, our implementation architecture of the project is based on the highly modular and low coupling software artefacts. All these are implemented individually as a separate maven project which provides the realization of a small concept. For example the concept of Veidblock is implemented in org.acreo.veidblock where it has verifiable identity

(JWTToken). The authentication and verification protocols are implemented in org.acreo.auth and the project org.acreo.ipv provides identity management services. All registered users can access ledger services through REST endpoints implemented in the org.acreo.ledger project. In addition, the transactional data is stored in the databases created using mysql server while the Apache kafka is used as a messaging backbone for peer-to-peer synchronized communication. These both features are implemented in database and messaging projects. The implemented system has many supporting projects. For example ip is used to store and manage identity data, init is used for initialization of RESTfull services, activation project facilitates citizens to activate their account, and AsymmetricKey is used to manage the key pair of citizen on their local devices. All these projects are shown in the following Figure 5:

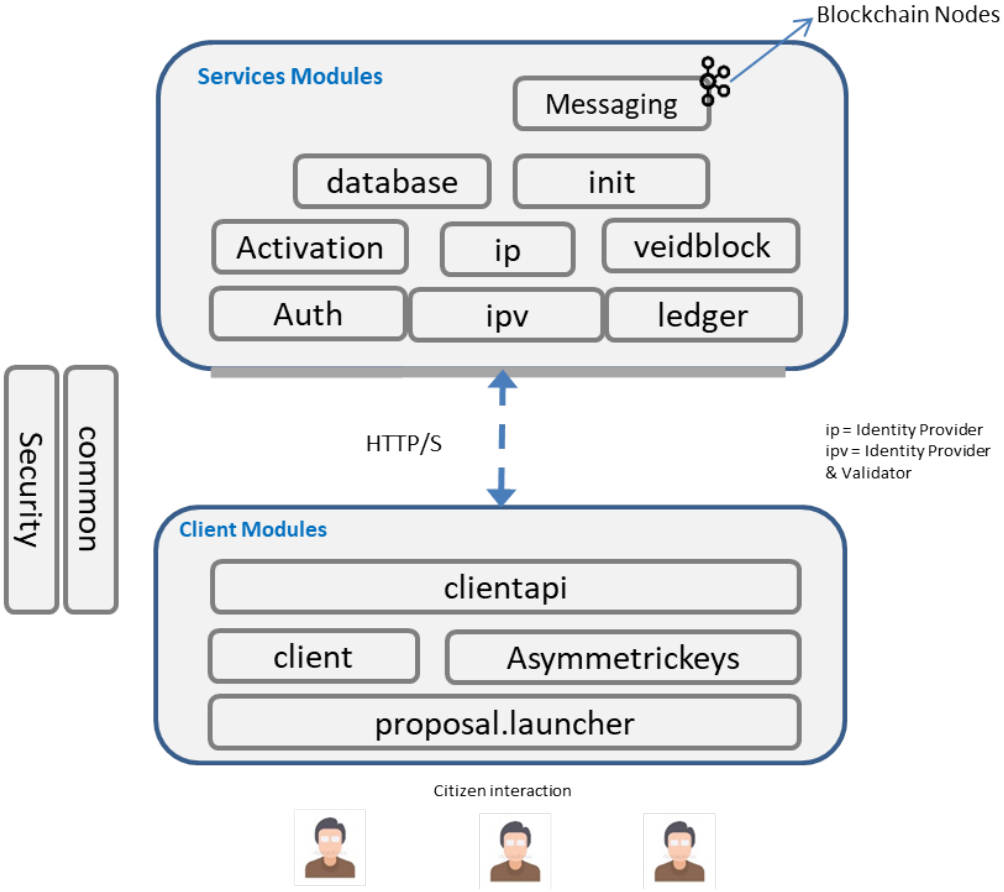

Figure 5: Implementation – proof of concept

In the above system, the following Table 2 shows that ledger services provide following main endpoint to facilitate the client apps while the main path of the resource is "/vc".

Table 2: service endpoints

Endpoint (where '/vc' is as prefix)	Description
@POST path("/chain")	Accepts an object of BlockHeader to create a new chaincode and returns its reference (ref).
@POST path("/trans/ref/{ref}")	Accepts an object of TransactionBlock to add in the *ref* chaincode and returns transaction reference (also known ref and should not be confused with chaincode ref. It comes with trans as a prefix).
@GET path("/chain")	Returns refs of all registered chaincodes.
@GET path("/chain/ref/{ref}")	Returns all transactions registered in a ref chaincode.

@GET path("/chain/owner/{creator}")	Returns all ref of chaincodes registered by a user.
@GET path("/chain/chainName/{chainName}")	Returns all transactions registered in a specified 'chainName' chaincode.
@GET path("/trans/ref/{ref}")	Returns a transaction registered against a ref.
@GET path("/trans/sender/{sender}")	Returns list of transactions sent by a user as a sender.
@GET path("/trans/receiver/{receiver}")	Returns list of transactions where a receiver is mentioned as recipient.
@GET path("/trans/ref/{ref}/sender/{sender}")	Returns a transaction sent by a user as a sender heaving ref as a transaction reference.
@GET path("/trans/ref/{ref}/receiver/{receiver}")	Returns a transaction received by a user as a receiver heaving ref as a transaction reference.
@GET path("{ref}")	Returns a complete chaincode which has specified ref.

The client modules are implemented in org.acreo.cleint and org.acreo.clientapi projects which use security and common projects for basic functionality. These two projects provide citizen registration, key pair generation, interaction with ledger services, and identity verification from auth service. Therefore they are considered as engine modules of the clientapp (for example proposal launcher).

The clientapp interacts with services modules through RESTfull API (HTTP based communication). For example, in our implementation the following code at the client side is used to generate and manage key pairs in certificates.

```
public class CertificateConnector {

public boolean createCertificate(String uid, String password, String verifyerURL) throws VeidblockException    {

                        CertificateSuite certificateSuite = new CertificateSuite(uid + "", 3);

                        RestClient restClient = RestClient.builder().baseUrl(verifyerURL+"/cert/request").build();

                        ClientCertificateHandler clientCertificateHandler = new ClientCertificateHandler(certificateSuite);

                        return clientCertificateHandler.issueCertificate(restClient, uid, password);

            }

}
```

The second most important functionality of the clientapp is to add a transaction into the ledger. In our implementation following code snippet shows how to create a transaction for chaincode and then send it to the ledger for creating a new chaincode (in our use case is a proposal).

```
public TransactionHeaderCO addTransationHeader(BlockHeaderCO chainCode, String verifier) throws VeidblockException {
    Representation<?> response = null;
     try {
            response = restClient.post("/vc/chain", chainCode, AuthenticationHeader.authHeader(verifier, authenticator));
            if(response.getCode() != 200 ){
                        throw new VeidblockException("Error Code: "+response.getCode()+", Message :  "+response.getBody().

toString());
            }
```

```
TransactionHeaderCO transactionHeaderCO = new ObjectMapper().readValue(response.getBody().toString(),

TransactionHeaderCO.class);
            return transactionHeaderCO;
    } catch (Exception e1) {
              throw new VeidblockException(e1);
            }
}
```

Once the chaincode is created then it can be viewed by using the following function:

```
public TransactionHeaders getTransactionHeaders() throws VeidblockException {
     Representation<?> response = null;
      try {
            response = restClient.get("/vc/chain", null);
            if(response.getCode() != 200 ){
                  throw new VeidblockException("Error Code: "+response.getCode()+", Message : "+response.getBody()
                                                                                                    1.toString());
            }
          TransactionHeaders transactionHeaders = new ObjectMapper().readValue(response.getBody().toString(),
                                                                                    TransactionHeaders.class);
            return transactionHeaders;
       } catch (Exception e1) {
            throw new VeidblockException(e1);
      }
}
```

The comments against a proposal are submitted by the citizens using following piece of code which is extracted from org.acreo.proposal.launch.districtadminA; package:

```
public void submitResponse(ResourceCO resourceCO, CONSENSUD_RESPONSE res, String comments)
                                                                        throws VeidblockException {
            Ledger ledger = Ledger.builder().resource(resourceCO).build(authenticator);
            TransactionHeaderCO transactionHeaderCO = ledger.getTransactionHeaderByName(proposalName);
            if (Objects.isNull(transactionHeaderCO)) {
                        logger.error("--- E --- Could not find chainblock with name '" + proposalName + "'");
            }
            ConsensusResponse consensusResponse = new ConsensusResponse();
            consensusResponse.setResponse(res);
            consensusResponse.setComments(comments);
            ledger.addTransaction(transactionHeaderCO, null, consensusResponse, new Configuration().getAuthServerUrl());
}
```

All components are deployed in different virtual machines (section 5) and each virtual machine is designated as node. We implemented the proof of concept from scratch and without using existing blockchain solutions as the required hybrid features were not available in one solution. It is designed to support participatory applications. In our citizen participatory use-case we needed a hybrid approach because proposals are only shared between citizens living in a specific district or area while city admins provide open access to funding data. At the time of implementing proof of concept, existing blockchain solutions were either open or close and were not supporting the required hybrid features including verifiable IDs. Also, our solution eliminates the need of consensus and mining processes and hence contributes

towards sustainable participatory sensing in smart cities. Complete code for authentication, Veidblock, ledger, and clientapi module is available from the github: https://github.com/abdulghafoorabbasi/veidblock.git.

As mentioned in Section 3.2, all required components *IP*, *Certificate Manager* are deployed in City Admin whereas *Distributed Ledger is* deployed at all nodes (Figure 2). For the proof-of-concept, we implemented clients for *Citizen Facing Apps*. We also implemented *Client API* which supports operations of all entities (citizens, district and city admins) registered in the *Identity Provider* component and have necessary security credentials (such as basic authentication attributes) to verify its identity. In simple words, *Client API* provides a controller interface to platform components. Different architectural components work in request and response mode. This means a component has a client part and service part. This means District admin, City admin, DAL, citizens etc are assigned service requesting interfaces. Service requesting interfaces define behaviour of a specific component and enables the component to interact with the system through *Client API*, e.g. DAL can verify integrity of a new block through *Client API*. This allows different smart city applications (or Citizen Facing Apps) to interact with the platform. This process can be managed based on our previously published work in [25][26].

In order to demonstrate the concepts of open governance, transparency, and citizens engagement with security and privacy services, we created two main working scenarios to demonstrate our use case (section 3.1): (i) The District Admin creates chaincode called *PlanningProposal* for engaging citizens in order to get their views about a proposed urban regeneration planning initiative in the district, and (ii) The City Admin creates *Request4Proposal* for District Admins to provide summary of results gathered through *PlanningProposal* chaincode. The process to create both chaincodes is same but different smart contracts will be used. For instance, the scope of *PlanningProposal* is local to a district; i.e. registered residents of the same district can participate. In contrast, the scope of *Request4Proposal* is open so each District Admin can submit shareable/permissioned results in this chaincode.

Similarly, the security level for *PlanningProposal* is digital-signature [37], because we are interested to ensure the integrity and source authentication of the vote. The security level for *Request4Proposal* is signed-and-enveloped [37] so we want to ensure its confidentiality, source authentication, and data integrity. Since the chaincode creation process is the same, the following subsections describe the *PlanningProposal* chaincode creation and citizens' participation scenario.

In the following, we present the implementation details of our proposed blockchain and edge computing based citizen participation architecture. We first discuss the details of Credentials Management (4.1), which ensures only authorised entities (city and district admins, and citizens) can participate in a city planning project. We then discuss the working details of Chaincode Creation (4.2) through which district admin initiates blockchain based citizen participation (opinions, polls etc.) on a certain city planning project. After that, Chaincode and Citizen Engagement (4.3) is discussed which records citizen participation as blockchain transactions; Credentials Management ensures only authorised citizens are able to contribute to a citizen participation request. In the last, we discuss Compilation and Submission to the City Admin (4.4), which enables district admin to compile the citizen

participation (i.e., making decision evidenced through blockchain transactions). Based on edge computing model, the district admin then only sends the compiled results to City Admin. This avoids unnecessary network bandwidth which would have been required to transmit each blockchain transaction i.e., individual citizen participation data.

4.1. Security *Credentials Management:*

In order to develop a trusted network and creating security credentials, City Admin deploys Certificate Management Service (CMS) to issue X509Certificate to all the resources including components, services, citizens, and users from city administration. In the same setup, we designed and deployed distributed ledgers which are connected with each other using a messaging-service in a peer-to-peer network to synchronize ORCs. Each distributed ledger creates its local security credentials (private key, public key) and it's X509Certificate which is certified by the CMS. In order to make these certificates available across the domains, we used the blockchain based mechanism to publish certificate by creating a certificate chaincode, as default chaincode to be available in every distributed ledger. When a citizen starts interaction with the Distributed Ledger, she must authenticate her credentials with the district level IP; once authenticated she obtains her X509Certificate using handling-certificate-protocol described in [38]. Each certificate holder also publishes her certificate in the certificate-chaincode (explained in the following section). In the citizen's certificate, the distinguished-name attribute (i.e. subject name in the certificate) is a random number (pseudorandom) that does not reveal the personal information of the user, therefore, the certificate is anonymous but traceable because the certificate issuer has its information stored in the local IP for reverse mapping. All the other stakeholders and resources follow the same process to create and possess security credentials.

4.2. Proposal *Chaincode Creation:*

According to the proposed working scenario, the District Admin creates a chaincode which will be used by the District Admin to create and publish the proposal in district level instance of the distributed ledger. To make it clear we shall refer it as DAL (District Admin Ledger) in the following text. This published chaincode will be used by the citizens to vote or comment on the proposal. For chaincode creation, the District Admin creates a list of constraints (or *smart contract* (sc)) and specifies the access level policy, constraints or restrictions of the chaincode such as its start and closing date & time and the security requirements. The District Admin digitally signs this information by using its own private key as shown in eq2 and then sends it to the DAL along with its VeidBlock. The DAL verifies VeidBlock to authenticate the District Admin and then verifies *PlanningProposal* signature (chp`) for source authentication and data integrity. In this verification process, the DAL relies on the credentials already published in the certificate-chain. After successful verification, the DAL appends local time, the hash of previous *PlanningProposal* header, reference number and its position in the header chaincode as shown in eq3. Then, it signs new *PlanningProposal* header and publishes it in the ledger which is also relayed to the other peer instances of the distributed ledger. The linkage between header and transactions are shown in Figure 6. Figure 6 is the implemented version of distributed ledger in Figure 3 and provides a snapshot of how chaincode headers are linked with its associated transaction blocks and next headers.

Figure 6 shows that the hash of each transaction block in DC is linked with the next transaction block through 'Previous Transaction Hash' field. These hash links are inspired by

our previous work on the VeidBlock which provides a tamper resistant solution at block header level [26]. Extending VeidBlock, tamper resistance at transaction level is achieved by implementing the concepts of hash chaining at the transaction level. In this case, the hash of the previous transaction is included in the next transaction and the hash of all the transactions is stored in the 'merkle hash' field of the chaincode header. The complete message of the *PlanningProposal* header in JSON format is shown in Appendix A. Using the edge computing model, the *PlanningProposal* is available to residents of a specific district (i.e. edge node - Figure 2), therefore its scope is local so only local residents are able to view the published proposal and vote, comment etc.

$chp = n \mid sc$ ---- eq1

$chp` = sign(chp, DA_c)$ ---- eq2

$ch` = sign(R|d|\ hash(ch^{-1})|chp|t, DA_l)$ ---- eq3

Where chp = chaincode payload
N = chaincode name
sc = smart contract
DA_c = Private key of District Admin
ch^{-1} = Header of previous chain code
DA_l = PrivateKey of Ledger
t = current time
R = Proposal Reference Number
d = depth of a transaction

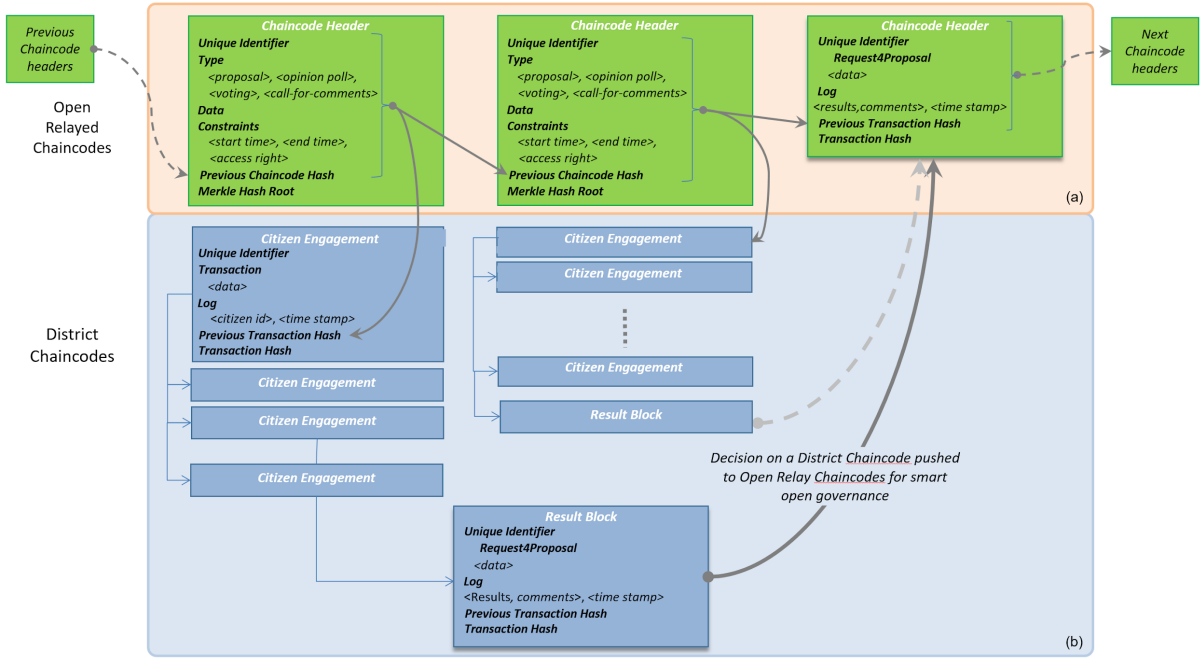

Figure 6: *Structure of chaincode and linkage between headers and transactions in each chaincode. a) links between headers protect against tampering and deletion of chaincode; b) the links between transactions 'b' protect against tampering and deletion of a transaction from chaincode.*

4.3. Publish *Proposal Chaincode and Citizen Engagement:*

The District Admin defines a new proposal (*PlanningProposal*) using its service requesting app. It specifies the proposal name, its objective, location, description and voting/comment attributes. Once the app compiles this information it then passes the information to the Client API with the header's reference number as payload. The reference number is used to uniquely identify the chain in the blockchain network. The Client API processes this

information according to the rules defined in the associated smart contract. The proposal (payload) is encapsulated using PKCS7; the Client API digitally signs it according to the process described for creating *PlanningProposal*. Then it sends this information to the DAL where hash of previous *PlanningProposal* is created, depth is included in the referred chaincode, and date and time are added. For previous hash, if the depth of the transaction is 1 then it includes the hash of chaincode header as a previous hash of current entry as shown in Figure 6. After creating the entry, the DAL service requesting app checks the different parameters of the *PlanningProposal* and associated smart contract. In this scenario, our chaincode scope is local so the DAL will not relay this chaincode on the message bus, which is used to synchronize open relay chaincodes. In addition to this, the DAL will check that the start and end date of the proposal is valid.

For citizen engagement, the service requesting app uses *ConsensusResponse* interface. This interface is implemented for citizens service requesting app to enable citizens to select options like Agreed, Disagreed and Don't Know; and also provide comments as free text on a specific *PlanningProposal*. This information will be used as a payload to be published in the *PlanningProposal* as a transaction.

4.4. Result *Compilation and Submission to the City Admin:*

The District Admin automatically downloads a complete chaincode (*PlanningProposal*) and performs the following operations to verify the correctness of the chaincode:

a. Verifies that the complete chain is not tempered: This is performed by verifying the hash of the previous transactions.
b. Verification of each transaction: This is verified by verifying the hash of transaction and signature of each transaction which was signed by the DAL.
c. Payload Verification: The service requesting app also ensures the integrity of the payload by verifying and extracting PKCS7 signed data.
d. Time Validation: The District Admin can also check that the vote is submitted in time by checking the creation time of the transaction.

Once the verification process is complete, the District Admin opens the encapsulated payload and then processes citizens responses to generate the result (or aggregated quantitative and qualitative report). After that, the District Admin creates a transaction to submit compiled results (i.e. summary report) to the City Admin through *Request4Proposal*. For provenance and on-demand data sharing we established another efficient protocol to share the original transactions data with the City Admin. Since anonymised comments gathered from citizens can require more storage space and bandwidth for sharing through distributed ledger; therefore, we created a document which contains all the comments and then encrypted it using random key (symmetric key). The District Admin uploads the protected document on the central repository and receives a downloadable URL. The District Admin passes random key, the hash of the document, URL, and results as a payload to the ledger Client API. The Client API encapsulates the payload in PKCS7 enveloped-data, based on the instruction given in the *Request4Proposal* constraints or smart contract and follows the same process as described in eq2 and eq3. The City Admin service requesting app can download all compiled results through the *Request4Proposal* chain and process it for further analysis and decision making.

5. Evaluation and Discussion

In this section, we test and evaluate our system implementation under control conditions to derive different quantitative and qualitative results. In our deployment scenario, we have created one city level node and 2 district level nodes (also referred as edge nodes). Then for testing, we created city admin, district admin, distributed ledger (DAL and City Admin Ledger) and test data for 100 citizens in each district. For all entities, authentication credentials including certificates were created and published in the certificate chaincode. City Admin launched *Request4Proposal.* To demonstrate citizen engagement, a sample set of 100 citizens is used in each district. In each District we increment number of users sequentially and measure the overall transaction time. At peak, 100 users simultaneously provide comments on the proposal in their respective district. While running all these transactions, we measured the time taken for verification, certificate creation and registration, and the time taken for preserving and processing comments. For our deployment scenario following experimental setup is used:

The experiment setup comprises of 3 OS Ubuntu 16.0.4 LTS Virtual Machines. Each VM had 4GB RAM; CPU 2GHz; and 32GB HDD storage. Oracle JDK1.8, MySQL, Maven, and Apache Kafka were installed on each VM. The proposed architecture is implemented in Java and it uses Apache Kafka as a messaging service to synchronize chaincodes between the ledgers installed on City Admin and Districts Admins nodes. MySQL is used as a database to store citizen's registration data, chaincodes and VeidBlocks.

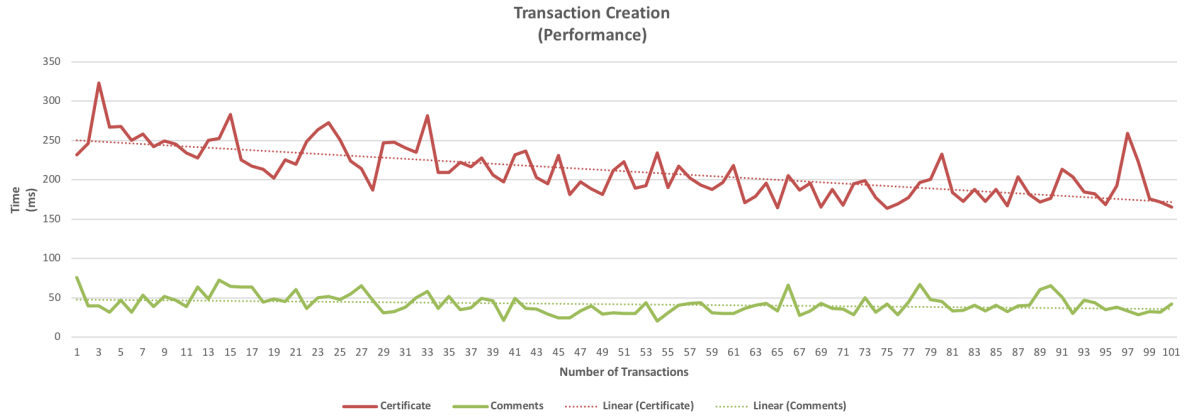

Figure 7: *Average time for certification creation and publication and comments transactions in PlanningProposal.*

After deploying our system with all necessary services, we executed our use case. First, the City Admin created *Request4Proposal* and then District Admin created *PlanningProposal*. Then each citizen authenticated with IP, created a certificate and then published her certificate in the Certificate chaincode. After that, each citizen provided her consents on published planning proposal through comments as mentioned in above paragraphs. We measured the performance by calculating its transaction creation time. For each transaction (i.e. x-axis in Figure 7 and Figure 8) same experiments were repeated at least 5 times. In this arrangement, we performed first test for 100 citizens and saved transactions processing time. In addition, we eliminated results storage and its processing time to calculate the actual time required for transaction processing. In order to bring more reliability and fairness in the results and eliminating the CPU utilization by other processes, the same procedure was repeated five times and average of transaction processing time is plotted in the graphs. As

shown in Figure 7, the certificate transaction takes more time than participating on the *PlanningProposal* because the size of certificate transaction (approximately 1947 bytes) is larger than the planning proposal (approximately 531 bytes) which takes more time for hash generation, certificate chain verification and processing before storing or publishing the certificate. In addition, we observed overall performance trends for multiple users and noticed that in both cases the creation time is linear which indicates that the proposed architecture is scalable - capable of processing and verification of large chaincodes (higher number of concurrent transactions) without compromising performance. The average time for certificate transaction creation is 211ms while for *PlanningProposal* takes 42ms to register a new transaction in the chaincode.

Verification of transactions in the distributed ledger is a key feature so in our testing setup we measured its verification time as shown in Figure 8. Based on the results, we observed that the first transaction verification time is slightly high because in this phase the software loads required libraries for verification. In overall we observed that the trend is linear as we increase the number of transactions, it increases verification time as well because in each case previous transactions are processed and verified in order to verify the complete chain of selected transaction.

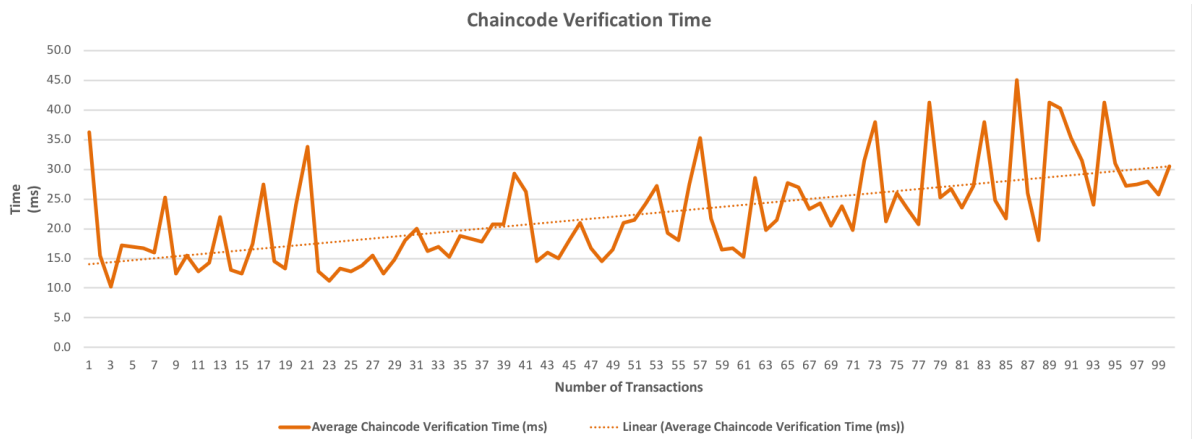

Figure 8: *Average time to verify PlanningProposal chaincode.*

Based on the above results the following observations are made:

- **Architecture scalability and flexibility:** All cities are different but they face similar challenges. For example, the number of districts, wards or neighbourhoods can be different from one city to another city (e.g. smaller, medium and large scale cities) and number of residents in those geographical boundaries can vary. Our proposed architecture makes use of Edge computing nodes to handle the complexity of a city governance as the city grows in terms of geographical boundaries, citizen engagement, cross-departmental collaboration and the number of residents. The Edge computing model allows to store and process data close to where it is generated (i.e. district level) and only aggregated results are shared across the city, making efficient use of IT infrastructure resources e.g. bandwidth utilisation, computation power etc. In addition, our system testing results yield negligible effect on the performance of the system when the number of users increase.

- **Economically and environmentally sustainable approach:** Since the intensity of data generation and processing for a participatory application is often low for an individual citizen, our system does not employ edge nodes on citizens' devices. In addition, our event driven approach eliminates the need of compute intensive mining process to append new block (transactions) to a chain. This combination makes our architecture economically and environmentally sustainable.

- **Permissioned sharing:** In our system, we make novel use of hybrid distributed ledger, and managed the district data in their respective District Chaincodes, and shared compiled citizen engagement data in Open Relayed Chaincodes. This results in keeping data from a specific district to be stored and processed at district node only. District admin can share the comments or data on demand to City admin for evidence based decision making, provenance, and traceability. The data published in the distributed ledger is cryptographically signed by the data producer (i.e. citizens). Therefore, it gives credit to its owner and ensures ownership of the data, along with traceability.

- **Immutability, openness, transparency and trust:** All citizen engagement in different planning proposals, are persisted in the hybrid distributed ledger which is immutable therefore cannot be tampered and ensures the integrity of the data. If any part of the transaction is changed or tempered, it can be easily detected by verifying the chaincode. Due to our hybrid distributed ledger design, transactions in district level chaincodes are open to read and provide transparency in public affairs and raises the level of trust in public administrations.

- **Privacy and confidentiality:** Data Protection and digital privacy related regulations (European Union's General Data Protection Regulations (GDPR) and alike) mandate not to publish personal information (e.g. name, address, bank account, health information etc.) in public domain through web apps or remotely accessed software systems. Our system allows to share information through blockchain with selected recipients by cryptographically encapsulating it to protect the data from unauthorised access. This can be useful for GDPR's controlled sharing requirement by the data owner. In most of the existing blockchain implementations, anonymous identity is used, which cannot be tracked back to its original owner. In our system, these identities are linked with identity provider which acts as a citizen registration authority (e.g. city administration), so that these can be used to produce personal information for court and legal needs only. We used VeidBlock [26] approach which provides a mechanism to self-verify identities of users involved in transactions and it contains only anonymous identity information which does not reveal citizen's private information.

- **Compliance with standards:** In our system, we used standard cyber security techniques like symmetric, asymmetric keys encryption, X509 certificates, authentication, and authorisation, etc. This enables our system to be adopted by existing IT infrastructures which are looking to manage their data transactions through blockchain. In our proposed framework we have used standard cryptographic primitives regularly used in IoT and blockchain [45] and has been rigorously evaluated [46]. This helped to move security and privacy services at the

framework level, instead of at application level. However, due to the recency of blockchain in smart cities domain, there are yet no blockchain standards exists [5]. Our design and implementation of hybrid distributed ledger can be a stepping stone in this direction.

- **Smart contracts:** Our system implements minimal smart contracts to perform automated transactions between various entities i.e. Client API, District Admin, City Admin, DAL. For example, using smart contract's scope field (i.e. LOCAL and OPEN), it is automatically decided whether a transaction is accessible to all or to specific users.

It was observed that our system fulfils many GDPR requirements; however, there is more research needed to make a system that conforms fully to GDPR e.g. right to forget. We did not use existing blockchain solutions like Ethereum, Hyper ledger, Corda, etc., because we needed a hybrid approach that could allow necessary security and privacy provision, enable permissioned sharing and also can work along with Edge computing model. Therefore, our system is mainly designed to manage smart city applications where citizen participation is a key requirement. However, in our implementation, we used minimal rules to define constraints (or smart contracts) and require more investigation to design an event and logic based smart contract to address the requirements of the broader set of future transactions. Furthermore, a robust consensus algorithm among ledger nodes is also required when there are multiple District nodes within a district. These challenges will be addressed as part of our future research direction.

6. Conclusions

In this paper, we introduced a novel architecture that uses hybrid distributed ledger and edge computing to manage citizen driven city enhancement projects through secure, privacy-aware, and transparent citizen engagement. Our proposed architecture made use of edge computing to process and manage citizen engagement data in their respective district level chaincodes, whilst making administrative decisions on compiled citizen engagement data without consuming unnecessary mining process, bandwidth, and computation power, resulting in an architecture that is economically and environmentally sustainable. It utilised hybrid blockchain based data management and persistence to realise open and transparent governance. The innovative use of two different types of blockchains (district chaincodes and open relayed chaincodes) leveraged the edge computing model, and ensured anonymity, security, and immutability of citizen engagement data.

The practicality and efficacy of the proposed architecture were demonstrated through a realistic citizen engagement scenario for open governance. The rigorous evaluation of the architecture showed promising results for processing, compiling and validating citizen engagement on a commodity hardware and network settings. The average transaction creation time in the PlanningProposal chaincode is about 42ms. The system was able to verify the 100 citizen responses and associated time for compilation of the results in about 2.4s.The architecture was built considering security and privacy of the citizen engagement data, it anonymous identities, and blockchain transaction encryption through symmetric and X509 certificates to ensure confidentiality of the data persisted in the blockchain both at the district and city levels. From the evaluation, it is was demonstrated the proposed architecture underpinned by edge computing and blockchain is scalable to be deployed in smart citizen

environment for secure and immutable handling of citizen engagements and city administration through city enhancements projects.

Our work provides numerous future research opportunities. In our future work we focus on blockchain enabled automation of city administration through smart contracts. In the current implementation smart contracts manage the chaincode through rules specified in a contract; we will extend the capabilities of smart contracts to handle financial transactions and process automation. For mega cities, scaling district nodes i.e. more than one district nodes in one district, will also require mining and new consensus algorithms. We will also evaluate applications of consensus algorithms to timely synchronise open relay headers across all district nodes, and for specifically managing smart city's day-to-day activities and processes. We also aim to investigate how to handle mobile users who relocate from one district to another. This may lead our work towards mobile blockchain where a district node will be like an edge server and citizens' devices (e.g. smartphone, tablet, PC) will become edge nodes. The compliance of the architecture with GDPR will be meticulously evaluated.

Acknowledgement:
This research activity is partially sponsored by the RISE Competence Platform for Cybersecurity, Stockholm, Sweden. The icons used in Figure 1., are taken from TheNounProject under Creative Common License CCBY, created by AlfredoCreates.com/Icons, US; Creative Stall, PK; parkjisun; Maria Kislitsina, RU; and Stock Image Folio, RO.

References:

1. Khan, Z., Anjum, A., Soomro, K. and Muhammad, T. (2015) Towards cloud based big data analytics for smart future cities. *Journal of Cloud Computing: Advances, Systems and Applications*, 4 (2). ISSN 2192-113X
2. Khan, Z., Liaquat Kiani, S. and Soomro, K. (2014) A framework for cloud-based context-aware information services for citizens in smart cities. Journal of Cloud Computing: Advances, Systems and Applications, 3. ISSN 2192-113X
3. Javed, B., Khan, Z. and McClatchey, R. (2018) An adaptable system to support provenance management for the public policy-making process in Smart Cities. Informatics, 5 (3). pp. 1-26. ISSN 2227-9709
4. Tien Tuan Anh Dinh, Rui Liu, Meihui Zhang, Gang Chen, Beng Chin Ooi, Ji Wang, Untangling Blockchain: A Data Processing View of Blockchain Systems, August 2017
5. Ashiq Anjum, Manu Sporny, Alan Sill, Blockchain standards for compliance and trust, IEEE Cloud Computing, pp. 84-90, July/August 2017.
6. Yuan Ai, Mugen Peng, Kecheng Zhang, Edge cloud computing technologies for internet of things: A primer, Digital Communications and Networks, 2017, DOI: https://doi.org/10.1016/j.dcan.2017.07.001
7. Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, Lanyu Xu, Edge Computing: Vision and Challenges, IEEE Internet of Things, 3(5), 637:646, October 2016.

8. Charith Perera, Yonguri Qin, Julio C. Estrella, Stephan Reiff-Marganiec, Athanasios V. Vasilakos, Fog Computing for Sustainable Smart Cities: A Survey, ACM Computing Surveys, 50(3), Article 32, June 2017.
9. Tarik Taleb, Sunny Dutta, Adlen Ksentini, Muddesar Iqbal, Hannu Flinck, Mobile Edge Computing Potential in Making Cities Smarter, IEEE Communications Magazine, March 2017. DOI: 10.1109/MCOM.2017.1600249CM
10. Teruo Higashino, Hirozumi Yamaguchi, Akihito Hiromori, Akira Uchiyama, Keiichi Yasumoto, Edge Computing and IoT based research for building safe smart cities resistant to disasters, 37th IEEE International Conference on Distributed Computing Systems, pp. 1729-1737.
11. Gopika Premsankar, Bissan Ghaddar, Mario Di Francesco, Rudi Verago, Efficient Placement of Edge Computing Devices for Vehicular Applications in Smart Cities, IEEE/IFIP Network Operations and Management Symposium, April 2018. [To appear]
12. Nader Mohamed, Jameela Al-Jaroodi, Imad Jawhar, Sana Lazarova-Molnar, Sara Mahmoud, SmartCityWare: A Service-Oriented Middleware for Cloud and Fog Enabled Smart City Services, IEEE Access, 5(2017), 17576-17588, DOI: 10.1109/ACCESS.2017.2731382.
13. Franco Cicirelli, Antonio Guerrieri, Giandomenico Spezzano, Andrea Vinci, An edge-based platform for dynamic Smart City applications, Future Generation Computer Systems, 76(2017), 106-118.
14. Amin M Khan, Felix Freitag, On Participatory Service Provision at the Network Edge with Community Home Gateways, The 8th International Conference on Ambient Systems, Networks and Technologies, Procedia Computer Science 109C (2017): 311-318. DOI: 10.1016/j.procs.2017.05.357
15. Jose Santos, Tim Wauters, Bruno Volckaert, Filip De Turck, Fog Computing: Enabling the Management and Orchestration of Smart City Applications in 5G Networks, Entropy, 20(4), pp.2-26, December 2017. DOI: 10.3390/e20010004
16. Michael Haus, Aaron Y Ding, Jorg Ott, Managing IoT at the Edge: The Case for BLE Beacons, Proceedings of the 3rd Workshop on Experiences with the Design and Implementation of Smart Objects, pp. 41-46, Snowbird, Utah, USA — October 16 - 16, 2017.
17. Aniello, Leonardo, Baldoni, Roberto, Gaetani, Edoardo, Lombardi, Federico, Margheri, Andrea and Sassone, Vladimiro (2017) A prototype evaluation of a tamper-resistant high performance blockchain-based transaction log for a distributed database , Geneva, Switzerland. 04 - 08 Sep 2017. 4 pp.
18. Edoardo Gaetani, Leonardo Aniello, Roberto Baldoni, Federico Lombardi, Andrea Margheri, and Vladimiro Sassone. Blockchain-based database to ensure data integrity in cloud computing environments. ITA-SEC. CEUR-WS.org, 2017.
19. Q. Xia, E. B. Sifah, A. Smahi, S. Amofa, and X. Zhang, ‘‘BBDS: Blockchain-based data sharing for electronic medical records in cloud environments,’’ Information, vol. 8, no. 2, p. 44, 2017.
20. Xia, E. B. Sifah, K. O. Asamoah, J. Gao, X. Du, and M. Guizani, ‘‘MeDShare: Trust-less medical data sharing among cloud service providers via blockchain,’’ IEEE Access , vol. 5, pp. 14757–14767, 2017. [Online]. Available: http://ieeexplore.ieee.org/document/7990130/
21. A Case Study for Blockchain in Healthcare, “MedRec” prototype for electronic health records and medical research data (White Paper) by Ariel Ekblaw, Asaph Azaria,

John D. Halamka, Andrew Lippman, published by MIT Media Lab, Beth Israel Deaconess Medical Center (August 2016).
22. ProvChain: A Blockchain-Based Data Provenance Architecture in Cloud Environment with Enhanced Privacy and Availability (citation not available as of yet)
23. Security and Privacy in Decentralized Energy Trading through Multi-signatures, Blockchain and Anonymous Messaging Streams (citation not available as of yet)
24. Kamanashis Biswas, Vallipuram Muthukkumarasamy, Securing Smart Cities Using Blockchain Technology, 18th IEEE International Conference on High Performance Computing and Communications; 14th IEEE International Conference on Smart City; 2nd IEEE International Conference on Data Science and Systems, Sydney, Australia, 12-14 Dec. 2016. DOI: 10.1109/HPCC-SmartCity-DSS.2016.0198
25. Khan, Z., Pervez , Z. and Abbasi, A. G. (2017) Towards a secure service provisioning framework in a smart city environment. Future Generation Computer Systems, 77. pp. 112-135. ISSN 0167-739X
26. Abbasi Abdul G, Khan Zaheer, (2017) VeidBlock: Verifiable identity using blockchain and ledger in a software defined network. In: SCCTSA2017 co-located 10th IEEE/ACM Utility and Cloud Computing Conference, Austin, Texas, 5-8 December 2017., pp. 173-179
27. Khan, Z., Dambruch, J., Peters-Anders, J., Sackl, A., Strasser, A., Frohlich, P., Templer, S. and Soomro, K. (2017) Developing knowledge-based citizen participation platform to support Smart City decision making: The Smarticipate case study. Information, 8 (2). p. 47. ISSN 2078-2489
28. Berntzen, L., & Johannessen, M. R. (2016). The Role of Citizen Participation in Municipal Smart City 527 Projects : Lessons Learned from, 299–314. Edited book: Smarter as the New Urban Agenda: A Comprehensive View of the 21st Century City. https://doi.org/10.1007/978-3-319-17620-8
29. Arnstein, S. (1969). A ladder of citizen participation. Jaip, 35(4), 216–224. Retrieved from http://lithgow-schmidt.dk/sherry-arnstein/ladder-of-citizen-participation_en.pdf . Last Accessed: 4 March 2018.
30. Ludlow, D., Khan, Z., Soomro, K., Marconcini, M., Metz, A., Jose, R., Perez, J., Malcorps, P. and Lemper, M. (2017) From top-down land use planning intelligence to bottom-up stakeholder engagement for smart cities – a case study: DECUMANUS service products. International Journal of Services Technology and Management, 23 (5/6). pp. 465-493. ISSN 1460-6720
31. Soomro, K., Khan, Z. and Ludlow, D. (2017) Participatory governance in smart cities: The urbanAPI case study. International Journal of Services Technology and Management, 23 (5/6). pp. 419-444. ISSN 1741-525X
32. Juan-Gabriel Cegarra-Navarroa, Alexeis Garcia-Perez, José Luis Moreno-Cegarraa, Technology knowledge and governance: Empowering citizen engagement and participation, Government Information Quarterly, Volume 31, Issue 4, October 2014, Pages 660-668
33. Blockchain: The next innovation to make our cities smarter, pwc, online: https://www.pwc.in/assets/pdfs/publications/2018/blockchain-the-next-innovation-to-make-our-cities-smarter.pdf
34. Philip Boucher, Susana Nascimento, Mihalis Kritikos, , How blockchain technology could change our lives – in-depth analysis - European Parliamentary Research Service, Scientific Foresight Unit (STOA) – PE 581.948. February 2017. Available online:

http://www.europarl.europa.eu/RegData/etudes/IDAN/2017/581948/EPRS_IDA(2017)581948_EN.pdf
35. Distributed Ledger Technology: beyond block chain. A report by the UK Government Chief Scientific Adviser, 2016. Available online: https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/492972/gs-16-1-distributed-ledger-technology.pdf
36. Enterprise Blockchain | Guardtime. https://guardtime.com/. Last Accessed: 12 March 2018.
37. B. Kaliski, "PKCS #7: Cryptographic Message Syntax - Version 1.5", RFC: 2315, organisation RSA Laboratories, East, March 1998
38. Abbasi, AG, "CryptoNET: Generic Security Framework for Cloud Computing Environments" published in 2011 by KTH Royal Institute of Technology.
39. Sharma, P. K., Moon, S. Y. and Park, J. H. (2017) Block-VN: A Distributed Blockchain Based Vehicular Network Architecture in Smart City. Journal of Information Processing Systems, .13, No.1, pp.184~195, February 2017. 5
40. Xiong, Z.m Zhang, Y., Niyato, D., Wang, P. and Han, Z. (2017) When Mobile Blockchain Meets Edge Computing. IEEE Communications Magazine 56(8): 33-39. 2018
41. Article, “Pattern: Microservice Architecture”, https://microservices.io/patterns/microservices.html
42. Article, "Dropwizard is a Java framework for developing ops-friendly, high-performance, RESTful web services", downloaded from https://www.dropwizard.io/1.3.8/docs/
43. Java Messaging API implementation, “Apache kafka”, downloaded from https://kafka.apache.org/
44. Bouncy Castle, downloaded from https://www.bouncycastle.org/
45. Khan, Minhaj Ahmad, and Khaled Salah. "IoT security: Review, blockchain solutions, and open challenges." *Future Generation Computer Systems* 82 (2018): 395-411.
46. Licheng Wang, Xiaoying Shen, Jing Li, Jun Shao, Yixian Yang, Cryptographic primitives in blockchains, Journal of Network and Computer Applications, Volume 127, 2019, Pages 43-58, ISSN 1084-8045,
47. Reyna A, Martin C, Chen J, Soler E, Diaz M, 2018, On blockchain and its integration with IoT. Challenges and opportunities, Future Generation Computer Systems, 88 (2018), 173-190.
48. Luong NC, Xiong Z, Niyato D, (2018), Optimal Auction For Edge Computing Resource Management in Mobile Blockchain Networks: A Deep Learning Approach, IEEE International Conference on Communications, 20-24 May 2018, Kansas City, MO, USA. ISSN: 1938-1883, DOI: 10.1109/ICC.2018.8422743
49. Xiong Z, Zhang Y, Niyato D, Han Z, (2018), When mobile blockchain meets edge computing, IEEE Communications Magazine 56 (8), 33-39. DOI: 10.1109/MCOM.2018.1701095
50. Abdur Rahman Md., Hossain M S., Loukas G, Hassanain E, Rahman SS, Alhamid M F., Guizani M, (2018), Blockchain-based Mobile Edge Computing Framework for Secure Therapy Applications, IEEE Access, 6(2018), pp. 72469 - 72478. DOI: 10.1109/ACCESS.2018.2881246
51. Xu Y, Wang G, Yang J, Ren J, Zhang Y, Zhang C, (2018), Towards Secure Network Computing Services for Lightweight Clients Using Blockchain, Wireless Communications and Mobile Computing, DOI: https://doi.org/10.1155/2018/2051693
52. Stanciu A, (2017), Blockchain based distributed control system for Edge Computing, 21st International Conference on Control Systems and Computer Science, pp. 667 - 671, 29-31 May 2017, Bucharest, Romania. DOI: 10.1109/CSCS/2017.102

53. Davenport Amanda, Shetty Sachin, Liang Xueping, (2018), Attack Surface Analysis of Permissioned Blockchain Platforms for Smart Cities, 2018 4th IEEE International Smart Cities Conference (ISC2), pp. 1-6, 16-19 September 2018, Kansas City, Missouri, USA.
54. Xiong, Z., Zhang, Y., Niyato, D., Wang, P., & Han, Z. (2018). When mobile blockchain meets edge computing. IEEE Communications Magazine, 56(8), 33-39.
55. Chen, J. (2018). Devify: Decentralized internet of things software framework for a peer-to-peer and interoperable iot device. ACM SIGBED Review, 15(2), 31-36.
56. Pahl, Claus, Nabil El Ioini, and Sven Helmer. "A Decision Framework for Blockchain Platforms for IoT and Edge Computing." In IoTBDS, pp. 105-113. 2018.
57. Jo, Byung, Rana Khan, and Yun-Sung Lee. "Hybrid Blockchain and Internet-of-Things Network for Underground Structure Health Monitoring." Sensors 18, no. 12 (2018): 4268.
58. Wright, K. L., Espinoza, M., Chadha, U., & Krishnamachari, B. (2018). SmartEdge: A Smart Contract for Edge Computing. 2018 IEEE Confs on Internet of Things, Green Computing and Communications, Cyber, Physical and Social Computing, Smart Data, Blockchain, Computer and Information Technology, Congress on Cybermatics.
59. Von Leon, David & Miori, Lorenzo & Sanin, Julian & El Ioini, Nabil & Helmer, Sven & Pahl, Claus. (2019). A Lightweight Container Middleware for Edge Cloud Architectures. In Rajkumar Buyya, Satish Narayana Srirama (Ed.), Fog and Edge Computing: Principles and Paradigms (pp. 145-167). John Wiley & Sons, 6 Feb 2019.

Appendix A: Chaincode Header

Metadata:

[

ref: Reference number of chaincode header block
version: Distributed Ledger version number, currently it is i0.7
hashPrevBlock: Hash of previous block
creationTime: Chain code creation time
extbits: Extra bits for future purposes
nonce: Random value, for future purposes
height:Position of block in header chain
hashMerkleRoot: Accumulated hash of all transaction in this chaincode
creator: The owner of the chaincode
chainName: User friendly name of chaincode, used for searching purposes
smartcontract: see Appendix C
creatorSignature: Signature of the source (sender), used for source authenticity
creatorURL: Used to verify public key of the sender
signedBy: Identity of the source distributed ledger instance
signerUrl: Used to verify the public key of the ledger
signature: Signature on the the above fields

]

Example in JSON format

{
"**ref**":"774e67546c757a4c53354a346d79536c6c336f45573b79584143654753643333",
"**version**":"0.7",
"**hashPrevBlock**":"K4l2vJ6Y7KvNS4lzQsMDSMOTDYIpVh6mXl2TWOre0aA=",
"**creationTime**":"2018-03-20 11:33:24",
"**extbits**":"",

"**nonce**":"55545748624763636246444f584d4c3b",
"**height**":2,
"**hashMerkleRoot**":"MazKoKfLZH1LA0USi9BvlvMv+jfQxTg0gZJLM5YZzjc=",
"**creator**":"43293293",
"**chainName**":"propose-park-area",
"**smartcontract**":"{"scope":"*LOCAL*","payloadSupportingTypes":["org.acreo.proposal.launch.
entities.PlanningProposal","org.acreo.proposal.launch.ConsensusResponse\"],"start":
1521542003861,"end":1522834403861,"securityLevel":"*DIGITAL_SIGNATURE*"}",
"**creatorSignature**":"aal4+2HHPBhT3CiE/rAs9VkmwHWkbDqay1p0L+uOiRGDRq6Uriw4qnPKf==",
"**creatorURL**":" http://localhost:9000/auth",
"**signedBy**":"653456706",
"**signerUrl**":"http://localhost:10000/pubkey",
"**signature**":"mZz6Ox0rFjrxMFszBVM/pDdDFFWpR1MCzdsHZ3F7kGR0oL2zXXv==”
}

Appendix B: Transaction block e.g. PlanningProposal publication

Metadata

[

Ref: Reference number of chaincode header block
depth: Position of transaction in chaincode
hashPrevBlock: Hash of previous transaction
creationTime: Transaction creation time
scope: It should be the same as smart contract but can be used in future for fine grained access control
sender: The owner of the transaction
receiver: Recipients of the transaction
payload: Actual data
payloadType; Types of the data stored in the payload
cryptoOperationsOnPayload: Cryptographic functions, inherits from smart contract Security Level attribute
creatorSignature: Signature of the source (sender), used for source authenticity
creatorURL: Used to verify public key of the sender
signedBy: Identity of the source distributed ledger instance
signedDate: Transaction signing time
signature: Signature on the the above fields
signerUrl: Used to verify the public key of the ledger

]

Example in JSON format

{
"**ref**":"774e67546c757a4c53354a346d79536c6c336f45573b79584143654753643333",
"**depth**":1,
"**hashPrevBlock**":"MazKoKfLZH1LA0USi9BvlvMv+jfQxTg0gZJLM5YZzjc=",
"**creationTime**":"2018-03-20 11:40:05",
"**scope**":"LOCAL",
"**sender**":"43293293",
"**receiver**":"",
"**payload**":"MIAGCSqGSIb3DQEHAqCAMIACAQEVwpbcLQWwRqqXAG1FrmnHgfbAAAAAAAA",
"**payloadType**":"org.acreo.proposal.launch.entities.PlanningProposal",
"**cryptoOperationsOnPayload**":"DIGITAL_SIGNATURE",
"**creatorSignature**":"kmewUe7sJp4Ae45JTbrv3VqXUciBx+jYxA9eSL7sIAY3CIUU2oz+4PUM80fNi==",
"**creatorURL**":"http://localhost:9000/auth",
"**signedBy**":"653456706",
"**signedDate**":"2018-03-20 11:40:05",

"**signerUrl**":"http://localhost:10000/pubkey",
"**signature**":"dLLCvxbaIJJdnuM+lAU8vFfTlZCrnqc3+2TX9tkeBlTSOzRM3AalHbsKTL3LHdj089r2=="
}