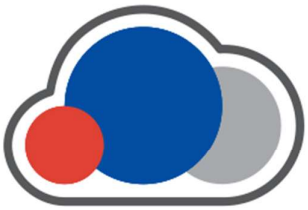

DECENTER

Decentralised technologies
for orchestrated Cloud-to-Edge intelligence

D4.2

Initial Report on Cross-Border Application Data Management

14/1/2020

Revision history

Administration and summary		
Project acronym:	DECENTER	
Document identifier:	D4.2: Initial Report on Cross-Border Application Data Management [M18]	
Leading partner:	UL/KETI	
Report version:	1	
Report preparation date:	14/1/2020	
Classification:	PU (Public)	
Nature:	R (Report)	
Author(s) and contributors:	Uroš Paščinski, Janez Brežnik, Petar Kochovski, Sandi Gec, Vlado Stankovski, (UL) Jaewon Moon (KETI)	
Status:	-	Plan
	-	Working
	-	Draft
	x	Final

The DECENTER Consortium has addressed all comments received, making changes as necessary. Changes to this document are detailed in the change log table below.

Date	Edited by	Status	Changes made
05/11/2019	Uroš Paščinski	Plan	Table of Contents
06/11/2019	Jaewon Moon, Orlando Avila Garcia	Plan	Revised Table of Contents
29/11/2019	Jaewon Moon	Working	Add KETI's contribution (sections 2.2 and 3)
02/12/2019	Uroš Paščinski	Working	Added content to SOTA
04/12/2019	Uroš Paščinski	Working	Updates to SOTA
20/12/2019	Seungwoo Kum	Working	Returned comments on KETI's contribution
24/12/2019	Jaewon Moon	Working	Comments addressed
02/01/2020	Janez Brežnik, Petar Kochovski	Working	Content consolidation
03/01/2020	Seungwoo Kum, Levent Gurgen	Working	Internal review
04/01/2020	Petar Kochovski	Working	Comments addressed
14/01/2020	Vlado Stankovski	Final	Final version

Notice that other documents may supersede this document. A list of latest *public* DECENTER deliverables can be found at the DECENTER Web page at https://www.decenter-project.eu/.

Copyright

Acknowledgements

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no. 815141 (DECENTER: Decentralised technologies for orchestrated Cloud-to-Edge intelligence)

This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT/IITP) (No. 1711075689, Decentralised cloud technologies for edge/IoT integration in support of AI applications).

The partners in the project are FONDAZIONE BRUNO KESSLER (FBK), ATOS (ATOS), COMMISSARIAT À L'ENERGIE ATOMIQUE ET AUX ENERGIES ALTERNATIVES (CEA), COMUNE DI TRENTO (TN), ROBOTNIK (ROB), UNIVERZA V LJUBLJANI (UL), KOREA ELECTRONICS TECHNOLOGY INSTITUTE (KETI), GLUESYS (GLSYS), DALIWORKS (DW), LG U+ (LGUP), SEOUL NATIONAL UNIVERSITY (SNU).

The content of this document is the result of extensive discussions within the DECENTER © Consortium as a whole.

More information

Public DECENTER reports and other information pertaining to the project are available through DECENTER public Web site under http://www.decenter-project.eu.

Contents

List of figures

List of tables

Executive Summary

The essential idea in this deliverable is to describe new mechanisms for regulated, policy based cross-border data management. In the Big Data era, the variety, velocity, volume and other aspects of data are prominent and are currently being addressed by many researchers world-wide. Therefore, the goal of this work is not to implement cross-border data management mechanisms for all types of Big Data. Instead, we focus on the essential requirements of the DECENTER use cases, one of which is the AI processing of video-streams. In this context, a basic Big Data pipeline starts from a camera that provides a raw video stream. The AI processing models/methods are distributed to various resources, which means that we need to deal with different input and output data streams. In the final stage, an outcome of the AI process takes the format of a structured file with specific information derived from the video-stream, for example, the identity of the person shown on the video.

The cross-border data management scenario assumes that the described Big Data pipeline may start at one provider, such as a camera resource owned by the municipality of Trento, Italy, and can proceed through secure (e.g. encrypted) Internet channels towards the other processing application stages, which may be implemented in other administrative domains, for example, private or public Cloud providers in Slovenia, and extended further to Cloud providers in Seoul, Korea. Hence, the term “border” in the context of our work refers to any administrative, organisational policy or government regulation in which the data stream, data or information file, should pass and under which conditions it must abide to specific policies and regulations, requirements for certification, permissions, including personal permissions and preferences.

The goal of this work is therefore to analyse, design and deploy specific cross-border data management mechanisms that enable the participating entities control all aspects of the data transport and management when it comes to their administrative domains. Apparently, the requirements extracted by specific parties taking part in a data management scenario may be too hard in which case it may prove impossible to establish the required quality of cross-border data management and transport. For example, a futuristic European regulation may require certification for dealing with sensitive private data of all cloud providers that process personalised AI models/methods, another futuristic Korean regulation may require to process sensitive data of Korean citizens only on hardware resources that are capable of using strong security mechanisms (e.g. SGX chips). Furthermore, new GDPR-like European legislation may require to empower the citizen with an ability to approve or disapprove the use of her/his own personalised AI model in specific settings, such as at an airport in Korea. In such case, the cross-border data management mechanisms must be designed in a way to allow strict assessment and application of the user’s preferences in the specific context where the use of AI will be required.

In this preliminary design deliverable, we present a cross-border data management use case scenario that focuses on a personalised (in other words biometric) AI model, which is stored in a repository of models in Korea. The repository, designed by KETI, can be used to store sensitive (private) AI models. KETI also implemented mechanisms for injection of a selected AI model from the repository to the running AI (method) container. Based on this functionality, UL designed the cross-border management tasks, which involve cross-border movement of the sensitive AI model, that is, across regulatory and organisation borders. In addition to this, UL experimented with the design of several Smart Contracts that include the use of Smart Oracles to implement (1) full and shared governance of the AI model transport, (2) various monetisation scenarios that could be used in scenarios where the AI models are deployed for temporary use.

Moreover, we designed and developed two exploratory scenarios in order to investigate the ability to manage trust as a high-level property required to realise AI-based fog computing applications, and the ability to develop multi-party smart contracts for multi-party data governance in the cross-border management scenario.

The first exploratory scenario focused on the definition of trust and trust attributes, presented two use cases for trusted video stream access and trusted data flow, which are in the scope of the data management task. Then we presented an architecture and realised the proof-of-concept use cases. The results of this work are encouraging and will inform the final design of the cross-border AI model/data management scenario.

The second exploratory scenario on multi-party service level agreements for multi-party data governance was realised through the use of class-hierarchies of Smart Contracts. It presented seven business models that can be applied to resources, including AI model/data management tasks. The smart contracts actually allow the implementation of various mechanisms, such as consensus before the AI model or data is being transferred from a country to another country, price negotiations to use AI models, condition-based AI model/data management (e.g. based on geolocation or permissions by the user and similar. The implemented smart contract will also serve as basis for the implementation of the cross-border data management scenario in the next phase of the project.

1 Introduction

The goal of this work is to analyse, design and implement innovative mechanisms for regulations, policies, certifications and permissions based, cross-border data management. In the context of this work, borders can be geographic boundaries between states, but also, abstract boundaries of different organisations and their infrastructures, ownerships of resources and services that can dynamically change during the execution of applications and similar. It is therefore necessary to start our work with elicitation of all aspects of the problem at hand. An important aspect of the DECENTER project concerns the application of various AI methods and models on video streams. Such applications can be deployed by the DECENTER's Fog Computing Platform across several countries and administrative domains.

The focus of this work is therefore to design mechanisms that can help the multiple parties, such as physical persons with their privacy and security concerns, different public and private organisations and countries impose their requirements for governance of data that moves across multiple borders. Following is a detailed account of the scenarios that were developed to serve as basis for subsequent implementation in the DECENTER project.

1.1 DECENTER's focus on data management

The illustration in Figure 1 is designed to explain several of the data items that take part in the cross-border data management scenario. The scenario notably involves all sorts of data items, which may be stored and transported by using different protocols (e.g. TCP or UDP) and formats. This includes video streams that originate from a camera device, which can be owned by a private or public entity, video frames that could be stored as files and used as evidence, sensor streams (although not depicted on the figure, represent important aspect of the sensiNact platform that is part of DECENTER's design), private or public (Open Source) AI methods and models that can be stored in repositories ready for deployment and use, extracted metadata i.e. information, which can be sensitive, as well as intermediate results of the AI process in case the process is split in several phases (see deliverables of Task T4.1 for more details about splitting the AI models).

Figure 1. Data management aspects in an application that applies AI models and methods to video streams.

The components of multi-staged AI applications may therefore start at the Edge of the network, then data may move from one organization to another, and from one country to another. Our cross-border data management scenario assumes that a camera resource owned by a construction company in Ljubljana, Slovenia will produce a video stream. Then, this video stream will proceed through secure (e.g. encrypted) Internet channels towards the other processing application stages, which may be implemented in other administrative domains, for example, public Cloud providers in Ljubljana, Slovenia and Seoul, Korea. This means, that sensitive data (e.g. privacy or security related information) may move across several borders. The term "border" in the context of our work therefore refers to any administrative, organizational, policy or government regulation in which the data stream, data, AI model or information file, should pass across and under which conditions it must abide to specific policies and regulations, requirements for certification, permissions, including personal permissions and preferences.

In summary, what we need to assure is the ability to govern the data transported by all concerned participants and actors, such as the owner of the camera in Trento, the cloud providers in Ljubljana and Seoul, the preferences of the user whose private data is being transported from one place to another, and last, but, not the least, the legislation of the states of Italy, Slovenia and Korea, including European legislation. This is a very challenging task for which the DECENTER project may have some practical solutions which are designed and presented in this deliverable.

Our goal is not to address all possible data types and formats, but to focus on one particular example of sensitive AI model movement from one country to be used in another country.

1.2 High-level requirements for cross-border data management

In this sub-section we first analyse the scope of the problem investigated, and we try to pinpoint specific high-level requirements for cross-border data management. For the purpose of illustration, we use the idea of Use Case 3 on Smart and Safe Construction of the DECENTER project. Emerging smart applications address requirements for construction monitoring, construction progress tracking, early disaster warning, safety at work and similar. Expected benefits from using such applications are improved safety, productivity and use of assets and resources among others.

Existing solutions rely on well-known service providers, however, the emergence of construction sites that use many IoT devices (cameras and sensors) raise new issues of privacy, security and trust. This problem becomes particularly concerning when sensitive data, such as video streams have to be processed in leased infrastructures, such as Fog nodes and stored for future use in Cloud storage. Therefore, the focus for implementation considerations can be on the design of a smart container-based application, which is designed to implement video surveillance for safety at work measures as shown in Figure 2, where the AI-empowered application uses video-surveillance footage as input data for the detection of safety violations. For example, a video frame which is an input to deep learning method (e.g. object detection) is used to check if a worker wears a hard hat. Another AI model may be used to identify the identity of the worker. In case the worker does not wear it, a safety alert is issued to the construction supervisor.

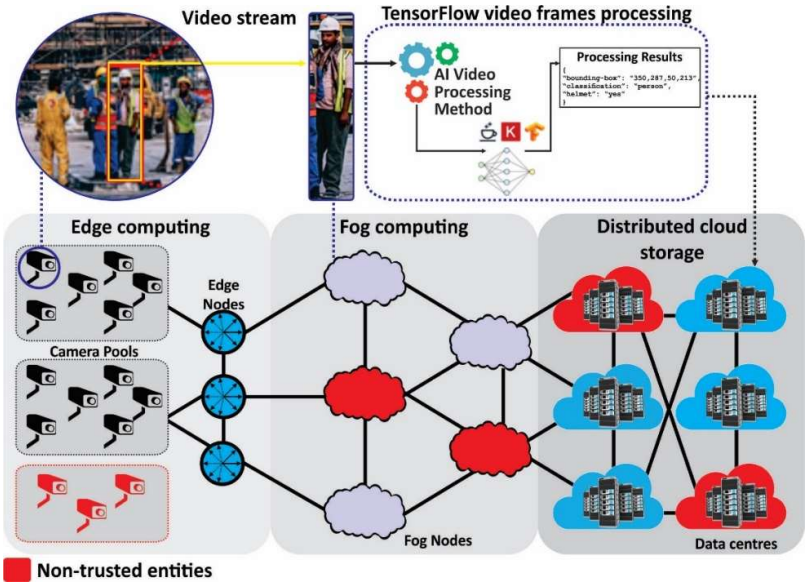

Figure 2. Trust issues in AI applications data flow.

This scenario reveals a variety of data security, privacy, availability, reliability, dependability, Quality of Service, and other trust-related problems. Trust, as a high-level concept can be applied to all scenarios where data management is concerned. For this reason, we have put specific emphasis on investigation of the concept of trust in the context of DECENTER and its intelligent Fog Computing applications. For instance, it is necessary to trust the actual on-site devices, cameras and sensors, that are used to generate input data to the AI methods. Trust issues may be even more critical in the case when the application uses cameras that dynamically enter and exit the smart construction environment, for example, cameras mounted on workers' hard hats. Consequently, it is necessary to build an application that relies on a pool of trusted devices, for example a Trusted Cameras Pool. Furthermore, it is necessary to assure that the Fog nodes used for processing sensitive (video streaming) data are also trusted. The data (video frames) are forwarded to a selected Trusted Fog Infrastructures Pool, where the selected AI methods can be started in order to analyse the incoming video frames and detect safety violations. The privacy-sensitive AI models should also be trusted, and never be moved or used in case the country regulations are not applied.

In case the AI method detects a situation (e.g. construction worker without a helmet), a safety alert is sent to the construction site manager, and a log of the event containing captured video frames could be generated and stored, according to this scenario, on a Cloud storage. Cloud storage is used in the scenario to be able to explore the whole spectrum of cross-border data management issues, as the Cloud storage may spread across country- and administrative-borders, for example, in case it is decentralised (e.g. storj.io). The Cloud storage must also be trusted as storing sensitive data in the Cloud inevitably involves privacy and security issues. Therefore, trust in the fog application requires trust in all entities on the way of the data and information that moves across boundaries.

In summary, physical persons, organisations and countries must establish trust that all data management and movements can happen only when all requirements are fulfilled, including governmental legislation and regulation, organisational policies and personal permissions. From a technical viewpoint trust can be described with some probability, however, the trust management approach employed by the Fog Computing Platform should rely on binary

decisions: trusted or not trusted. Therefore, it is a hard requirement that must be fulfilled in specific circumstances that makes it possible to manage data across administrative and country borders.

1.3 Existing regulations with respect to data collection, processing and storing

With respect to data collection, processing and storing there already exists plethora of regulations and standards. These relate to the problems of: (i) data sets collection, processing or generation, (ii) the data management life cycle for data sets; (iii) the methodology and standards following which data are collected, processed or generated; (iv) whether and how the data are shared and/or made open; and (v) how the data are curated and preserved.

In the context of our data management deliverable, we focus on the uppermost layer of the necessary data management architecture. In this area, we explore various data transport protocols and all other important data management aspects. In order to achieve this goal, the project participants intend to work towards providing a meaningful data management infrastructure that addresses various standards contained in, but not limited to:

- The EU General Data Protection Regulation
- ISO/IEC 29100:2011 and ISO/IEC 27001:2005 - both on Security
- Charter of Fundamental Rights of the European Union 2009
- Council of Europe Convention for the Protection of Individuals regarding Automatic Processing of Personal Data 1980
- Guidelines on FAIR Data Management in H2020 (Findable, Accessible, Interoperable, Reusable)

This deliverable therefore also relies on the ethical principles, outlined in ethical deliverables and the Data Management Plan of the project. The consortium is committed to respect the Initiative for the Budapest Open Access, with the movement of open access to knowledge gained from joining the Berlin Declaration and Institutional Policy on Open Access.

One important research aspect which is tackled under the work in Task 4.3 is the requirement to protect and provide control mechanisms for the use of personal information during the operation of DECENTER applications. All handling mechanisms for personal data must be supported by data handling mechanisms that contribute to trust among the participants. It is also our intention to abide by the principle and methodology of anonymisation, following the recommendation, best practices, and anonymisation techniques described in Article 29 Data Protection Working Party – Opinion 05/2014 on Anonymisation techniques. If any sensitive data will be managed by the DECENTER's software platform and developed applications and services, during the project and beyond, it will be anonymised and rigorously protected. When processing personal data, our new set of solutions for data management aimed to comply with the Data Protection principles which are set out in the General Data Protection Regulation. The Data Management Plan is complemented by an Ethical Protocol with some indications for fulfilling:

- How to manage, store and destroy sensitive data; how to anonymise data which will be made public and openly accessible;
- How to ensure a constant quality control for the collected, generated and managed data;

- How to use the data collected and/or produced during the project, respecting the privacy of all participants, the intellectual property and the exploitation in further research of the collected findings.

All data and accompanying DECENTER software will be handled and operated only by qualified members of the consortium under confidentiality agreements, who will ensure that data access, data protection and privacy standards are in compliance with national and European regulations.

1.4 Achieving dynamic AI application dataflow

Based on the current analysis of DECENTER's use cases, sensitive AI models represent an important part of the whole system. KETI's initial work in this context concentrated on establishing mechanisms to (1) store AI models in repository, (2) provide basis for data management operations, such as injection of AI models into running applications, (3) design of AI method containers that can accept injected AI modes at runtime. It is therefore a rational decision in this project to focus on the sensitive AI model as an essential data item that can be managed and operated across countries borders. The key part of our cross-border data management design is developed to address the requirements for such AI model (data) management operations.

In DECENTER, we are focusing on delivering AI on to edge device through the DECENTER Platform, and for that purpose an architecture for AI application on the Fog is being defined. The current AI application architecture can be found in Deliverable D4.1.

Following this architecture, we also describe the interaction between newly designed AI model repository and the AI microservice. The proposed server takes into account that the containerized application (the AI microservice) to be deployed on the edge does not include a static main AI model. The AI microservice on the edge device can import the necessary model directly from the AI model repository. The AI microservice on the Edge can import static AI model, but also can find and import the appropriate AI model based on the real-time needs.

1.5 Goals

The goals of this work are therefore to:

- Provide means of multi-party governance of AI model transport across borders to realize end-to-end, cross-border AI application on the Fog Platform. Borders are understood as personal permissions and requirements, organisational policies, and country legislation and regulations, such as GDPR for private or sensitive data management.
- Develop a repository that can help store, search and reuse existing (potentially sensitive) AI models.
- Provide methods for AI method injections into running AI method containers. This activity is being realised along with other activities of WP4.
- Blockchain-based methods and techniques to verify the trustworthiness and security of the various actors, such as physical persons, camera and IoT device providers, cloud providers, cloud storage providers, AI model providers and similar.
- Analyse trust-related solutions as key aspect that has to be addressed in cross-border data management, particularly when the data passes through the "Fog".

- Implement initial trust-management scenarios based on Smart Contracts.
- Implement multi-party Smart Contracts, initially for the monetisation approaches for various Fog Computing resources and services, that have the potential to be upgraded and be used for ***multi-party data governance***.

1.6 Scope of the document

This document provides details on the basic methods used, the background of this study, including analysis of trust-related approaches and concepts. It proceeds with design considerations for the repository of AI models. It then presents the full scenario of cross-border data management which is prepared for implementation in the following stages of the project. Moreover, two studies are presented that intend to highlight the ability (1) to manage trust in the Fog Computing environment, and (2) to realise multi-party Smart Contracts that can be used for various cross-border data management scenarios including the potential of monetisation of the AI models.

2 Background

2.1 Blockchain, Smart Contracts and Smart Oracles

2.1.1 Blockchain

Our cross-border AI model management scenario relies on the use of blockchain. In order to prepare a self-complete document, in the following passages we explain some of the essentials. Further to that, we also explain several trust-related approaches for Fog Computing, some of which, rely on Blockchain based services.

Blockchain is a linked-list data structure that forms an ordered list of reverse-linked transactions blocks. The structure is designed to withstand network attacks and can be stored in files or a simple database [5].

In many uses, Blockchain has been seen as a new form of business or economy because it allows the elimination of the intermediate and exchange transactions in a decentralised way; thus, it enables global scale, transparent, auditable and secure applications [6], [7]. Decentralisation refers to the decisions taken by the network (user group) to maintain its consistency (via Proof-of-Work - PoW or other consistency mechanisms), so a change in the network needs to be agreed by the majority of network participants, making the decision not been monopolised by a certain group [8].

The need for global and decentralised applications are emerging by the amount of data generated by different services, such as traffic, healthcare, government, logistics, marketing, electrical and safety, and the distrust of providers which are responsible to store data [9] as a consequence of the recent history of leaking information by them.

Although decentralisation in the context studied is linked to the use of Blockchain, other protocols can provide decentralisation of applications, especially banks, such as the Ripple, which has a consensus record and a native currency called XRP (also called ripples). Ripple allows the creation of a global, decentralised network of computers to make payments quickly, cost effectively and safely [10].

Another example is the Stellar, which is implementing the Stellar Consensus Protocol (SCP), based on Byzantine agreement, which seeks reaching an understanding between all the participants in a distributed network; thus, aiming to ensure low network latency, high reliability, decentralised control and security [11].

As Cloud computing, Blockchain is also formed by a set of techniques that allows the creation of a safe model of decentralised transactions, free of intermediaries. The fundamental technologies that enable the Blockchain are: timestamp server, PoW, Merkle trees and cryptographic hash functions.

The block is the fundamental structure of the chain. It stores the transactions and is divided into three parts: a header, a transaction counter and a transaction. The header has three metadata blocks: the first refers to the hash of the previous block; the second adds the data from the timestamp, difficulty and nonce; and the last block is the root of the Merkle tree.

A server timestamp adds the date and time to a specific data at the present time which was created by a computer, making sure that data existed at a certain point of time. To produce

this, it is necessary for an authority called Time Stamp Authority (TSA) to publish and verify the timestamp. However, before this process, the data must be encrypted using a hash function [12].

In the case of Blockchain, it serves as a certifying of the creation date for data, with no need of a central authority. Blockchain also provides integrity to the data, because each block has the previous timestamp in its hash, thus forming a chain that makes it difficult to change the timestamp. After confirming the timestamp by the network, the difficulty to modify it progressively becomes almost impossible as the chain becomes longer, thus increasing the confidentiality of the data [13].

PoW is a principle, first proposed by Dwork and Naor [14], where a user must compute a moderate-to-difficult function to access a resource, for example, increase processing performed in the Central Processing Unit (CPU). The original idea was aimed to decreasing the amount of spam; however, it is used in various contexts, such as: combating DoS attacks, connection exhaustion or abuse services, evaluation of website popularity, under overload situations, as a currency for p2p and grid applications, and even in micropayments [15].

Blockchain uses the nonce to vary the output of the calculation of PoW, thus hampering the work performed by the customer to find out the original hash and thereby increasing the protection of the distributed infrastructure.

Merkle Tree are complete binary trees with pointers hash proposed by Merkle [16] in order to produce digital signatures of a single, one-time use. It is designed such that data blocks are grouped into pairs and the hash of each is stored in its parent nodes. The parent nodes are then grouped into pairs and their hashes are stored one level above. The process continues until the root node is reached [17].

Generally, the Merkle tree is used to summarise and check the integrity of a large data volume. In Blockchain, Merkle tree is used to summarise all the transactions into a block, this keeping the history of all transactions already included in the Blockchain [5]. A copy of the Blockchain is stored locally, and is constantly updated with new blocks as they appear in the network. However, the storage requirement to store all the blocks is gradually increasing and may end up consuming too much disk space. Instead when using the Merkle tree in Blockchain, only the root of the tree that contains the summary of transactions needs to be stored, which significantly reduces the required storage space [13].

Hash functions refer to a family of non-injective functions that compress certain data of any size (for example a string, binary files or TCP packages) into a string of a fixed size. They are essentially one-directional functions, meaning that while it is simple to compute a hash of some value, calculation of one of its inverse values is computationally impractical. It is therefore difficult to restore the original value from its hash. As a consequence, hash functions can be used in applications that require privacy, authenticity, integrity and non-repudiation of the carried information.

Another advantage of using hash functions is the reduction of storage space, if the application allows to store hashes instead of the original data, given that the original data is larger than its corresponding hash value [8]. Among various applications forms hash functions can highlight password protection, digital signatures construction, building blocks in authentication protocols, and cryptographic algorithms structuring [18].

Blockchain uses a variant of the Secure Hash Algorithm (SHA), namely SHA-256, for calculation of hashes in order to verify the transactions and calculate the PoW. SHA-256 is a hash type that turns smaller two 64-bit messages into a hash of 256 bits. Therefore, SHA-256 uses the iterative structure of Merkle-Damgard [19] to transform a resistant collision and fixed-size compression function into a hash function that accepts values of any size [17].

Briefly, Blockchain allows the creation of a distributed architecture where each node runs and stores a linked transaction block, encrypted using a hash function. Each block, added one at a time, refers to the previous block and is verified through the PoW by network peers connected to the network. Hence, a consensus across the network is maintained, as any attempt to change a block breaks the linking sequence.

Despite offering protection against network attacks, the technology is not immune to vulnerabilities and may suffer attacks that compromise the entire security aspect which is proposed. The main attacks on Blockchain are the majority hash rate attack (i.e. 51% attack) and double spending [8], [17].

Although the vulnerabilities are recognised, they are mitigated by the design of Blockchain bitcoin, such as the need for six confirmations by the network of a transaction, or by preventing a user and/or user group, having considered the amount of computer control [1].

Using blockchain and SCs within existing Cloud architectures has much potential. Carminati et al. [27] investigated blockchain as a platform for secure inter-organisational business processes management. Zhang et al. [28] presented TOWN CRIER (TC) aiming to provide trustworthy (trustful) data to SCs through a middleman service (TC Server).

2.1.2 Smart Oracles

Smart Oracles are useful means that reduce the necessity of costly operations on a blockchain, such as storing and using data within SCs. For example, a Blockchain Oracle is a service that provides a SCs with off-chain data. Specifically, external data provided by Smart Oracles can be used within an SC in order to decide, if a AI model operation should be performed or not, if a Cloud provider or Cloud storage could be trusted, an AI model injected in a running AI method container, and similar actions, automatically. Advanced Smart Oracle solutions, such as Oraclize[1] provide Smart Contract templates, which ensure Oracle correct data flow. Another Smart Oracle solution is the Ethereum based Chainlink network[2] that provides reliable tamper-proof inputs and outputs for SCs on any blockchain. These few useful studies form the basis for the present work, which aims at using blockchain, SCs and Smart Oracles to provide regulatory and policy enforcement aspects, alongside with transparency, traceability and a great level of autonomy to the DECENTER's Fog Computing Platform.

2.2 AI Model Management and Related Techniques

2.2.1 AI model as Data

Artificial intelligence is anything which enables devices to have a brain like a human. The device can have intelligence by receiving a proper AI model. In terms of AI applications,

[1] http://www.oraclize.it/.
[2] https://chain.link/.

application-related data must be transferred from the cloud to the target device to maintain reliable service. Cloud must manage AI-application-related data efficiently and deliver proper data to the device. In that case, the device that receives the related data including AI model can do a specific task such as decision making, classification, and prediction.

There are works on AI model deployment on the field, however those works are focusing no AI models onto cloud resources rather than edge resource, which are usually have very limited resources. Therefore, it is not easy to explicitly manage, register and utilize modified models according to the service characteristics and purposes. In this project, micro-service is considered as an independent unit of monolithic service in allocated resources. Therefore, the AI model repository is designed to distribute AI models in various units of micro-service.

We suggest the AI repository server which is able to distribute AI models to each microservice. Stored AI models can be easily distributed to microservices by connection between AI model repository and each micro-service. For example, if the resource on which micro-service is installed has relatively poor performance, it may be better to choose a lightweight AI model to use, even if the accuracy is degraded. We take this requirement for the future scenario and designed a repository server deploying AI models efficiently.

2.2.2 Formats and Serialization Methods of AI Models

A trained AI model is serialized and stored in a file for future use. For example, a store AI model file can be transmitted to another device (such as Fog node) to be used in an AI application, or can be loaded onto memory for re-training. Although the needs of a standard format for the AI model serialization are high, there are multiple serialization formats being used in the field. Several famous formats for AI serialization are described in this section.

2.2.3 Hierarchical Data Format

Hierarchical Data Format version 5 (HDF5)[3] is a data format and a library designed for efficiently storing and accessing large amounts of heterogeneous data. It was initially developed as a tool to store large amounts of complex data and is mainly used for storage of large-scale image data, flight recordings of aircraft and ships. It supports an unlimited variety of data types and is very flexible for handling efficient I/O and complex data. The HDF5 library can be seen as a kind of an embedded database that does not provide a database management system itself, unlike some other systems. When used with Keras, the high-level artificial neural network framework, the weights of a model can be stored into the HDF5 format while the model architecture can be saved using JSON or YAML format.

2.2.4 Protocol Buffer

Protocol buffer[4] is a protocol developed by Google and used for serialising structured data. It is extensible, and, unlike the pickle module, programming language- and platform- agnostic. Thus, protocol buffers can be written in various programming languages, including Java, C++ and Python. When used with the TensorFlow artificial neural network framework, both the graph definition and the weights of a model can be represented as a protobuf (.pb) file.

---

[3] Folk, M., Heber, G., Koziol, Q., Pourmal, E., & Robinson, D. (2011, March). An overview of the HDF5 technology suite and its applications. In *Proceedings of the EDBT/ICDT 2011 Workshop on Array Databases* (pp. 36-47). ACM.

[4] https://developers.google.com/protocol-buffers

2.2.5 Open Neural Network Exchange Format

The Open Neural Network Exchange Format (ONNX)[5] is a new open standard format for representing an AI model. The ONNX enables AI models to be trained in a specific framework, such as CNTK or TensorFlow, and to be later reused in another framework for inference. ONNX models are therefore interoperable and supported in common famous frameworks such as Caffe2, CNTK, TensorFlow, Core ML, MXNet, and PyTorch. The ONNX is based on protobuf in which strings can identify types of elements in the model's graph. A predefined dictionary of operators and definitions is provided. Model can be saved in the model.onnx format to use ONNX for interoperability. This format is serialised representation of the model in a protobuf file.

2.2.6 Neural Network Exchange Format

Similar to the ONNX, the Neural Network Exchange Format (NNEF)[6] is an exchange format to execute networks trained with different frameworks on different platforms. The NNEF is intended to describe the network structure and the trained parameters of the model and defines a container that contains a network structure and a set of tensor data files. The network structure includes used operations and activation functions. The goal of the NNEF is to enable a developer who made an AI model to easily transfer the trained networks into a wide variety of inference engines. This format can find its use when the intelligence is required on a variety of edge devices which makes an AI model widely available.

2.2.1 Pickle

The pickle[7] module implements binary protocols for serialising and deserializing Python object structures. As the serialisation uses Python-specific data format that is suitable for representation of common Python objects, the de-serialisation is typically done exclusively in Python and not in some other programming language. In practice, the pickle module has various use cases; one notable AI library that utilises it is the Scikit-learn package.

2.3 Confidential computing trends for AI

2.3.1 Confidential Computing Consortium

In August 2019, the Linux Foundation announced and established the Confidential Computing Consortium to improve security for data in use, and to define and accelerate the adoption of confidential computing. The community will embody open governance and open collaboration that has aided the success of similarly ambitious efforts, which includes commitments from Alibaba Cloud, Arm, Baidu, Google Cloud, IBM, Intel, Microsoft, Red Hat, Swisscom and Tencent.

Participants plan to make several open source project contributions to the Confidential Computing Consortium, including: Intel Software Guard Extensions (Intel SGX); Microsoft

[5] https://onnx.ai/
[6] https://www.khronos.org/nnef
[7] https://docs.python.org/3/library/pickle.html

Open Enclave SDK; Red Hat Enarx; and Google's Federated Learning, Asylo and Private Join and Compute.

Confidential computing focuses on securing data in use. Current approaches to securing data often address data at rest (i.e. storage) and in transit (i.e. network) but encrypting data in use is possibly the most challenging step to providing a fully-encrypted lifecycle for sensitive data. Confidential computing will enable encrypted data to be processed in memory without exposing it to the rest of the system and reduce exposure for sensitive data and provide greater control and transparency for users. In the following, some of the projects of those consortia are explained into more details.

2.3.2 Intel SGX

Intel's Software Guard Extensions (SGX) is a set of extensions to the Intel architecture that aims to provide integrity and confidentiality guarantees to security-sensitive computation performed on a computer where all the privileged software (kernel, hypervisor, etc) is potentially malicious. According to Google Scholar, to date more than 5,000 academic and other technical works have been published. The applicability of this technology has conducted research in various areas, including automated application partitioning, functional encryption, secured Linux containers, confidential data synchronisation (e.g. with Apache Zookeeper). Many works have studied various attacks, bugs and exploits on Intel SGX, such as cache attacks, synchronisation bugs, keys extraction with out-of-order and speculative execution.

Despite the vulnerabilities, Intel SGX represents the foundation to designing hardware for confidential computing. In DECENTER, the data management aspect addresses the problem of privacy and security which is tackled by relying on trusted computing environment. Such an environment can be represented by a particular Edge in a particular region or by a group of hosting nodes of a Cloud environment that utilises hardware extensions for confidential computing, e.g. Intel SGX. Related, Microsoft Open Enclave, Red Hat Enarx and Google Asylo, described in the following, provide software tools to facilitate the development of confidential applications by providing hardware-platform-agnostic enclaves.

2.3.3 Microsoft Open Enclave

Open Enclave SDK is an open source SDK targeted at creating a single unified enclaving abstraction for developer to build Trusted Execution Environment (TEEs) based applications. As TEE technology matures and as different implementations arise, the Open Enclave SDK is committed to supporting an API set that allows developers to build once and deploy on multiple technology platforms, different environments from cloud to hybrid to edge, and for both Linux and Windows.

2.3.4 Red Hat Enarx

Enarx is an application deployment system enabling applications to run within TEEs without rewriting for particular platforms or SDKs. It handles attestation and delivery into a run-time "Keep" based on WebAssembly, offering developers a wide range of language choices for implementation. Enarx is CPU-architecture independent technology, enabling the same application code to be deployed across multiple targets, abstracting issues such as cross-compilation and differing attestation mechanisms between hardware vendors. Work is currently underway on AMD SEV and Intel SGX.

2.3.5 Google Asylo

Asylo is an open-source framework for supporting and facilitating the creation and use of enclaves. It is designed to be independent of the underlying hardware platform, which contributes to easier software development, reducing the friction developers experience when building software to run in a confidential computing environment. For example, an application can be built to run in an Asylo enclave on hardware with Intel SGX today, and in the future, it is intended to run on chipsets from other hardware vendors without code changes from the developer as well. Asylo shares similar design goals with Microsoft Open Enclave and Red Hat Enarx.

2.3.6 Google's Federated Learning

Federated learning leverages Edge devices, such as Smart Phones, to collaboratively train a shared AI model while preserving training data on the Edge. Whereas the primary design goal of federated learning -- in contrast to the traditional centralised training -- might be to distribute the training of an AI model across Edge devices, it also has a unique approach with respect to the confidential computing, as the training dataset in its original form never reaches the Cloud. However, depending on the training data and the target AI model, potentially sensitive information might still leak through the model itself. To the data management aspect of DECENTER which focuses on delivering AI models for inference with respect to private data regulations, the federated learning might be complementary.

2.3.7 Google's Private Join and Compute

Private Join and Compute is an open-source cryptographic tool that allows untrusted parties to merge their respective data without having to disclose the data to one another. With the technique called commutative encryption, the data is encrypted with multiple keys and decrypted with a key schedule of an arbitrary order. However, due to the computational intensity, private join and compute is not always feasible.

2.3.8 Microsoft Decentralised and Collaborative AI

Recently, Microsoft open-sourced a framework that allows for training and hosting AI models on the Ethereum Blockchain by utilising Smart Contracts. The framework uses two different incentives to encourage contributions to new models and their updates.

2.4 Analysis of trust management approaches in the context of cross-border data management

Our novel cross-border AI model management approach has to address several high-level requirements. These can collectively be understood as trust-related requirements. Some recent review studies [34], [32], [35] explain that research in trust management has been concentrated in the area of edge computing in general, and mobile computing in particular. There is a notable lack of trust management solutions for Fog computing due to the challenge of monitoring and evaluating the behaviour of a large number of heterogeneous and distributed Fog nodes in the network while maintaining satisfactory QoS.

A large amount of trust management solutions can be separated into two major trust models: evidence-based and monitoring-based trust models [33]. A monitoring-based model can be any model that instantiates trust based on the observed behaviour of past interactions (e.g.

social networking, cooperative solutions, crowd-sourcing, etc.) between entities. For example, Habib et al. [36] developed trust management mechanism that aids users to identify trustworthy Cloud providers that offers two approaches to calculate the trust score of Cloud providers. The default approach implements The Consensus Assessments Initiative Questionnaire (CAIQ) that is provided by the Cloud Security Alliance (CSA) for Cloud consumers and auditors to assess the security capabilities of a Cloud service provider. The second approach allows the user to tailor the trust requirements according to their needs and receive personalised results personalised. Mostajeran et al. [37] proposed a multifaceted run-time trust framework that is based on Edge node's security assessment. The framework monitors the Edge nodes and is capable of identifying Edge-server security vulnerabilities, detect unauthorised actions and thus categorise Edge nodes based on the estimated level of trust. Prajapati et al. [38] introduced a trust management model for calculating the trust level between the user and Software-as-a-Service providers based on past usage experience and implement concepts such as reputation and satisfaction level. Chen et al. [39] described a socially trusted collaborative Edge computing platform for dense networks, which implements payments as an incentive mechanism for edge nodes to collaborate. In particular, instead of offoading the workload to a remote cloud, an overloaded Edge node could pay nearby nodes that have spare computing resources in order to process the remaining workload. The study of Sharma et al. [40] focuses on trust and privacy by building social relationships between the IoT devices, where crowd-sources have a role of mini-Edge servers and entropy modelling for maintenance of trust.

Furthermore, the trust management scheme of Li et al. [41] can provide real-time protection from malicious attacks in traffic. Moreover, the scheme evaluates the trustworthiness of data and nodes (e.g. cars) within a vehicular ad hoc network. It allows us to determine the degree to which data can be trusted and whether nodes can be trusted. Similarly, Wang et al. [42] proposed a Fog-based hierarchical trust mechanism framework for detecting hidden data attacks and ensure communication only with credible edge nodes. In this study, the Fog computing layer acts as a trust buffer between the Cloud computing layer and the sensor network layer. In particular, it detects the trust state of the wireless sensor network, monitors their behaviour and performs analysis tasks. Chen et al. [43] described a distributed trust management protocol, where each user maintains its trust assessment towards devices. This protocol implements a filtering technique based on similarities in the social interests of the IoT nodes' owners. The studies [44] and [45] proposed implementing fuzzy logic for trust management in order to evaluate trust from multiple parameters, such as reputation, authentication, confidentiality and so on.

Evidence-based trust model can be any model that proves trust relationship among entities based on public-key, address, identity or any similar evidence that the entity can generate for itself or other users. For instance, Mora-Gimeno et al. [46] present a security model for data processing that combines applications to multi-tier mobile edge architectures with ability to adjust the security levels of each tier (i.e. each tier requires authentication). The model is based on the degree of trust that each level may have, which requires a different level of security. It determines the number of security mechanisms to be used and the degree of trust for each component.

Lately, a novelty in evidence-based trust models are the blockchain-based trust models, which allow full transparency and traceability. Unlike the previously known models, the blockchain-based trust model is not dependant on a trusted third-party certification authority. Di Pietro et al. [47] proposed a distributed trust model for IoT based on blockchain technology that assures trust between IoT devices. The devices are grouped by domains in so-called "islands of trust", where trust is not established between groups. The model establishes trust among devices

belonging to different domains by implementing a three-way handshaking protocol. Alexopoulos et al. [48] proposed a blockchain-based trust management system that allows global scalability to different clusters. For that purpose, the authors designed a three-layered architecture for trust management in IoT. The first layer is composed of devices in close proximity. The second layer corresponds to a decentralised ledger that governs a specific subset of embedded devices. The third layer is a global decentralised ledger that provides guarantees for access delegations between devices and users belonging to different layers. Yu et al. [49] introduced a trustworthy trading platform, that allows trading of IoT devices or data generated by IoT devices between multiple entities, such as device manufacturers, retailers and users. In order to achieve high trust between entities, they utilise different permissions for each entity when trading with devices and data over the blockchain. As a result, no entity can cheat others or modify the existing data related to a specific device. Hammi et al. [50] introduced a new concept called Bubbles of Trust, which represent secure virtual zones in IoT environments. It groups devices in trust zones. Therefore, only devices that belong to the same zone are allowed to communicate with each other, whereas every other device is considered as malicious. In this study, communications between devices are considered as transactions and are validated by their public blockchain. Missier et al. [1] proposed a conceptual model of a decentralised data marketplace for data produced by Edge devices. Their decentralised marketplace is designed to dictate the data price base on data quality, demand and offer. To provide a higher degree of trust between the producers and consumers of data, the authors propose transparent blockchain transactions.

Our work intends to complement the above efforts by focusing on blockchain-based trust management. In contrast to previous studies the implemented trust management architecture assures trust for complex interactions among humans and artificial agents across the complete Edge-to-Cloud continuum. It also addresses some limitations of previously implemented blockchain-based trust models, particularly, it functions based on less costly off-chain data which is facilitated through the use of trustless Smart Oracles.

3 AI Model (Data) Management Operations and Architecture

This section introduces a scenario of cross-border data management – a Data Management Usage Scenario which is being implemented with our new Data Management Architecture. The scenario addresses a situation of the Korean worker (Kim) visiting a construction site in Slovenia, which is a Member State of the European Union that requires a simple, efficient and rapid identification of the person together with compliance with European and Slovenian legislation regarding the protection of personal data.

3.1 Data Management Usage Scenario

Here, we follow several steps of our cross-border data management scenario that is developed in order to provide mechanisms for multi-party governance of the transport of sensitive AI models across administrative and country borders.

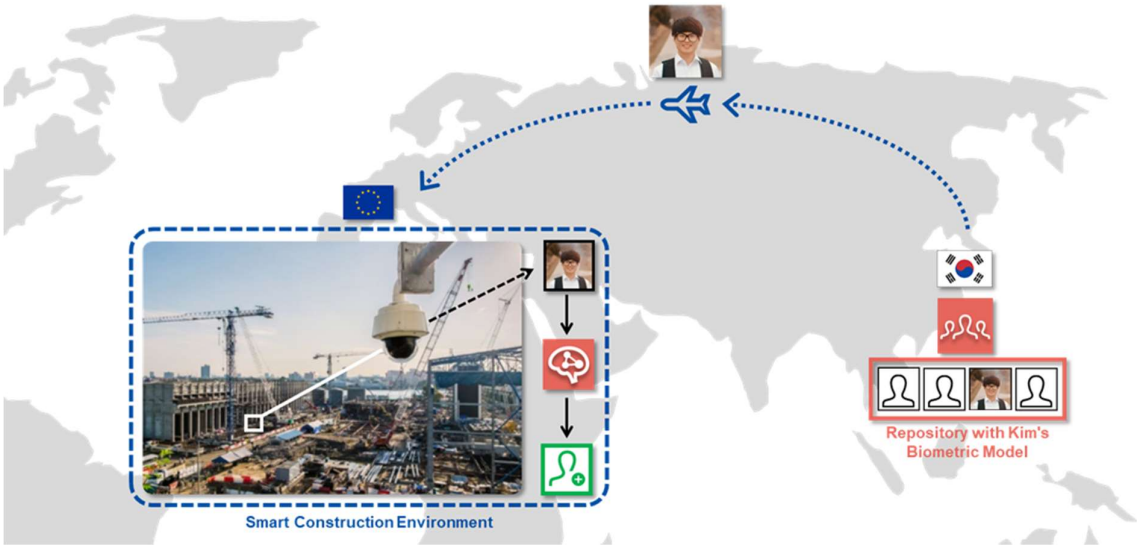

Figure 3. Data management use case scenario. Kim is a resident of the Republic of Korea who visits the construction site in Ljubljana, Slovenia (an EU country), where his access to the site has to be authorised by using his biometric model. The model for face recognition is stored in an AI model repository in Korea.

Setup and requirements

The biometric model to verify Kim is stored in a repository in Korea. To protect his privacy, he decides to use the model that is split into two parts. The inference done in the front part of the AI model is computationally more intense to compute and should be deployed in an edge near the site. The inference in the rear part of the AI model is computationally less intense and is computed in a fog infrastructure that is relatively close to the first infrastructure.

At the construction site entrance, a video camera will record Kim's face for the authorisation purpose. Therefore, the construction site needs to deploy the AI method which uses Kim's biometric model for face recognition. While doing so, Kim wants to make sure that the Korean

privacy laws are respected. To further reduce the possibility of leaking private information, he is interested only in the certified infrastructure providers that use special-purpose hardware, such as Intel SGX, to process the data in a secure enclave. In addition, with respect to Korea's legislation, the EU is allowed to process personal data of Korea's citizens only upon their explicit consent.

Step-by-step use case description

1. Kim turns on his phone and provides consent about processing his personal data (i.e. data management) in the EU.
2. The application sends consent confirmation to the Smart Oracle that is registered for the Smart Contract template of the model repository in Korea. The data sent contains information about the sender, his consent, his origin country and the country he is currently at.
3. The construction company provides to its application service provider information about the repository of AI models to use (its URI), the target AI model to use, Kim's ID and the infrastructures to use (because of certification).
4. The application service provider then uses DECENTER fog platform to compose the application where it selects the parameters provided by the construction company. The application contains specially crafted container stub for the AI model to be fetched from the repository at runtime.
5. The application service provider annotates the application constraints.
6. The app composer passes over application manifest to the resource selector.
7. Resource Selector calculates a possible deployment and passes over the deployment plan to the GUI and Data Management Module.
8. Data Management Module grants access based on a) fulfilled EU regulations for certification of Cloud providers where the model is to be deployed provided by the Smart Oracle, b) fulfilled Korean regulations, for chips compliance, e.g. SGX provided by the Smart Oracle, c) permissions of the AI model owner – the physical person whos AI model is to be transferred.
9. After the access is granted by a mechanism implemented in the code of the Smart Contract, App service provider sends tokens for the usage of the AI model for the prespecified time given the price of the model.
10. Once the payment is completed, the Smart Contract locks the received tokens and triggers.
11. When selecting the target AI model, the application service provider has to pay the predetermined amount of tokens from his wallet for the usage of the model. For the open Smart Contract, he specifies the target model repository URI, the target AI model to use. The respective Smart Contract takes the parameters. Upon successful payment the Smart Contract returns an API key for the retrieval of the AI model.
12. The application manifest (in the form of an SLA) is handed to the infrastructure.

These are initially planned steps that the project partners are still refining during the implementation of the cross-border AI model (data) management scenario.

3.2 Architecture for AI Model (Data) Governance and Management

3.2.1 AI Model (Data) Management Architecture for DECENTER

The proposed architecture of the data management module of DECENTER Fog and Brokerage platform is presented in Figure 4. The module comprises five dedicated components, which are on the figure designated with blue colour: *AI Model Repository, Trusted Model Manager, Blockchain Service, Data Management Service* and *Smart Oracle*. They are explained in the following into more details.

Trusted Model Manager is a dedicated front-end service. It is the interface between the actor (i.e. Application Service provider/Developer) and the rest of the data management services. Its responsibility is to collect input from the user and to read and present the status of the executions of the Smart Contracts. The input data includes the URI of the target *AI Model Repository* and the query for a particular AI model. Moreover, the front-end is able to communicate with the user's digital wallet, meaning that the user is able to pay for the use of the model and also to pay the fee (i.e. gas) for the execution of the Smart Contracts. The reported status would communicate to the user whether the AI model has been successfully approved by regulators, and whether the consent has been collected from the people whos' privacy might be exposed.

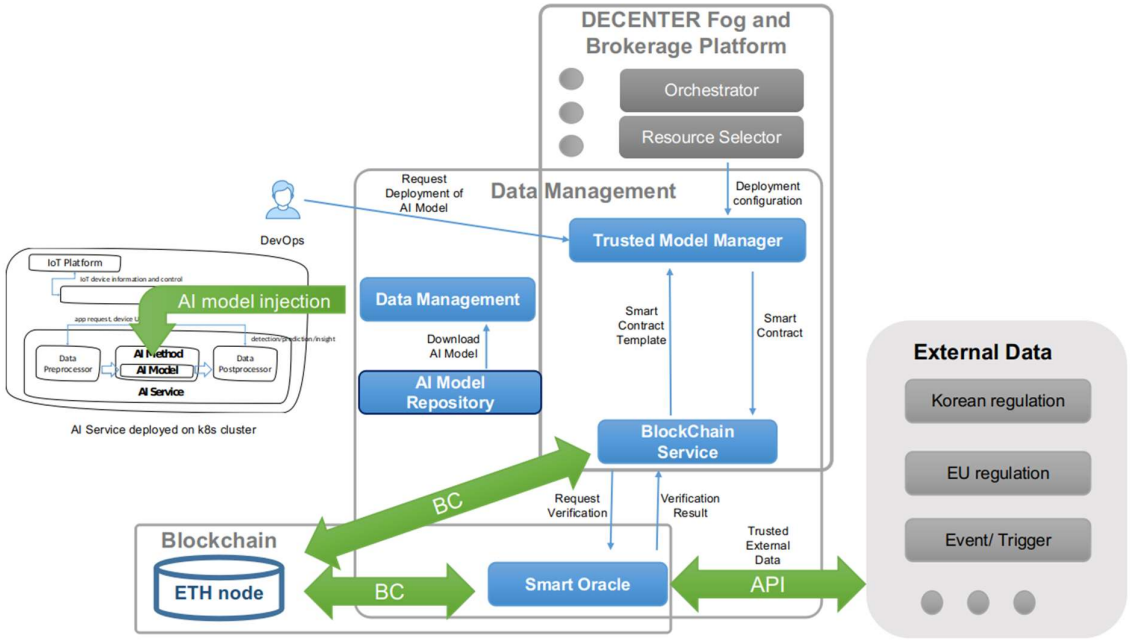

Figure 4. Data Management Architecture.

AI Model Repository is the off-chain storage, registry and API service that stores AI models and its associated metadata. Unlike the *Trusted Model Manager*, it is decoupled from the DECENTER Fog and Brokerage platform, similar to how the Docker Hub is independent from individual installations of Docker. Typically, many independent *AI Model Repositories* could coexist, each containing their own set of AI models. However, the DECENTER platform does not maintain a list of the repositories nor it provides any kind of discovery services to find them.

Therefore, it is the sole responsibility of the user to provide a URI of an *AI Model Repository* to the *Trusted Model Manager*.

Data Management is a reverse authentication proxy to the *AI Model Repository* with the aim to handle secure communication and to ensure authorised access to the AI models stored in the *AI Model Repository*. Like the repository, it is decoupled from DECENTER Fog and Brokerage platform, and unlike the repository, it is publicly exposed service. In fact, the URI of a repository points to this service. The authentication is handled via Smart Contracts, and it is able to intercept events emitted from the Smart Contracts that belong to the repository. The communication between *Data Management* service and the *AI Model Repository* is typically implemented over a private network, or, when not possible, over a secured channel.

The HTTP queries defined up so far for *AI Model Repository* are supposed to retrieve the requested AI model from the repository. In this case the Data Management service ensures that the access to the AI model has been authorised by the Smart Contract that corresponds to the repository. Possible other queries, which would result in metadata retrieval, could be simply passed through the *Data Management* service without any authentication.

The access to AI models is resolved individually per each model. In order for the access to be granted, several checks have to be performed. Firstly, AI model owners can set price for using their models for a specific amount of time. The access is revoked if the user who is requesting the AI model fails to pay the sufficient fee for using the model.

Secondly, an AI model which consumes input from a video camera is subject to potential disclosure of private data. In this regard, several regulations exist that prescribe the rules and terms under which private data can be processed, stored and moved. Moreover, the regulations typically vary among different geographical regions and special agreements might exist for cross-border exchange of private data. Thus, for a successful conformance with regulations it is necessary to know which countries the data flows through, and what type of data is processed and stored at each stage. For these reasons, it is required to know the deployment configuration. The deployment is regarded as conformant if all the regulators verify it. In addition, the regulators might enforce the use of trusted infrastructure providers who have attained adequate certification. For example, a regulator might require that the data is processed exclusively in secure enclaves, provided by specialised hardware such as Intel SGX. However, while essential, mere knowledge of deployment configuration might not be sufficient, because often regulations are concerned about collecting consent from persons whose private data is in question. In this case, regulators can only approve the use of an AI model once all the exposed persons have provided the consent.

Even though some of the concerns about data management have been discussed under the description of the *Data Management* service, most of the verification is in fact done on the Blockchain where Smart Oracles are used to collect off-chain data in a hopefully truthful manner.

Blockchain Service provides an interface between the Blockchain Ethereum network and the data management modules. Through this interface, Smart Contract templates can be obtained, then off the chain populated with concrete data, and finally, deployed.

Smart Oracles are trusted providers of external data into the user's Smart Contracts. In a failure-tolerant setup, they form their own network, such as Chainlink[8], and consist of the on-chain and the off-chain part. The on-chain part resides on the Blockchain and communicates with the user's Smart Contracts. In turn, the user's Smart Contract might request for some external data to Smart Oracles, more precisely, to the Oracle Smart Contract. The later would negotiate the Oracles to be used in the off-chain to harvest data based on their reputation and user's Service Level Agreement.

[8] https://chain.link/

In our data management scenario, Smart Oracles are used to ask for the conformance of the deployment configuration, the use of AI model and the data flows with regulations of all the relevant regions. They can also be asked to obtain end users' consent if a regulator requires so and if the list of the end users in question is known in advance.

Resource Selector is a component of the DECENTER Brokerage platform which is involved in the application deployment process. It looks upon the advertised resources on the Resource Exchange Broker and based on the results of matchmaking and algorithmic decision finds the "best" deployment option. For more details, the reader is referred to deliverable D3.2. In the context of data management, the *Resource Selector* is not a part of data management module, but is used to provide information to the data management module on the deployment regions of an application, its dataflow and topology.

Orchestrator is a collection of services, potentially spread over several hosts, with the joint task of coordinating a cluster (or clusters) of machines, providing the connectivity between them, deploying applications on them and maintaining applications' QoS. In the DECENTER platform, the *Orchestrator* is implemented with Kubernetes (see deliverable D3.2 for more details). The *Orchestrator* has the possibilities of live-migrating or re-instantiating particular pods, which may result in that the pod is moved to a different region. The change of the region has to be communicated to the data management module so that it can verify if the new deployment still complies with regulations. Similar to *Resource Selector*, *Orchestrator* is not a part of data management.

External Data represent a collection of external data providers, which are in our context denoted as regulators that prescribe the use, processing and storage of private data, and the exchange of data across borders. While regulators do exist in reality, to the best of our knowledge they do not offer (yet) an API to be used for automated verification purposes; therefore, for our needs, regulators will be implemented as simple standalone services.

Finally, on the left-hand side of the architectural figure an **AI Application Service** is depicted to which an AI model is injected. *AI Application Service*, as presented in D4.1, employs microservice architecture and is composed of three components: (1) *data preprocessor* ingests a video stream and dissects it into a sequence of images of suitable resolution (size) that an AI model can accept; (2) *AI Method* is a container serving a swappable AI model, obtained from the *AI Model Repository*; and (3) *data post-processor* consumes the results of the inference and performs some business logic.

Data management imposes two requirements for the application: (1) it has to contain at least one component that can perform AI inference from a loaded AI model, which is represented in the same format as used in the AI Model Repository (i.e. a container implementing the *AI Method*); and (2) it has to expose RESTful service to designate location of AI model to be used in it. The design of an application is the result of task T4.4 and has been described in deliverable D4.1. However, due to the advancements in the data management task (T4.3) ever since, the application design as reported in D4.1 is not yet fully compliant with these requirements, as it is only capable of downloading models but not receiving them via push.

The intended users of data management module are developers and/or operators, who are later in this document also referred to as *Application Service Providers*. These users are interested in the deployment of AI applications to serve their set of end users.

3.3 Sequence Diagram for Data Management

Following the description of the data management architecture, the sequence diagram depicted in Figure 5 demonstrates the process of a typical AI model management that is highly regulated, transparent and traceable. The traceability and transparency are achieved due to the fact that the data management leverages the Blockchain and Smart Contracts technologies, of which execution sequence is transparent and immutably stored onto the

Blockchain. The country regulation is enabled through the use of Smart Oracles which are designed to securely exchange data between the Blockchain and the databases of the respective countries (e.g. European Union, Slovenia, Italy, Korea).

The sequence diagram is composed of nine objects (i.e. *Application Service Provider*, *Trusted Model Manager*, *Resource Selector*, *Blockchain Service*, *AI Model Repository*, *Data Manager*, *AI Container*, *Smart Oracles* and *Ethereum PoW*), which depending on their role are grouped into four groups (i.e. Platform, Home, Remote and Blockchain) and explained below.

Services in the Platform group, i.e. *Trusted Model Manager*, *Resource Selector* and *Blockchain Service*, should be implemented as a part of DECENTER Fog and Brokerage platform. Data Management and *AI Model Repository* services are decoupled from the DECENTER platform and are therefore deployed on some site that is typically remote to the location of the platform. However, in the eyes of our scenario presented above, the *AI Model Repository* is located in Korea, which is Kim's homeland (thus grouped into the Home group). By analogy, the *AI Container*, which requests and consumes the AI Model, could be deployed in some remote location, and would therefore require cross-border data management.

The Ethereum network resides on a Blockchain and represents the place where user's Smart Contracts are deployed and executed. Related, *Smart Oracles* partially reside on chain and partially off chain. The on-chain part of *Smart Oracles* is represented by three associated *Smart Contracts*, each having a particular role, as follows. One Oracle's Smart Contract represents a bridge between the data stored into user's Smart Contract and the data coming from or going to the external world (i.e. off-chain data). Another Oracle's Smart Contract is responsible for negotiating a set of Oracles that can be trusted, based on a given Service Level Agreement. Finally, the third Oracle's Smart Contract aggregates off-chain data provided by the selected set of Oracles into a single answer. The off-chain part of *Smart Oracles* forms their own network of independent nodes, each capable of communication with off-chain services.

The remaining entity that is depicted in the sequence diagram and has not been discussed so far is *Application Service Provider*, a stakeholder/actor who triggers the deployment of an AI application by using DECENTER Fog and Brokerage platform, and following that initiates the sequence of AI model deployment.

The following steps occur in case of the successful retrieval of the AI model.

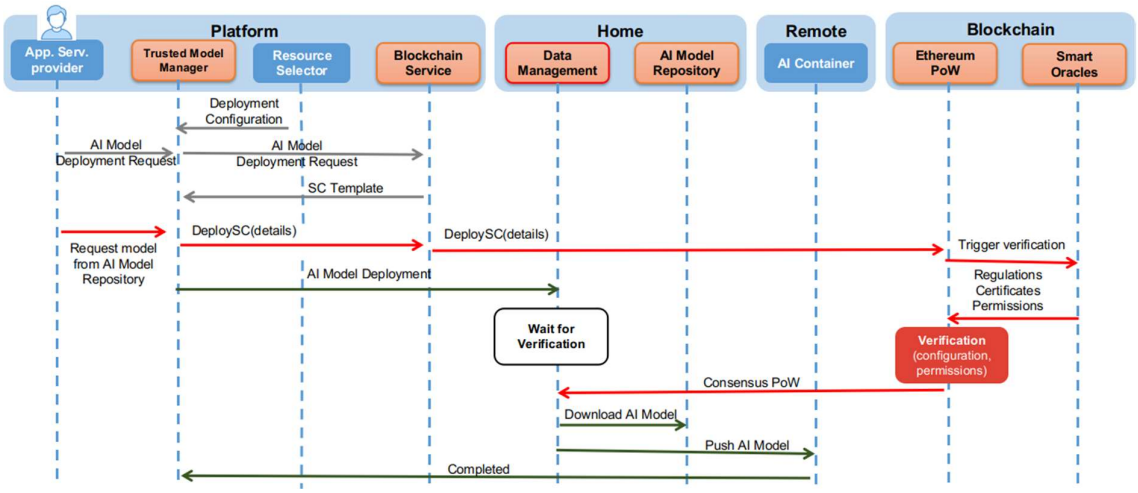

Figure 5. Sequence diagram presenting highly-regulated, transparent and traceable AI model management.

The assumed initial state is that the AI application is already deployed and running on the DECENTER Fog platform. We note here that before the deployment has occurred, the *Application Service Provider* had a chance to specify exact regions in which an application could/should be deployed, but is not mandatory. The *Resource Selector* can take that decision on its own, if not specified. However, for proper operation the application needs a specific AI model to run the inference. In our use case scenario, the model refers to the biometric model of a Korean visitor at the construction site in Ljubljana and is used to recognise his face for the identification/authorisation purposes. After the AI application has been deployed, the *Resource Selector* knows about exact regions in which the application is deployed and is designated as Deployment Configuration. In particular, the Deployment Configuration would know the regions of video cameras (or IoT devices) and of all the application components. This information is communicated to the *Trusted Model Manager*. This is considered a precondition state.

Application Service Provider then initiates the sequence of retrieving the specific AI model into the *AI Service* container. She does so by specifying the AI model deployment request, which contains a query that an *AI Model Repository* understands and by which it can uniquely determine the target AI model, and the URI of the target *AI Model Repository*. In our use case scenario, she knows that the target model is stored in the specific repository in Korea. The request is communicated to the *Trusted Model Manager* which in turn forwards the request to the *Blockchain Service*.

The *Blockchain Service* retrieves a Smart Contract Template for the given request and returns it to the Trusted Model Manager. The template contains the ETH address of the AI model owner in request, a set of functions and variables required for accessing the AI model of the particular *AI Model Repository*, and a set of uninitialised variables that can be provided through links to the Smart Oracles for collecting approval for using AI model from regulators and obtaining consent from end users (in our scenario the end user is Kim).

The content of the Smart Contract Template is presented to the *Application Service Provider*, who, by paying the requested price and fees agrees to the Smart Contract. The payment is done through the *Trusted Model Manager* front-end which in turn relies on the web browser

add-on, such as MetaMask. The *Trusted Model Manager* deploys the Smart Contract on the *Ethereum PoW Blockchain* through the *Blockchain Service*.

Upon successful deployment of the Smart Contract on the *Ethereum Blockchain*, the *Trusted Model Manager* sends a request for deployment of the particular AI model to the *Data Management* service that accompanies each of the *AI Model Repositories*. The *Data Management* service then waits for the verification that should arrive as emitted events triggered by the Smart Contract based on the external data collected by Smart Oracles once the regulators and end users approve the use of the AI model.

If and when the regulators and end users reach consensus, the Smart Oracles securely provide that external data to the deployed Smart Contract which then emits an event that is handled by the *Data Management* service of the respective *AI Model Repository*.

Finally, upon receiving the event, the *Data Management* service downloads the AI model from the *AI Model Repository*, and, once downloaded, pushes the model into the *AI Service* container.

4 AI Model Repository Design and Implementation

4.1 AI model repository

4.1.1 Purpose of AI model repository

In DECENTER, we are focusing on delivering AI from the cloud to the edge in efficient ways, instead of building AI for the edge from the scratch. To this end, it is required to handle the resources on the edge in efficient way as well. Unlike the resources on the cloud, where it is considered to have infinite resources, the resourced on the edge are limited according to the hardware they are relying to. Suppose that we have two edge devices for AI, one with the Nvidia 1080ti GPU and the other one with Nvidia Tegra GPU. Even though we have a good AI model with higher accuracy, it is hardly expected to be run on the second edge due to the hardware performance restrictions. To handle those different resources on the edge, a proper model according to the resource capabilities needs to be deployed. To provide AI models efficiently, DECENTER took approach with separating AI model delivery from the AI microservice container. Instead of storing AI model in a container for deployment, a container without AI model will be deployed onto resources, and AI model will be deployed on the runtime afterward. This separation of AI model delivery and container delivery has a few benefits on resource utilization. First, it will increase re-usability of an AI container. Instead of building a new container with different AI models and the same application logic for various edge resources, this method will enable a single AI container can be used on edges with different resources. Second, the network resources for the microservice will be optimized since only the container and specified model will be delivered to the edge.

In DECENTER, the inference will take place on either edge or the cloud, and the AI models can be used for inference at the edge side. Inference requires much less computing power than a training model in terms of processing. The computational complexity of the inference is just a set of the matrix multiplication operation. The advantages of inference at the edge side are as follows.; The model does not require Internet access. And the service latency can be reduced by handling data directly to the model at the edge. Therefore, nowadays, the edge is trying to get the AI model trained in the cloud and infer directly. This chapter describes the new AI model repository in this project. The proposed server takes into account that the containerized application to be deployed on the edge does not include a static main AI model. Instead of this way, the edge device can import the necessary model directly from the AI model repository. The edge can import static AI model, but also can find and import the appropriate AI model based on the real-time needs by proposed. In addition, this server is able to handle partitioned AI model, which has been investigated in T4.1 and described in D4.1.

4.1.2 AI model repository structure

In this chapter, the structure of AI model repository is presented. Figure 6 depicts how the application and AI model can be deployed to the edge devices. The figure on the left shows the way to deploy an application containing the AI model directly to the edge. The figure on the right shows the proposed method deploying the application without the main AI model on the edge. In this way, the edge is able to find the proper AI model from the AI model repository considering its computing resources.

The AI model repository acts as a pipeline to provide intelligence on individual edges. If edge does not know exactly which model to download, the repository server can search based on the requirements of the model and find the appropriate model.

As mentioned earlier, in DECENTER project, the trained AI model is used alone, but two or more split models are also used on the edge. N divided models are distributed to N edges respectively, and the output of one split model is used as the input of the other model. Therefore, above-mentioned characteristics should be reflected in model repository design. Related metadata should be also managed well so that lot of models can be easily reused and utilized. Therefore, DECENTER model repository server is designed to manage AI models and related information efficiently.

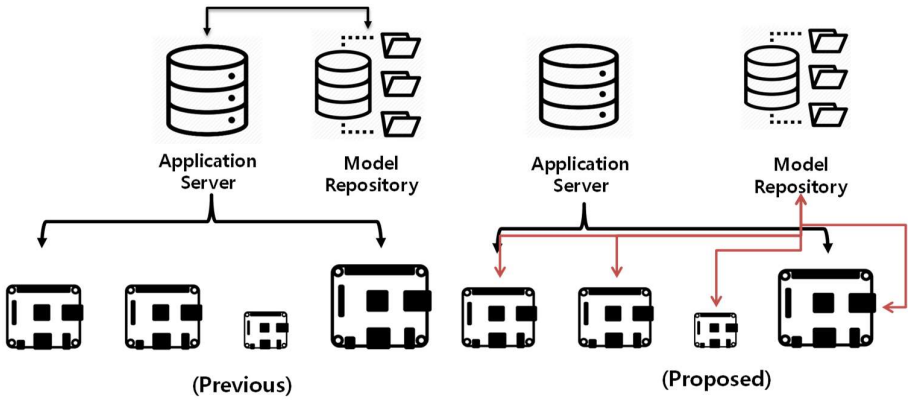

Figure 6. Two different ways to use the AI model on the edge side.

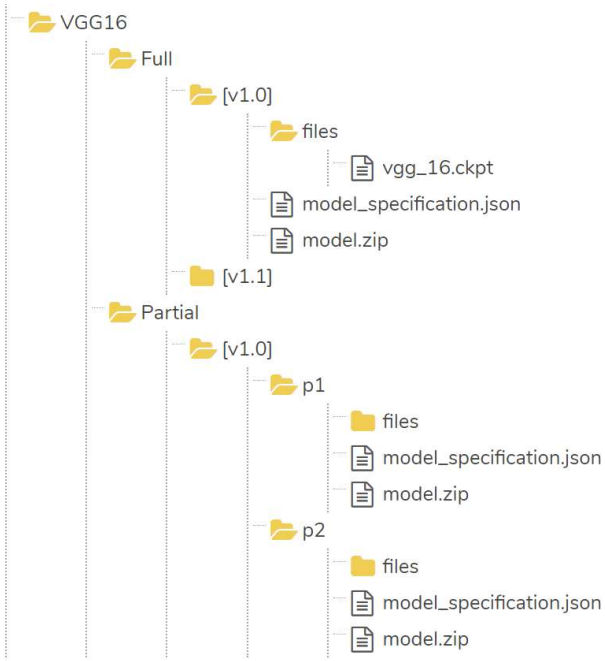

Figure 7. The folder structure of the model repository.

We applied a file system structure providing an interface to consistently find the proper AI model for the first version of the AI model repository. Developers can register their model and the system can manage a variety of models based on the pre-defined structure. As we mentioned, the prominent part of the advanced AI application is the flexibility caused by the

model splitting. The AI application with partial AI model can be in conjunction with other edge's corresponding application. Therefore, this repository is designed to properly save and deliverer the partitioned AI model to edge. Edge can access the repository server which has a structure by combining information of the model name, version, whether it is partitioned or not, and the number of the partition.

Figure 7 shows the stored structure of several variants of the modified VGG16 model. Each structure is divided into two parts: Full and Partial. Full is the directory where common whole models are stored. Partial is the directory to store the models which are split into more than two. Each model is managed by version. The subfolder contains the JSON file describing the metadata for the stored model and the file to be downloaded from the edge. Based on this structure, the edge can access and download the proper model from the model repository server. The subdirectory named partial stores the data of the partitioned models. After a model is divided into a number of sub-models for collaboration between devices, this subdirectory is used for storage of split models. Partial directory subdivisions are grouped by version. The subdirectories contain the order of split models.

4.1.3 Semantics of the main elements

This repository provides the model specification file named "model_specification.json" associated to a specific AI model. It can be easily changed with any document-oriented database such as MongoDB. This specification file describes basic information about AI models such as model information, partial information, and train information. Model_specification.json file is created based on the pre-defined schema structure. In this stage, we registered and tested existing training models. We will then register customized models by individual use cases and make them available for use.

In the current stage, we registered and tested existing training models. We will then register customized models by individual use cases and make them available for use.

Model data type

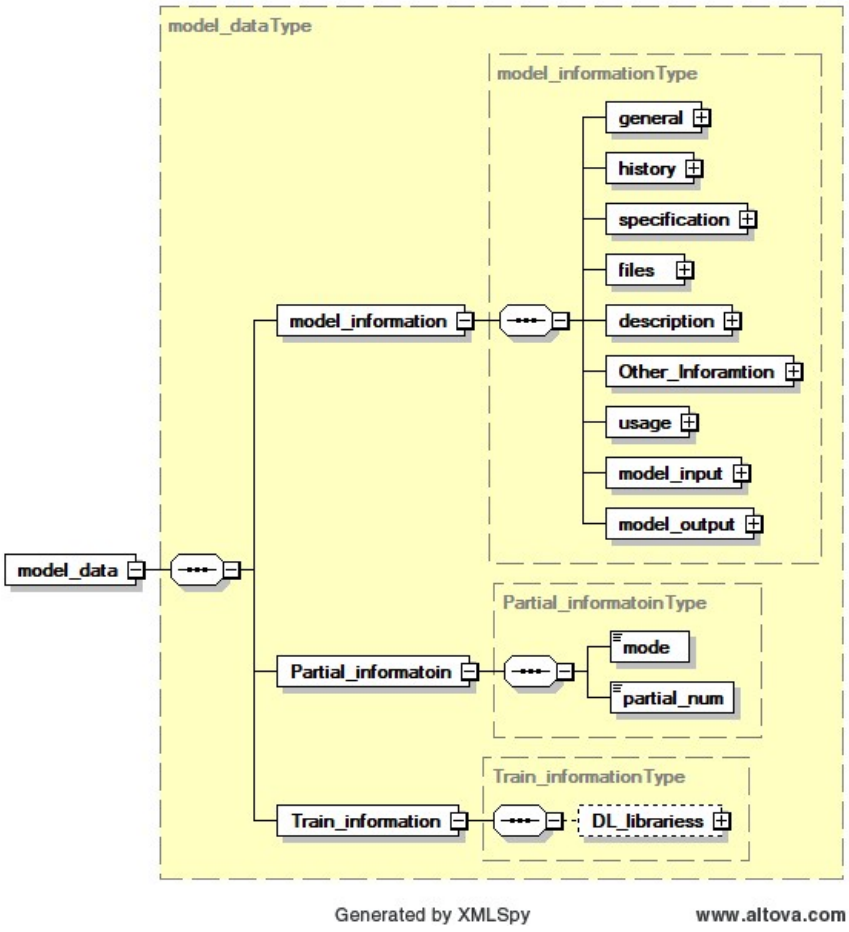

Figure 8. The basic structure of the AI model data type.

The DECENTER AI model repository provides schema to describe the AI models. The serialization methods described above does not have explicit ways to include metadata for an AI model for the identification, and usually file names are used for identifying a model, which might be very inaccurate. In order to describe the AI model, each model should have metadata information. For example, the edge device is able to select an appropriate AI model based on the model cost, the model performance, the resources required by the model execution, and the ability of the privacy protection. The basic structure of metadata is defined as follows and can be extended in the future.

The AI model can be described using one model element having three main elements: model information, partial information, train information. The model information describes basic model information, and partial information describes the partition information of the model. Finally, train information describes the information used during model creation.

Figure 8 shows the structure of model data element, which is the root element of the schema.

- Model information has 9 sub-elements: general, history, specification, files, description, other information, usage, model input, and model output. It contains basic information for searching.

- Partial information has 2 elements: mode, partial num.
- Train information has only one element: DL libraries.

Model information element

Figure 9 shows the overall structure of model information type. Files element, description element and usage element can have multiple sub-elements which describe the detail information.

The files element describes all the files needed to run the model. The files element includes related file names. The server provides archive file which is composed of one or more model related files which are described in the files element. The description element describes additional information, such as basic information about the model provided or information to be used for the search. The usage element describes how this model can be used. An AI model can be used for various purposes such as classification, forecasting, and recommendation.

Complex elements other than the elements described above defines a semantic below.

Semantics of the generalType is:

Name	*Definition*
id	This element describes the unique id to identify the AI model.
license	This element describes the rights associated with the model.
manufacturer	This element describes the manufacturer making the model
Name	This element describes the name of the AI model

Semantics of the history Type is:

Name	*Definition*
date final available	This element describes the date the model was last registered.
historys	This element describes the relevant version information for the model. If there are multiple versions, they contain information about all versions.
this version	This element represents the specific version number of this model.

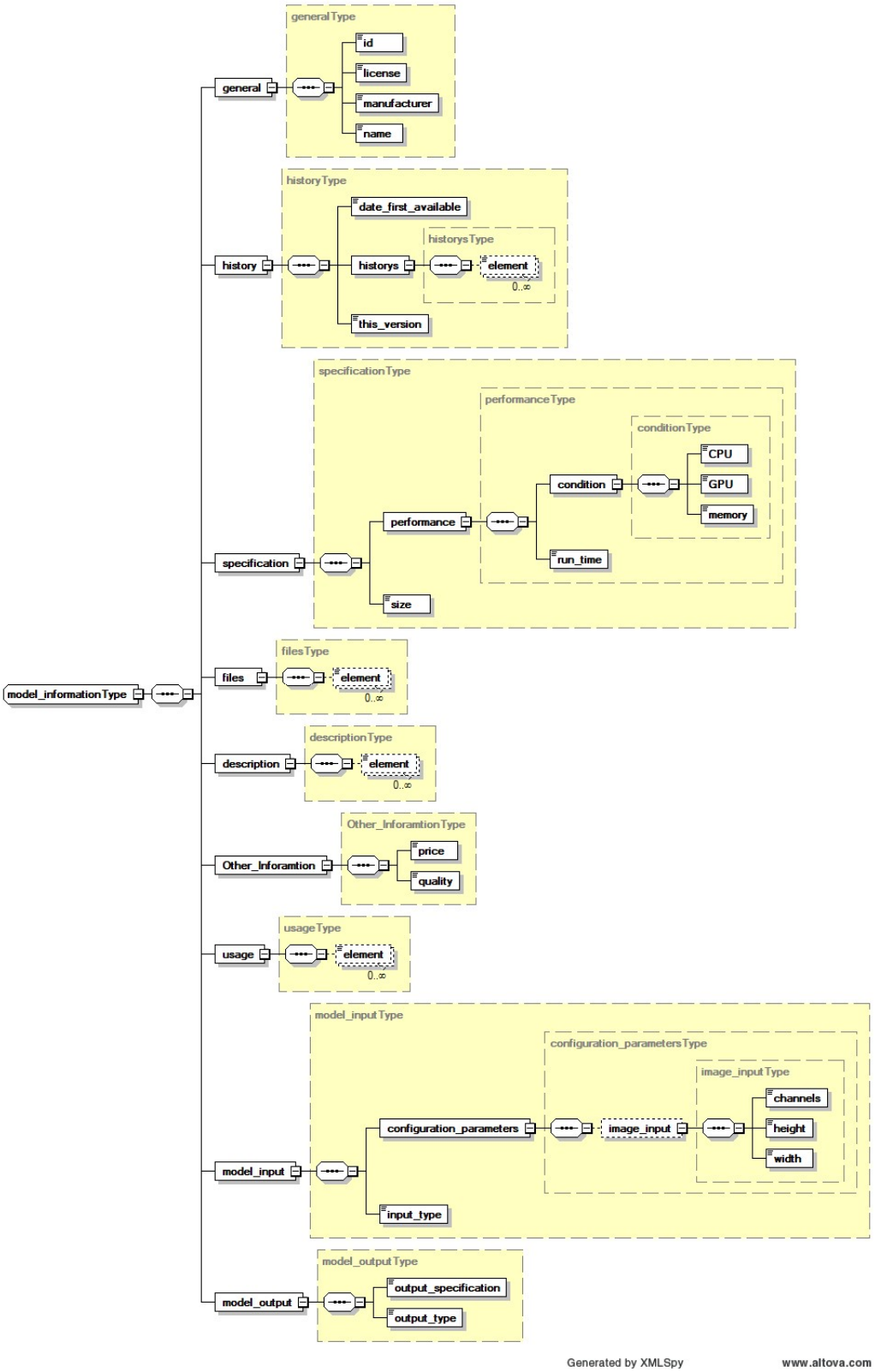

Figure 9. Overall structure of model_information type.

Semantics of the specification Type is:

Name	*Definition*
performance	This element describes the predictable performance when this model is used. It is not easy to test performance on all combinations of hardware during model operation. Therefore, it describes the minimum required hardware, and the performance when the model is running on this combination described.
size	This element represents the size of the model. When deploying and saving the model on the edge hardware, the import prerequisite is the size of free memory space which is bigger than the model size.

Semantics of the other Information Type is:

Name	*Definition*
configuration parameters	This element describes the specific conditions for the input. At this stage, it describes specific conditions for the image. We need to extend the element scope.
Input type	This element represents the input type. It can be currently the image, audio, data, and sensors as an input type.

Semantics of the mode output Type is:

Name	*Definition*
output specification	This element describes the specific information of output.
output type	This element describes the type of output.

4.2 Implementation result

4.2.1 Implementation environment

A preliminary AI model repository has been implemented within the period. The repository is developed with the Django Web framework on the VM instance on the cloud. It is able to connect to this storage by general web interface and download the proper model. DECENTER microservices will be able to easily access models from the model repository using REST API for accessing models that are suitable for the microservices. Table 1 shows the detailed information of registered three models that are registered in the model repository. VGG16, YOLO v3, and customised AI model for indoor status prediction were used as candidate AI models.

Table 1. AI models used to deploy in DECENTER for the first year of the project.

Name	**Description**
VGG16	• VGG16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper “Very Deep Convolutional Networks for Large-Scale Image Recognition”. • This model achieves 92.7% top-5 test accuracy in ImageNet (A dataset of over 14 million images belonging to 1000 classes).
YOLO v3	• You only look once (YOLO) is a state-of-the-art, real-time object detection system and one of the faster object detection algorithms. • YOLO v3 uses a few tricks to improve training and increase performance, including multi-scale predictions, a better backbone classifier, and more.
Indoor PM10 prediction model	• This model is a prediction model generated by KETI for UC 4 • It can predict the future fine dust concentration based on the pattern of indoor air. It uses indoor and outdoor air data as input for a specific time period.

4.2.2 REST API Interfaces for AI model access

A microservice can download the model using the REST API. In DECENTER project, we propose a simple method to download the AI model using four parameters. The API is generated based on the model information: name, version, whether it is split, and the order of the model. This is an example of downloading the model. The base URL is the combination of the main host URL and “model”. The full URL address passes the 4 input parameters as query parameters as follows.

BASE URL

- base url is http://[host_url]/model
- Ex) http://0.0.0.0:5000/model

EXAMPLE ADDRESS

- http://[host_url]/model?model_name=VGG16&model_version=[v1.0]&partial_mode=Partial&partial_num=1
- http://[host_url]/model?model_name=Yolo_v3&model_version=[v1.0]&partial_mode=Full&partial_num=0

Path Parameters

Name	Required/ Optional	Type	Description	Example
Model_name	Required	String	This server provides multiple AI models which can classify, predict the future status. First of all, micro-service needs to know the right model name to download it. Each microservice can download and manage the model based on the described information.	-model_name=Yolo_v3
Model_version	Optional	String	The AI model repository server manages the version of the AI models. The server must maintain a history of changes in the model because the AI model might be modified or improved. The microservice can download and use the appropriate version of the model.	-model_version=[1.0]

Partial_mode	Required	String	In order to have one independent model on each microservice, the repository server should be able to provide a single AI model or a partitioned AI model. In this project, there are two types of model (Full/Partial) so that the microservice can cooperate with other devices.	-partial_mode=Full
Partial_number	Optional	Integer	Partial Number is required to download the specific split model for the micro-service. In the case of Full mode model, its partial number is always 0. For split models, the order number can be set from 1 to the number of split models.	-partial_number=1

4.3 Implementation results

As of now, we tested TensorFlow checkpoint files and pickle files as serialized model files. The checkpoint file has meta-information about the model, so it is easy to separate the model itself. We need to modify the original model to split it. Therefore, the checkpoint file is the best option to save and modify it. However, we will have to use the saved file format without metadata in the future because the checkpoint file having a lot of information is relatively large. We will test some other serialization formats. To validate the AI model repository, we registered several pre-trained models and tested the deployment process. Micro-service is able to connect to this storage and download the proper model remotely to derive an AI application. DECENTER application can use this API to download the model easily from the Model Repository. This repository server provides a simple API to access and download the right model for a specific microservice. This API address is designed to have four parameters to identify a model of interest: model_name, model_version, partial_mode, and partial_number.

We have created an AI model repository server that can provide real models. We have also developed a website that allows users to easily manage the registered model. Figure 8 shows the data distribution of input data type of the registered model. We registered five test models: three models for image classification and two models for analysing time-series data in Python Panda's DataFrame data type.

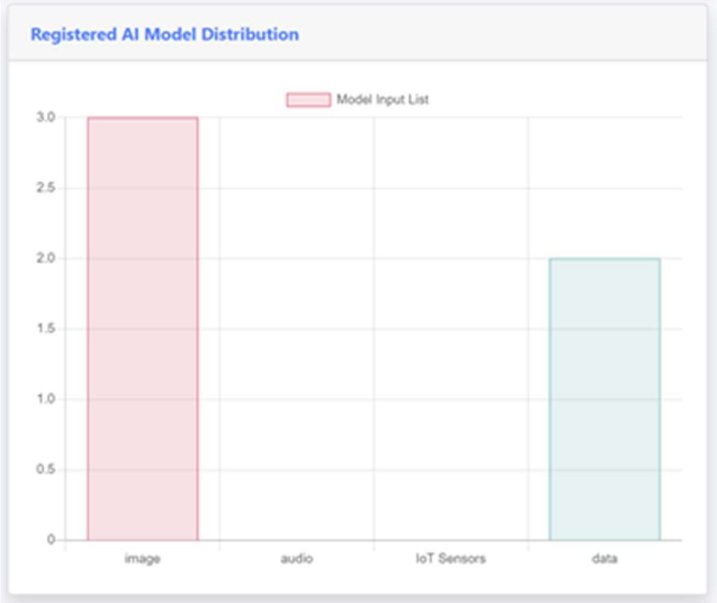

Figure 10. Registered AI Model Distribution.

The repository server provides summary information about the registered model. We can see the name, version, updated date, purpose of use, model input and output, library used for training, and model partition information. Various information can be shown on the website (Figure 6).

id	name	version	update date	usage	input	output	libraries	partial mode	partial number
00001	LR_salary	v1.0	2019-05-26	object detection	data	float	scikit-learn,tensorflow,keras	Full	0
00003	VGG16	v1.0	2019-04-26	object detection	image	integer	scikit-learn,tensorflow	Full	0
00004	PM10_1hour	v1.0	2019-05-16	status prediction	data	float	scikit-learn,tensorflow,keras	Full	0
00005	VGG16	v1.0	2019-04-26	object detection	image	float list	scikit-learn,tensorflow	Partial	1
00006	Yolo_v3	v1.0	2019-04-26	object detection,person detection	image	integer	scikit-learn,tensorflow	Full	0

Figure 11. The summary information of registered AI models.

5 Trust Management for AI Based Fog Computing Applications

This is an exploratory study that intends to analyse the various aspects of cross-border data management. Fog computing per se represents cross-border "everything". The resources (e.g. AI models, Cloud providers), services (e.g. Cloud storage services) and the new AI-based DECENTER service will be operated across multiple administrative domains and countries, which all have their own policies and regulations. While trust, as we already defined it, is a binary concept, our approach towards data management in the fog should be soft in the sense that one has to always balance between the various trade-offs. For example, if one AI model that has to be transferred from Korea to Europe contains sensitive information, in such case more regulations and policies would apply, in another occasion, perhaps only the time of the transfer of the model (as fast as possible) would represent the trust-issue. We have therefore made an initial study making efforts to understand the important aspects of cross-border data management. We therefore first start with various identified attributes to trust. Then we proceed with elaborating several scenarios, and further to that, we also present a trust-management architecture which is based on Smart Contracts implemented by using the Solidity language.

5.1 Definition of trust and trust attributes

The concept of trust is complex similarly to many aspects of human endeavour [21]. A suitable definition of trust was presented by Gambetta [22]: trust (or, symmetrically, distrust) is a particular level of the subjective probability with which an agent assesses that another agent or group of agents will perform a particular action, both before he can monitor such action (or independently of his capacity ever to be able to monitor it) and in a context in which it affects his own action. In computer science, there have been efforts to formalise trust, for example, to facilitate cooperation among autonomous agents [23]. The formalisation of trust, therefore, aims at improving the possibilities for trust management in dynamic and distributed computing environments. In this context, Viljanen presents various considerations towards the definition of an ontology of trust [24].

The goal of the present work is, therefore, to analyse key aspects and attributes to trust important for smart applications and environments, and to design and implement a new trust management system that can be used in various dynamic Edge-to-Cloud computing scenarios. Our practical goal is to also integrate this innovative architecture with the advanced DECENTER Fog Computing Platform ***and address its cross-border data management requirements.*** The design and implementation of Smart Oracles and their use in the context of achieving trust is currently a hot research topic [25]. The use of Smart Oracles reduces the necessity of costly transactions on the blockchain and represents a mechanism for balancing the use of on-blockchain and off-blockchain data in the operation of SCs. Ethereum's blockchain, SCs and specially designed trustless Smart Oracles represent the basis of the new trust management architecture presented in this study.
Following are the considerations of this study:

- trust attributes analyses and identification in order to address the requirements of dynamic, complex and multi-tier smart applications and environments,
- a blockchain-based trust management system applicable to multi-party decentralised Edge-to-Cloud computing,
- especially designed Smart Contracts and trustless Smart Oracles, which jointly support trustful autonomous transactions among the parties while reducing the transactions' costs, and

- implemented proof-of-concept trust management scenarios for the DECENTER Fog Computing Platform involving the registration of participants (users and providers) for transparency and traceability, access to decentralised resources including cameras (video streams), Fog and Cloud providers, and traceability of the data flow among the resources (camera, Fog, Cloud) with regard to security and privacy preservation requirements.

Here, our work focused on trust as a fundamental factor in achieving dependable interaction between entities, stakeholders and services in a decentralised environment underpinned by dynamic IoTs. Several recent studies have presented trust-related issues and proposed various trust modelling approaches and attributes. Alrawais et al. [29] investigate the requirements for trust management and find that a trusted system has to maintain reliable service, prevent incidental failures and handle misbehaviour issues. Bimrah et al. [30] describe reputation, security, risk, initial trust and privacy as the essential concepts that define trust. Furthermore, Carradini et al. [31] extend the understanding of trust with additional related concepts including dependability, security and reliability. Some trust-related concepts include the ability of the system to respond by satisfying specific QoS attributes, such as response time shorter than 50ms or throughput higher than 4 Mbps. Thus, trust can also be associated with the system's performance and quality attributes, similarly to various policies and regulations that should be applied to data management. Accordingly, a Fog computing platform, such as DECENTER will be trusted, if it is capable of satisfying application specific QoS requirements.

In other words, trust management requirements depend on the specific use cases where trust management must be applied and are multifaceted. Figure 12 presents several relevant trust related attributes. Some include:

Availability is a fundamental attribute of Fog nodes that evaluates the probability of the node's correct functioning at a specific moment in time.

Credibility defines the degree to which the data source or the data is seen to be believable, a concept that can be extended to any data item, such as a video frame or an AI model based on TensorFlow.

Privacy in the context of the IoT includes the following relevant aspects: awareness of security risks imposed by smart devices surrounding a human subject, individual control over the data collection and processing of personal data, and awareness and control of subsequent use and dissemination of personal information by those entities to any entity outside the subject's personal control sphere [32].

Response time is an attribute that represents the time necessary for data to be processed by a selected Fog node and data packets to be transferred to the client. For simplicity, in the present implementation, it is measured as the round trip time for the package to reach a microservice, perform necessary processing, and return extracted metadata to the client.

Throughput is an attribute that represents the rate at which data is transferred between endpoints (e.g. between a sensor and a Fog node). In other words, this attribute shows the amount of data that a specific component can successfully transfer per unit of time.

Security estimates the ability to protect the system from accidental or intentional external attacks. This attribute to trust is closely related to confidentiality that evaluates if the data within the distributed environment is protected from disclosure to unauthorised entities. For instance, Hu et al. [33] analyses and summarises the security and privacy issues in face identification platforms that are based on Fog computing in order to provide a trusted service.

Transparency is an attribute that allows the blockchain ledger to be fully auditable. That allows everyone participating in the ecosystem to view the stored transactions on the blockchain.

Traceability is an attribute that is closely related to transparency. Traceability allows to trace back the interaction between entities on the blockchain since all history can be traced back to the first transaction.

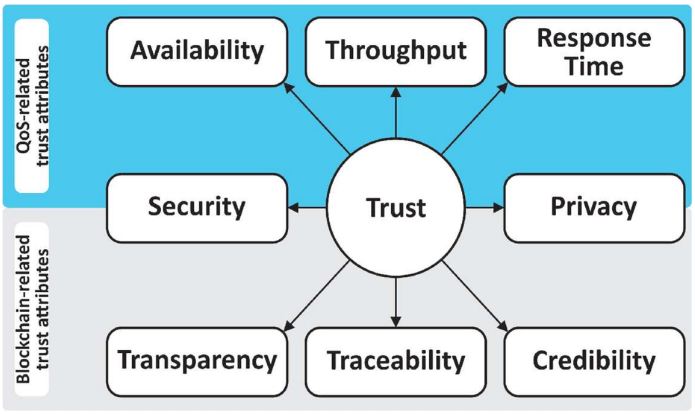

Figure 12. Initial list of trust attributes considered for implementation.

Our intention is to extend this list of attributes in the next phase of the project in order to address actual cross-border AI model and data management scenarios, as discussed in the previous sections.

5.2 Trusted video stream access scenario

The scenario of trusted video stream access starts by the user sending a request to the blockchain service through the Web GUI. The blockchain service deploys the instance of the SC with basic price estimation and triggers the selection of the camera through a specific Smart Oracle dedicated API. This process ensures that the payment is made only to a trusted camera service. In the next phase the prices are additionally set up by the Service Pool and Camera Pool components. Each execution of this function is completed with Ethereum event trigger that notifies the listening services. After a successful price determination and trust management decision the consensus is reached and the certified user is asked to pay the specified price in order to start the camera session. Besides, the camera deployment service is triggered in this function through a Smart Oracle dedicated API that deploys the actual container image instance service. Similar to this scenario are the scenarios for assuring trust for Fog nodes and Cloud storages.

5.3 Trusted data flow scenario

Similar to the Trusted video stream access scenario is the Trusted data flow scenario. The only difference of the implementation is in the different Smart Oracle APIs endpoints, which are now related to the Infrastructure Pool services.

Because the operational cost is an important part of this decentralised system, the trust management scenarios may follow fixed or pay-as-you-go pricing methodology. The users may not always know in advance the intended usage period of the smart application; they may be willing to pay a full price which is locked into a Smart Contract. The user's session ends

automatically when the duration determined in the Smart Contract is exceeded or on demand by the user, where ETH funds reimbursement to the user has to be calculated. In both cases, undeployment of the user's session is triggered through a dedicated Smart Oracle API. This process can only be triggered manually by the user or automatically when the exceeded time condition is reached. The undeployment process terminates the user's microservice, deletes the footprint, and releases the allocated resources. Finally, the user receives a notification about the successful completion of the video streaming and AI processing session.

Transparency and traceability of the data flow among the entities which is achieved through the same Smart Contract represents a trust building measure. However, the price determination and off-chain data sources are different. This scenario determines the prices from the Infrastructure Pool.

Trust in the Cloud storage can be significantly enhanced by improving the security and privacy of the stored data. For example, security and privacy in such scenarios can be improved by exploiting high distribution rate of cloud repositories and by splitting the data among them. This is the case of Storj[10], Sia[11] and other approaches that use fragmentation and distribution of data.

5.4 Architecture for trust management

Having presented the background, emerging smart video streaming applications and their requirements for trust management, we introduce a trust management system designed for DECENTER's Fog Computing Platform. Figure 3 depicts a multi-level architecture of the trust management system. Each level in the architecture is used at a different stage of the video stream processing Big Data pipeline and is described in the following.

Application Layer is the entry point for the user that wants to manage trusted cameras, Fog nodes or Cloud storage repositories. This layer is connected to the Ethereum ecosystem through the Ethereum bridge Metamask[12], which is a browser add-on.

Blockchain Layer uses the Ethereum (ETH) ledger as pillar blockchain environment that enables SCs. The two main components are SC templates and Smart Oracles. The deployment of the SCs occurs on demand through the blockchain service which is owned by the system. Each SC instance communicates with external services (e.g. Fog nodes, QoS monitoring system) through registered APIs that are part of external services. This approach results in enhanced integrity of the functions that verify the correctness of the API queries by using unique API keys and thus avoid calls from potential malicious SCs. The design of the SCs followed the oracle pattern proposed by Wöhrer et al [52] and best practices presented in the framework OpenZeppelin[13].

Edge-to-Cloud Orchestration Layer is composed of infrastructures (Cloud storage repositories, Fog and Edge nodes) for the deployment of containerised microservices and data, QoS monitoring of the infrastructures and IoT devices (cameras). The Cloud storage repositories, Fog and Edge nodes are the infrastructures that are available at runtime. Each of these infrastructures plays a different role in the video stream processing pipeline. Edge nodes are responsible for rapid data acquisition and data normalisation; Fog nodes are

[10] https://storj.io/
[11] https://sia.tech/
[12] https://metamask.io/.
[13] https://openzeppelin.org/.

responsible for data processing, compression and transformation and Cloud storage repositories store the data upon request. The orchestration capability is realised by using Kubernetes and is thoroughly described elsewhere [26]. Before using any of these infrastructures in the other layers of the architecture, they are first verified on the blockchain if they satisfy trust including QoS requirements. This layer continuously monitors the infrastructures and gathers QoS related information. The monitoring data is available to the blockchain layer through the blockchain Smart Oracles. Besides the layer consists of all available IoT cameras that the user can verify through the blockchain and use their video streams.

Decision-Making Layer is responsible for determining an optimal infrastructure for the deployment of containerised microservices (e.g. TensorFlow). This layer uses a Markov probabilistic decision-making method for automated decision-making [53]. In order to rank the infrastructures, the Markov method requires QoS monitoring data and QoS threshold values from a specially designed Smart Oracle. These metrics are necessary to generate a probabilistic finite automaton, which is later used to produce infrastructure ranking list. The first ranked infrastructure of the ranking list is considered as an optimal deployment option and returned to the certified user (i.e. to the trusted user) as a trusted infrastructure that satisfies the user's QoS requirements.

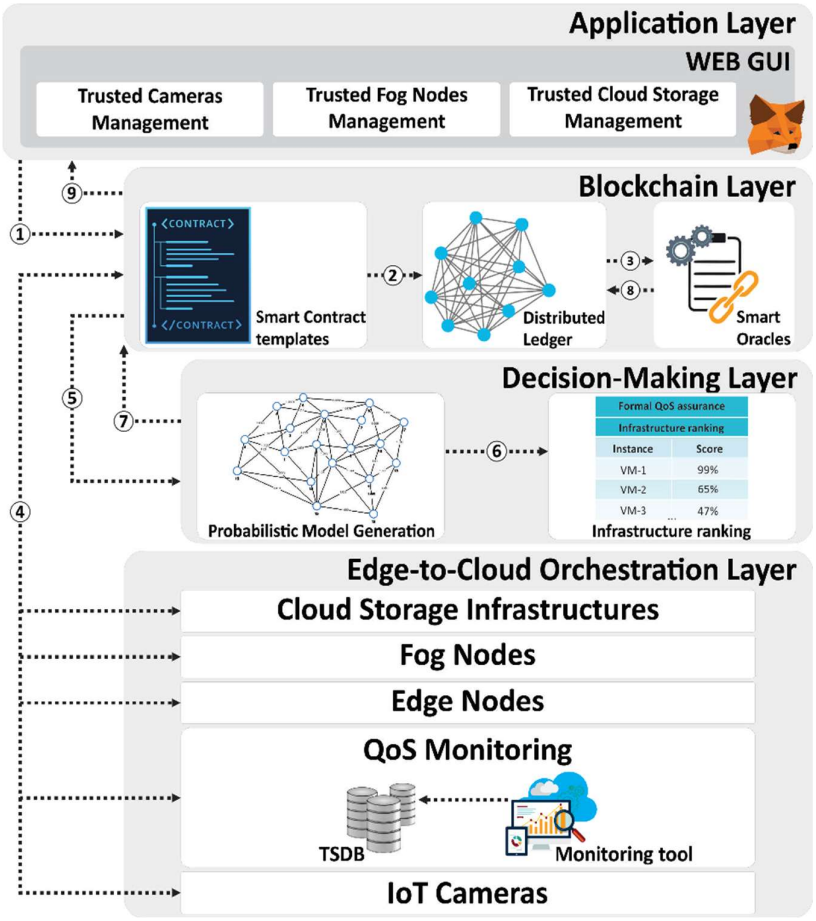

Figure 13. High-level architecture of the trust management system.

The operation of the above described trust management system is explained in the following section.

5.5 Proof-of-concept trust management scenarios

Our new trust management system was implemented in order to support smart multi-tier i.e. Edge-to-Cloud applications. The main purpose of our container-based video surveillance smart application is to prevent safety violations at a construction site and to timely predict and prevent injuries. The smart application subscribes to video streams from trusted cameras, processes them by using TensorFlow at Fog nodes and records marked video frames on a trusted Cloud storage for later use.

In order to ensure trusted operation, it is necessary that each person as well as hardware and software entity in the distributed environment is trusted. Moreover, it is also necessary that the data flow among the application components is also trusted. Here, the smart environment must precisely distinguish trusted surveillance cameras, Fog nodes and Cloud repositories from non-trusted infrastructures in the open environment.

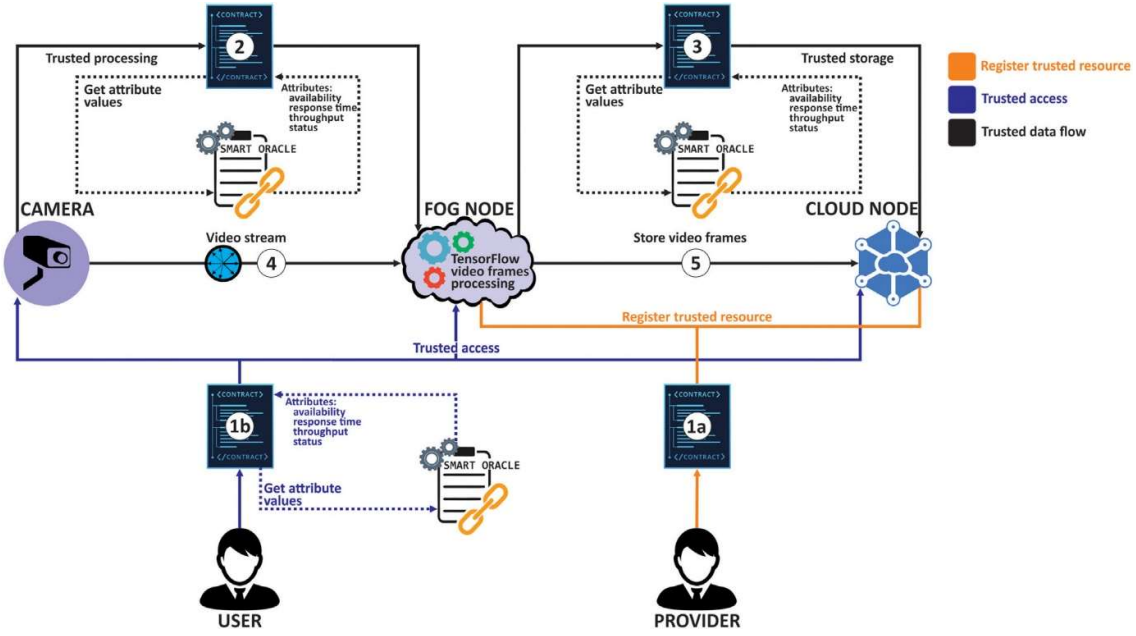

Figure 14. Using Smart Contracts in the operation of a video stream analysis application.

The figure above presents the investigated proof-of-concept trust management scenarios. A provider authenticated by using a blockchain wallet registers a Fog node or a Cloud storage on the blockchain (1a). Then, another user also authenticated by using a blockchain wallet can subscribe to a trusted camera video stream or use a Fog node for AI processing or Cloud storage for storing sensitive data only by executing a SC that takes into account QoS and trust related requirements (1b). Before the deployment of the smart application, an SC containing the previously described Markov method executes and selects a Fog node that satisfies the QoS requirements of the smart AI application (2). This part of the process also verifies that both entities have their valid blockchain wallets. For the Fog node to be able to forward the processing output or specific video frames to a Cloud storage node, it is necessary to execute another SC, which selects a Cloud storage node satisfying the QoS requirements based on a

Markov model, and again verifies that both entities are connected to valid blockchain wallets (3). Once trusted connectivity is assured, the video starts streaming from the camera to the Fog node for AI processing (4). The processing output is stored according to the application's needs on trusted Cloud storage (5).

The trust in the Cloud storage can be significantly enhanced by improving the security and privacy of the stored data. For example, security and privacy in such scenarios can be improved by exploiting high distribution rate of cloud repositories and by splitting the data among them. This is the case of Storj[16], Sia[17] and other approaches that use fragmentation and distribution of data.

Figure 15 is a convenient representation of the actions performed in the trust management scenarios and Figure 16 presents the data types used in the scenario.

Figure 15. Sequence diagram for the trust management scenarios.

[16] https://storj.io/.
[17] https://sia.tech/.

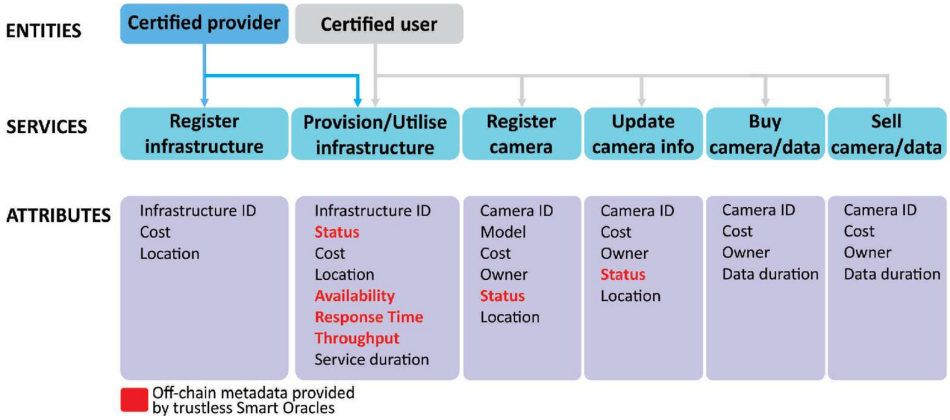

Figure 16. Smart Contracts data types used.

5.1 Summary

The implementation of Big Data pipelines across the Edge-to-Cloud computing continuum essentially requires trust management approaches. With the emergence of blockchain as an immutable ledger technology and the potential of Smart Contracts and trustless Smart Oracles, it is increasingly possible to overcome the limitations of traditional trust management approaches. In this work, we rely on transparency, traceability and autonomy as key features of blockchain-based services, and apply a new trust management approach to a highly dynamic and complex distributed smart application scenarios.

Our study considered several attributes that must be assessed and implemented in order to achieve trust in smart applications and the underlying decentralised system. While the values of some trust attributes can be obtained by using costly on blockchain operations, other trust attributes can be used through the use of less costly off-blockchain mechanisms, such as the use of QoS monitoring data as shown in this study.

A specific approach to trust proposed by this study is the ability to certify the entities and stakeholders (users and providers, IoT data sources, software components, Fog nodes, Cloud storage), and independently monitor their status through independent blockchain based services. This work also presents SC-based trust management scenarios for achieving data flow among the application components (from the camera to a Fog node, and from a Fog node to Cloud storage). Finally, this work concentrates on achieving high QoS in the operation of the smart applications. This is facilitated through the use of off-blockchain QoS monitoring metrics gathered through the use of a trustless Smart Oracle, and a Markov decision-making method that ranks the available Fog/Cloud node providers in order to select the optimal Fog node for the deployment of the AI part of the application.

Our work builds on earlier efforts that formalised trust. Its novelty lies in the practical implementation of a trust management system, which feeds the design of the advanced DECENTER's Fog Computing Platform.

6 Multi-Party Service Level Agreements for Multi-Party Data Governance

In this part of our experimentation, we concentrated on the possibility to establish multi-party Service-Level Agreements that would govern the exchange of data across the Edge-to-Cloud computing continuum. This assumes for example the ability to deploy a service container in a cloud provider that has been acquired via the DECENTER's Resource Exchange Broker. For such an occasion, it may be necessary to use various Smart Contract types. In our work, we first start with the essential features of Smart Contracts, that is the ability to monetise the use of the resource or service.

Since Solidity is a programming language with high-degree of expressivity, it can also be used to implement various additional mechanisms, such as permissions, certification, regulations and other mechanisms that would govern the AI models management. Therefore, our intention is to further extend this initial work so that we can develop our intended cross-border data management scenario which was described in the previous sections.

The application that we use for these experiments is based on a containerised version of the Jitsi Meet videoconferencing system. The implemented architecture is presented by using seven developed Smart Contracts, that help realise seven different pricing schemes for the service. As previously noted, this can be further applied to the AI models and it is therefore fully fitted to the realisation of the cross-border AI model management scenario.

6.1 Addressed requirements

In this work, we propose a new architecture for SLAs, which is demonstrated on an example of a Web based Video-Conferencing (VC) application, which is implemented in a Container Image (CI). We explore the possibility for QoS assurances, including the fulfilment of various hard data management requirements. By developing Smart Contracts for several generic monetisation use cases, we aim to cover a potentially large number of stakeholders according to their requirements for data and AI model transport and management, such as regular, occasional, demanding users, the application of constraints, negotiations and similar. This new architecture is designed having in mind the requirements for security and transparency, and monetization of the transactions. The new architecture is implemented for the VC application on top of blockchain. This makes it possible to empirically compare it with traditional monetisation approaches quantitatively and qualitatively.

Our work aims to improve the trusted data management, as well as the overall system reliability, dependability and robustness, when various software services are developed, engineered, deployed and operated in a highly distributed way, across the Edge-Fog-Cloud computing continuum. Several highly focused studies have used Blockchain technology to improve product traceability and quality preservation, such as the study of Lu et al. [1]. The work of Xia et al. [2] focused on adoption of Blockchain for trustless medical data sharing as a State-of-the-Art solution in the domain of medical Big Data. On the other hand, the overall performance of such systems has not been addressed sufficiently and they have not been described in the context of SCs. An attempt of integration of SCs in an existing Cloud system for VMI and CI management was presented on our recently completed Horizon 2020 ENTICE[18] project [3] as an agreement management component among users. The present

[18] http://www.entice-project.eu/

work uses the various benefits of SCs, in order to define, develop and test various monetisation strategies, which may be useful in the domain of the Cloud services economy.

6.2 Architecture for Multi-Party SLAs

In this work, we rely on advanced technologies, which are used in Cloud computing including Docker [19] for the management of containers, and Kubernetes [20] as a general purpose orchestration technology. Our VC application Jitsi Meet is therefore packed into Docker containers. In contrast to VMIs, containers can be spinned on and off much faster, practically within seconds, thus forming basis for fine-grained services orchestration, which is driven by various events (such as end users that need to run a VC). Whenever a specific software service is required, these services make it possible to dynamically deploy a container image and serve the particular end user or event.

The basic purpose of the proposed architecture is to efficiently offer VC sessions to the end users and enable flexible payments with Ethereum based cryptocurrencies. All the developed and used components representing the architecture are packed as Docker CIs. The focus of the present work addresses the needs to facilitate flexible usage of the resources and software related to the provisioning of the VC service. We also address the roles of the Smart Oracles that can be described as agents (e.g. monitoring services, sensor data providers and others) that finds and verifies real-world occurrences and submits this information to a Blockchain. There are many mixed opinions about the usage of Smart oracles and how to assure their objectiveness. So, in our work we describe a mechanism to minimise the bias decisions of the oracles with needed conformation of all involved system stakeholders.

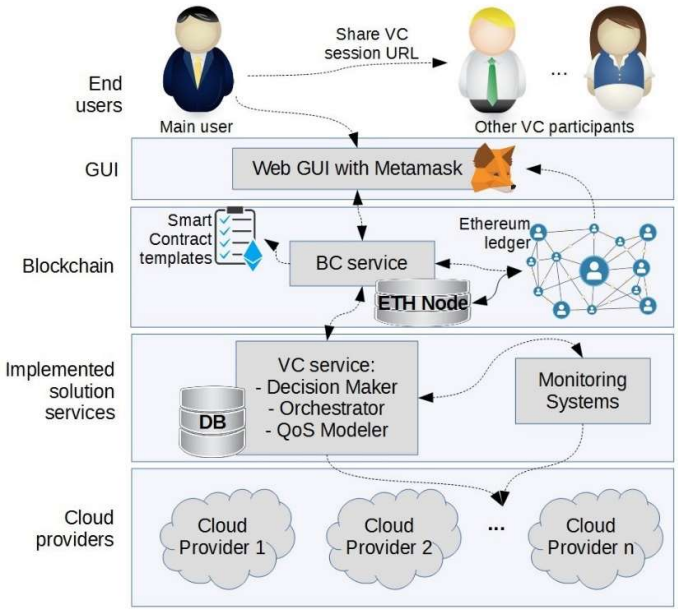

Figure 17. High-level architecture of a software as a service (SaaS) VC system.

[19] https://www.docker.com/
[20] https://kubernetes.io/

The novel architecture is designed to enable different monetisation approaches among four pillar component layers in any Cloud-based software engineering scenario: (i) Cloud providers, (ii) Implemented solution services, (iii) Blockchain components and (iv) Graphical User Interface (GUI).

The implemented architecture is depicted in and its main components are explained in the following.

Web based GUI is the main entry point for the end user who wants to establish a VC session. Each VC session consists of a main end user subscriber and other participants that are invited manually after VC session is established.

In the first step, the main end user subscriber defines main properties of the VC session such as number of participants, preferred quality of the video related to the bandwidth and optional properties such as estimated geolocations of the participants. In order to assure the Web GUI connectivity with the Ethereum ecosystem the users interact with the VC system through the Ethereum bridge Metamask[21] that is basically a browser add-on. It allows SC sign interactions (e.g. system login, VC session agreement conformation, VC session end request and others). In the second step, according to the end user's preferences, the SC terms are proposed to the main end user by our system by the rest of the components. In all monetisation use cases the agreement is reached when all involved parties sign the SC(s).

Blockchain components consist of a public Blockchain ledger, in our case Ethereum. The developed SC templates supports all the proposed monetisation use cases. See Section 5 for details. The distributed node infrastructure is maintained by the Ethereum community and offers a high level of distribution and availability. The main API technologies are written in the Java programming language, thus the core bridge to the Ethereum ledger is the Java library ethereumj[22] embedded in the core layer API. The deployed SC instances may be triggered by only those stakeholders that are registered by the SC creator. In all SCs trigger executions and phases are updated by the actual content in the Ethereum blockchain network.

Implemented service solutions are composed of different components that comprehensively perform different tasks. For example, to estimate the potential QoS of the deployed VC service it is mandatory to have QoS metrics defined and measured, while the VC service is running. By doing this, the QoS metrics are modelled, while the Decision Maker identifies potential Cloud providers, where the container image should be started.

Cloud providers are available on different geographical locations and are used for the deployment of VC service instances. The VC application components, which are packed 265 into Docker containers, can be deployed on demand when an agreement between the VC service and involved stakeholders is reached. This follows the SaaS delivery method. Besides, the deployed VC application instances can be monitored to provide metrics, such as usage time period, frame rate of the VC session and other QoS metrics that can be included in an SLA.

The general workflow of the architecture consists of the agreement among an end user and the VC service to determine the monetisation use case, QoS user's requirements and the price. After the VC service deploys the CI application components, the main end user gets unique URL for the VC session that can be shared among other participants. The application life-cycle is completed when the end user finishes using the VC application or if any other

[21] https://metamask.io/
[22] https://github.com/ethereum/ethereumj

condition occurs, e.g. the VC session time is exceeded. In addition, the resources are freed by the undeployment of VC session service on the Cloud providers' infrastructure. In the final stage the Ethereum SCs are signed by both parties, the end user and the VC service, and thus the final monetisation ow is executed. A complete workflow with the consensus decision monetisation use case is presented in the following section.

6.3 Smart Contracts design and implementation

Following the design and implementation of an architecture for using SCs, our next goal was to study different monetisation approaches that can be used by the stakeholders that contribute resources and services. This is illustrated through the development and deployment of a working VC software service. We present a basic comparison between conventional and Blockchain monetisation on the Ethereum network.

In order to ensure feasibility of the monetisation method applied in our system, we summarised basic properties of the Ethereum network and compared them with three popular payment systems: Visa, Mastercard and Paypal. The comparison results in Table 2show that the overall cost of Ethereum monetisation is significantly lower compared to traditional methods. Ethereum transaction cost represented with GWEI unit, where 1 ETH = 109 GWEI, is tightly dependent by the following factors: Ethereum network congestion, preferred transaction speed and the actual price of ETH coin. To minimize the ETH coin volatility there is a dedicated ERC20 token TrueUSD[23], which is an USD backed, fully collateraized, transparent and legally protected. Despite that, ETH funds have to be available in order to be substracted for fees as ETH Gas while triggering SCs (e.g. deploying, write data, update states etc.). In addition, SC custom defined functions allow a higher level of transaction exibility. For example, lock-in of transaction funds refers to the actual locking of funds in the SC until certain functional conditions have not been reached and releasing the funds, or just fractioned funds, to the appropriate user entity. In alternative cases these mechanisms are executed off-chain, without a direct access by the end user by trusting third party services.

Table 2. Main properties of different monetisation methods in the first half of year 2018.

Monetisation method	Transaction processing fee	Merchant or operational service cost [USD]	Advanced functionalities
Visa	1.43% - 2.4%	min 1.25%	off-chain
Mastercard	1.55% - 2.6%	min 1.25% + 0.05	off-chain
PayPal	2.9% - 4.4%	min 1.5%	off-chain
Ethereum	1 - 40 GWEI[16]	0	on-chain

6.4 Monetisation use cases

The monetisation processes for our VC system from the viewpoint of the various stakeholders was described with several use cases as follows. The use cases describe different user needs

[23] https://www.trusttoken.com/

based on actual usage patterns of resources and services. These can be achieved through the definitions of SCs. Another important aspect, which is considered concerns the income division among the parties, including the Cloud providers, container image deployment platforms, infrastructure providers and so on. In order words, we analysed the needs of all stakeholders who are in charge for an individual VC service setup process.

Fixed price is the most basic definition of the use case. The VC system, depending on the QoS end user's requirements, offers the usage of VC service for a fixed price and fixed maximum period of time on demand. The agreement is reached when VC system deploys and initialises the SC, while the end user sends the required ETH funds. On successful payment the event notifies the VC system and initialises the Docker instances as a VC session.

Dynamic price, in addition to fixed price use case that involves the same stakeholders (VC service and end user), offers higher flexibility of the actual VC service availability. For example, the end user knows the maximum time that that he/she might need the VC service. After the agreement is reached the user pays the full price but the ETH funds are locked by the SC. The funds become unlocked if the maximum period is reached or if the end user stops using VC service by triggering the SC function. In the unlocking phase the actual usage time is charged - the proportional ETH funds of unused time is returned to the end user while the rest is sent to the VC service.

Time-limited usage refers to the use case when the end user in advance agrees the per-minute conversation price, buys certain VC session minutes and uses the VC service gradually on demand. In this case VC system deploys and initialises the SC while the end user triggers VC session start/stop actions through the SC function linked to the event listening in our VC system. A typical usage is suitable for Big Brother-like TV shows and other content provided in real-time online. ***One could easily imagine the use of the AI model on an SGX chip for specific limited time after which the model is destroyed.***

Flexible usage period offers more flexibility for the end users that cannot define the exact period when the VC service will be used. In the proposed end user's period, the VC service has to be either available or the deployment of the VC service has to be optimised matching the more effective (faster) agreed QoS deployment. The charge methodology follows the dynamic price use case including the stakeholders where the SC contains also the minimum charge price for the maintenance of the faster deployment of the containerised VC services in the specified time period.

Division of income is a particular monetisation process where any additional parties, beside the VC service provider and end user, participate in the provisioning of the VC service, that is, its software or hardware infrastructure or provided content. For example, one provider offers specific VC containers, another offers infrastructure and the third provider offers other services, such as monitoring. The division of income is agreed upon the parties in advance and is written in the instance of the SC, e.g. the price per hour for using the specific service. An overall advantage of this approach for the end user is reflected as a lower overall price of the VC service. In addition, this approach aims to collaboratively include specific services, in the Cloud or even in the Fog, that does not need to be actually maintained in the infrastructure of the VC system owner.

Consensus decision is an upgrade to division of income use case. Consensus is a mechanism that could ***readily be used for cross-border data management.*** The consensus is reached through a democratic voting SC, similar to the one proposed by S. Bragagnolo et al. [4]. As result the management of VC service is divided among parties specialised for different infrastructure or service aspects and thus improved QoS, QoE, availability and overall cost

reduction. The end user is not directly aware of the ETH funds being distributed among VC enabling services although it can be read in the SC instance.

Constraint based focuses on the legal aspects determined by the end user's constraints. As an example, the EU General Data Protection Regulation (GDPR) compliant can be agreed by all end users through SC. In some cases, ***such as cross-border data management***, geographical definition is also important and should be determined by the end user and VC system in advance (e.g. not all data services are allowed in specific countries). In general, by constraint limiting, either the price for the end user, either the potential QoS, changes (VC service price increase, decreased QoS) in the proposition of SC by the VC service.

The presented monetisation use cases offer software services with ability to observe the achieved high QoS/QoE operation, which includes the ability for regulatory alignments, in relation to legislation, policies and permissions. The differences among the monetisation use cases are presented in Figure 18. It makes it possible to compare the VC service monetisation process in time.

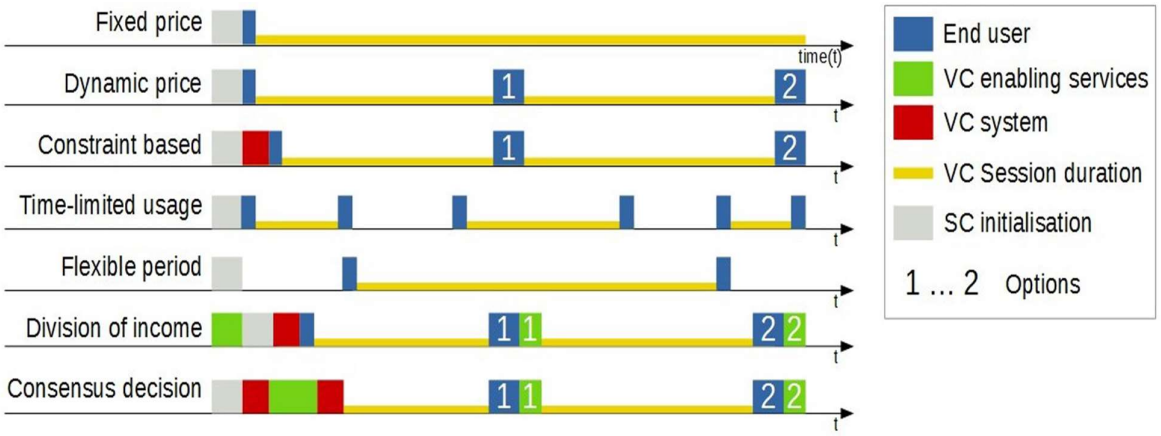

Figure 18. The designed Multi-Party Smart Contracts.

6.5 Implementation details

The development process of SCs started with environment setup where we aimed to facilitate the integration in existing components by using entirely Java programming language in the back-end while in the front-end we use standard JavaScript approaches that enable the integration with Metamask bridge.

During the development the following assumptions were considered: (i) For each monetisation use case we prepare only one individual SC to reduce the cost of deployment of another SC and consequentially the delay of the deployment operation. The disadvantage of this approach is decreased scalability that may be available in case of multi-SCs initializing new ones; (ii) The specific SCs developed for the service needs to be carefully designed and validated (e.g. by using Oyente SC validation tool), analysed and evaluated at different levels, such as code pattern comparison and simulation of actual usage with multiple parties on the same SC. Besides that, OpenZeppelin framework provided reusable tested SCs templates for Ethereum; (iii) The initial state of all deployment addresses owned by VC system contain 0 ETH; (iv) The implementation trends followed object-oriented programming approach, especially inheritance of SCs as shown in Figure 19. The actual Solidity code of the monetisation use cases is available in the BitBucket repository; and (v) Actual behaviour of SCs need to be adequately

observed (e.g. time elapsing of individual functions, cost estimation of operations etc.) in a main or testing Ethereum environment network.

6.6 Walk-through for dynamic price monetisation

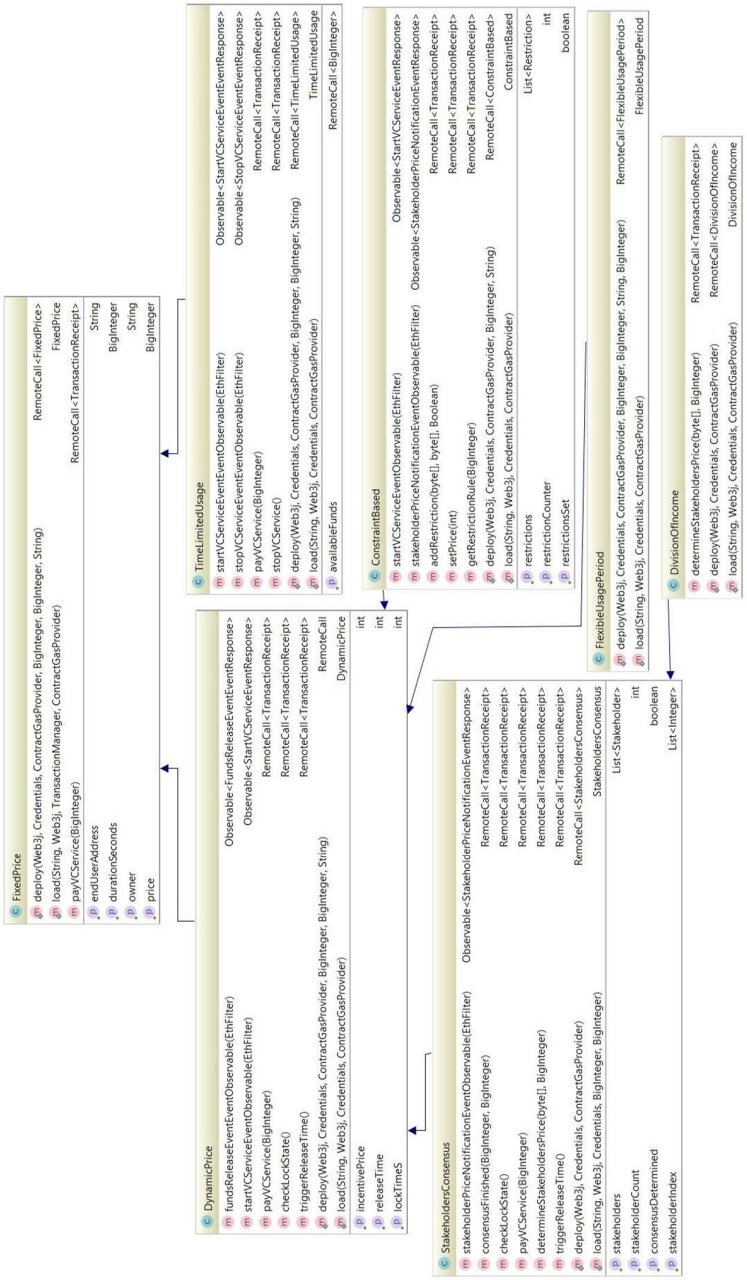

Figure 19. The class diagram represents SCs' main attributes, methods and inheritance class relationships.

6.7 Experiments

We conducted a series of VC system simulations by using available monetisation use cases on two Ethereum testnet environment, more concrete Rinkeby[24] using PoA.

The workflow of the proposed methodology is depicted in Figure 20. In the SCs development phase we use a variety of tools (validate, optimise, design pattern frameworks and others). The next phase, experimental setup consists of generating testing addresses of the stakeholders single addresses for all stakeholders, except enabling services that are multiple for each role (e.g. monitoring system, orchestrator and Cloud provider). In order to execute transactions all the addresses need to contain ETH funds that are claimed through so called Authenticated Faucet.

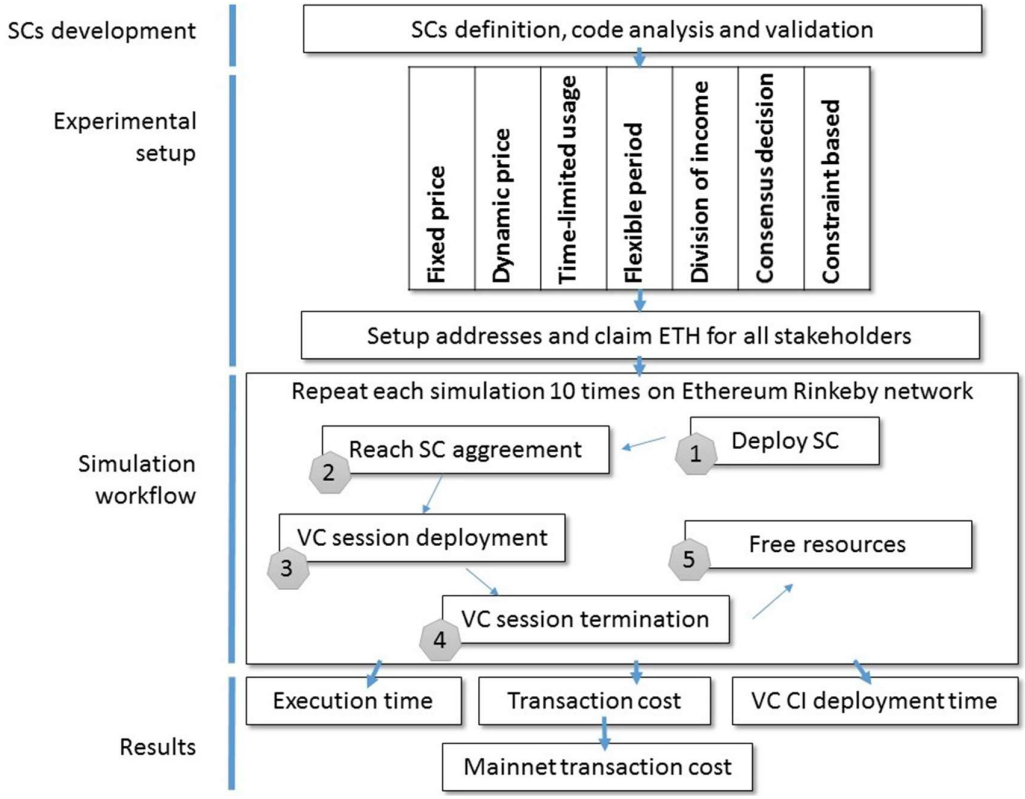

Figure 20. The main flow of the proposed VC system evaluation.

In the phase of the simulation workflow the VC system life-cycle is represented with the following steps:

1. Deployment of the SC on an address accessible from a VC system.
2. SC agreement is reached through function triggering depending on monetisation use case.

3. VC session is deployed according to the SC specifics (e.g. geolocation, required SLA etc.).

[24] https://www.rinkeby.io

4. VC session is terminated either when the reserved time is exceeded or on demand by the end user and thus SC event notifies the VC system.

5. The VC system resources are released.

Through the proposed simulation workflow, we are able to obtain Ethereum testnet based quantitative metrics such as execution time and transaction cost. These metrics can be further objectively mapped to Ethereum main net environment and discussed in the following Section. Furthermore, we measured deployment time of our VC CI on five different geolocations.

The experimental environment is based on running our own testnet Ethereum node (Rinkeby) to fully exploit the network performances. For example, by using the common known alternative Infura Web tool[25] with the API based access to the Ethereum testnet node we are not able to run observers for blocks and confirmed or pending transactions. Since we run our own Ethereum testnet node, we can observe the updates of the ledger after the execution of the (synchronous) transaction and at the same time observe (asynchronously) the content of the new blocks added into the blockchain, and thus determine the end time of the transaction based on which triggers first. In order to comprehensive integrate the SCs with other services we used Ethereum Virtual Machine (EVM) events. All available monetisation use cases has been simulated 10 times where the results has been averaged.

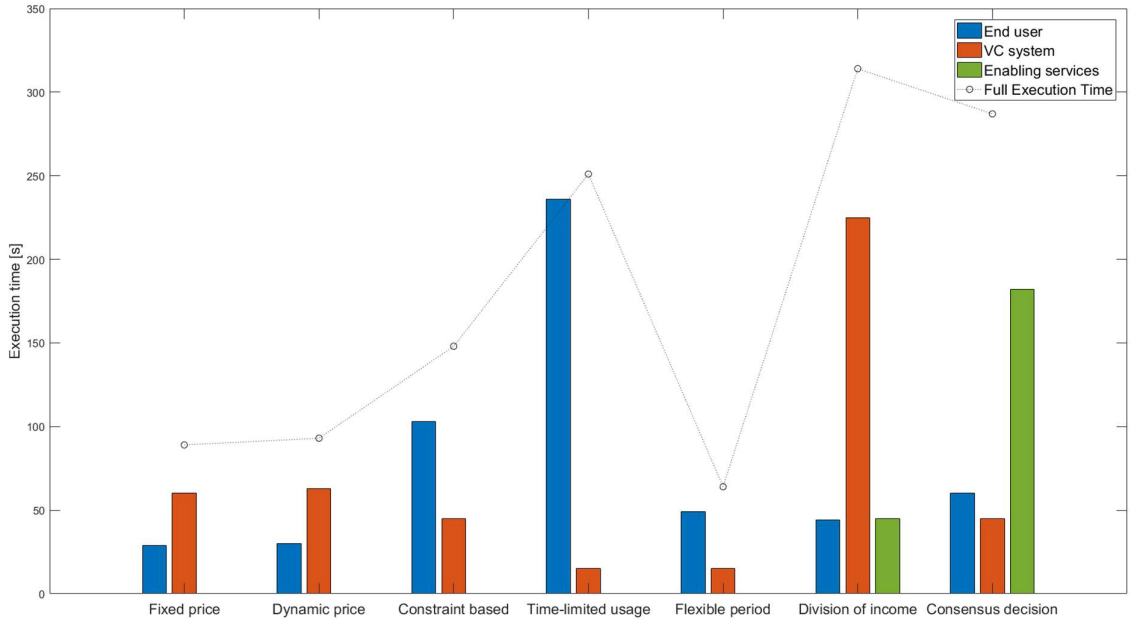

Figure 21. Performance results of SCs among different stakeholders on Ethereum Rinkeby testnet environments.

Figure 21 shows that the execution time of the transactions triggered by the stakeholders is mainly affected by the quantity of them where a theoretical minimum for each transaction is the block generation time which is approximate 15 second. Furthermore, deployment of SCs generally take time of two blocks cycle generation to be included into the block. In case of time-limited usage the execution times of the end user is affected drastically but it mainly

[25] https://infura.io

consists of triggering starting/stopping VC session 5 times where in the stopping phase does not affect the QoE of the end user. In cases when enabling services interact with the SCs (division of income and consensus decision), these transactions also does not affect QoE of the end users.

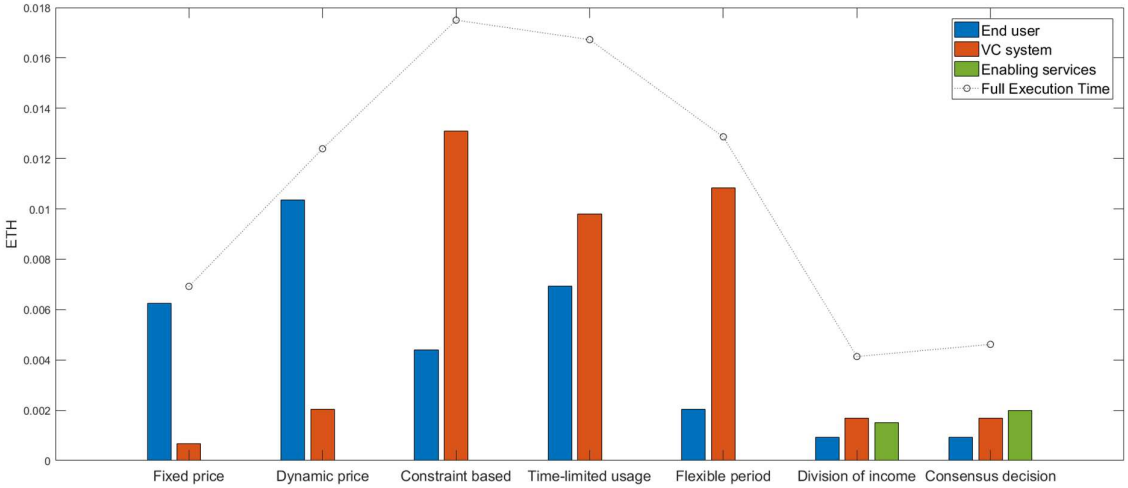

Figure 22. Comparison of GAS cost estimation when executing Multi-Party SCs.

Figure 22 describes the cost of the transactions by using the default GAS provider. In average the VC system transactions are the most expensive due to the higher data size that needs to be written in the blocks (e.g. deployment of SCs). Additionally, the transaction cost is affected by the selection of input data and quantity of the function attributes. In case of constraint based the constraints, defined as table of bytes (byte32), are significantly higher than numerically defined values such as integers. In cases when multiple different stakeholders are interacting with the SCs (division of income and consensus decision), the transaction costs are 8 times lower compared to the others.

After each successful monetisation agreement is reached among the end user and VC system, the actual deployment time of the VC system. The software is packed into CI and it is deployed through Kubernetes CI manager software. This aspect is very important while calculating the actual running time of the VC session because it needs to be summed up after the SC agreement is written into the block. Our infrastructure running Kubernetes has the following properties: 4 vCPUs with hardware accelerated CPU virtualisation powered by Intel Xeon E5649 CPUs clocked at 2.53GHz, 2GB of RAM and 20GB of HDD. In the deployment experiment we deployed the CI VC session 10 times on different geolocations as presented in Table 2.

Table 3. Deployment times of CI based VC session on different geolocations.

Cloud name	Continent	Deployment time [ms]
Arnes	Europe	3133
flexiOps	UK	8033
GKE-EU-WEST	Europe	1866
GKE-ASIA-EAST	Asia	5133
GKE-US-WEST	USA	2266

6.8 Summary

We presented an architecture for monetisation of a VC system that follows the pay-as-you-go concept. This is demonstrated by using seven Multi-Party SCs that cover the different demands of the end users. Compared to the traditional monetisation out system, this new architecture offers significantly lower costs (e.g. transaction, operational and others). Additionally, a trustless layer is implemented by using SCs and records of the provided QoS. It is shown that SCs are suitable for establishing transparent mutual agreements among end users and the VC system. This new architecture therefore facilitates the overall payment process without any additional third-party cost for the service owners (e.g. bank, payment cards and other fees).

Furthermore, by using SCs multi-party SLA can be setup more transparently on heterogeneous Cloud architectures. The results show that in practice it is better to have less functions in the SCs to reduce the transaction times, but on the other hand the input data of SCs should be pragmatically defined by using minimal required sizes of the data types, and thus lowering the transaction costs.

In the experimental study we analysed individual monetisation use cases with the intent to prove the feasibility of the proposed architecture for cross-border AI model management. With the introduction of a Blockchain layer, the system is implemented as a distributed Ethereum node storage and it represents a full environment that uses SCs. The running node requires upwards of 120 GB of storage and 8 GB+ of memory. For each monetisation use case we are able to determine the transaction costs for each stakeholder involved that is approximate 0.10 to 0.25 USD per transaction.

Currently, the main paradigm with SCs is the management of the data received from outside a Blockchain from so called Smart Oracles. Smart Oracles are still in their infancy, however they have the potential to provide trusted data, for example, from physical persons, organisational policies, and government agencies. ***Particularly, a Smart Oracle based system would provide the applicable rules for cross-border data management supplied by multiple parties that wish to govern each individual data transport and data management use case instance.***

7 Conclusions

Our initial study ***on cross-border data management*** managed to achieve the following:

- Key requirements for cross-border data management were analysed
- Specific use case of AI models cross-border management was selected for implementation
- Repository for AI models was designed and preliminary implementation was tested
- Detailed scenario and sequence diagram were drafted for implementation
- The attributes of trust as well as initial architectures for trust management in the DECENTER's Fog Computing Platform were studied
- Several Multi-Party Smart Contracts were designed to serve as basis for the implementation of mechanisms for Multi-Party governance of the data management

Having achieved these goals, the partners now proceed in implementing the designed scenario in the next project phase.

References

[1] A. Anjum, M. Sporny, A. Sill, Blockchain standards for compliance and trust, IEEE Cloud Computing 4 (4) (2017) 84{90. doi:10.1109/MCC. 2017.3791019.

[2] Q. Xia, E. B. Sifah, K. O. Asamoah, J. Gao, X. Du, M. Guizani, Medshare: Trust-less medical data sharing among cloud service providers via blockchain, IEEE Access 5 (2017) 14757{14767. doi:10.1109/ACCESS. 2017.2730843.

[3] S. Gec, U. Paščinski, V. Stankovski, Semantics for the Cloud: The ENTICE integrated environment and opportunities with smart contracts (Jan. 2018).doi:10.5281/zenodo.1163999. URL https://doi.org/10.5281/zenodo.1163999

[4] S. Bragagnolo, H. Rocha, M. Denker, S. Ducasse, Smartinspect: solidity smart contract inspector, in: 2018 International Workshop on Blockchain Oriented Software Engineering (IWBOSE), 2018, pp. 9{18. doi:10.1109/ IWBOSE.2018.8327566.

[5] A. M. Antonopoulos, Mastering Bitcoin: unlocking digital cryptocurrencies. ”O’Reilly Media, Inc.”, 2014.

[6] M. Gault, “Blockcloud: Re-inventing cloud with blockchains,” https://guardtime.com/blog/blockcloud-re-inventing-cloud-with-blockchains, 2015, online; Accessed December 2019.

[7] M. Swan, Blockchain: Blueprint for a new economy. ”O’Reilly Media, Inc.”, 2015.

[8] P. Franco, Understanding Bitcoin: Cryptography, engineering and economics. John Wiley & Sons, 2014.

[9] D. Shrier, W. Wu, and A. Pentland, “Blockchain & infrastructure (identity, data security),” connection.mit.edu, 2016.

[10] D. Schwartz, N. Youngs, and A. Britto, “The ripple protocol consensus algorithm,” Ripple Labs Inc White Paper, p. 5, 2014.

[11] D. Mazieres, “The stellar consensus protocol: A federated model for internet-level consensus,” Draft, Stellar Development Foundation, 15th May, available at: https://www. stellar.org/papers/stellarconsensus-protocol.pdf (Accessed December 2019).

[12] E. Barker, “Recommendation for digital signature timeliness,” NIST Special Publication, vol. 800, p. 102, 2009.

[13] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” 2008.

[14] C. Dwork and M. Naor, “Pricing via processing or combatting junk mail,” in Annual International Cryptology Conference. Springer, 1992, pp. 139–147.

[15] F. Koeppe and J. Schneider, “Do you get what you pay for? Using proof-of-work functions to verify performance assertions in the cloud,” in Cloud Computing

Technology and Science (CloudCom), 2010 IEEE Second International Conference on. IEEE, 2010, pp. 687–692.

[16] L. Lamport, “Constructing digital signatures from a one-way function,” Technical Report CSL-98, SRI International Palo Alto, Tech. Rep., 1979.

[17] A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder, Bitcoin and cryptocurrency technologies. Princeton University Press, 2016.

[18] B. Preneel, “Cryptographic hash functions,” European Transactions on Telecommunications, vol. 5, no. 4, pp. 431–448, 1994.

[19] J.-S. Coron, Y. Dodis, C. Malinaud, and P. Puniya, “Merkle-Damgård revisited: How to construct a hash function,” in Annual International Cryptology Conference. Springer, 2005, pp. 430–448.

[20] D. Bradbury, “The problem with bitcoin,” Computer Fraud & Security, vol. 2013, no. 11, pp. 5–8, 2013.

[21] S.A. Rompf, The concept of trust, in: Trust and Rationality, Springer, 2015, pp. 29–78.

[22] D. Gambetta (Ed.), Can We Trust Trust? Trust: Making and Breaking Cooperative Relations, Blackwell, Cambridge, Massachusetts, 1988, pp. 213–237.

[23] J. Sabater, C. Sierra, Review on computational trust and reputation models, Artif. Intell. Rev. 24 (1) (2005) 33–60, http://dx.doi.org/10.1007/s10462- 004-0041-5.

[24] L. Viljanen, Towards an ontology of trust, in: S. Katsikas, J. López, G. Pernul (Eds.), Trust, Privacy, and Security in Digital Business, Springer Berlin Heidelberg, Berlin, Heidelberg, 2005, pp. 175–184.

[25] M. Wöhrer, U. Zdun, Design patterns for smart contracts in the Ethereum ecosystem, 2018.

[26] U. Paščinski, J. Trnkoczy, V. Stankovski, M. Cigale, S. Gec, QoS-Aware orchestration of network intensive software utilities within software defined data centres, J. Grid Comput. 16 (1) (2018) 85–112, http://dx.doi.org/10. 1007/s10723-017-9415-1.

[27] B. Carminati, E. Ferrari, C. Rondanini, Blockchain as a platform for secure inter-organizational business processes, in: 2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC), 2018, pp. 122–129.

[28] F. Zhang, E. Cecchetti, K. Croman, A. Juels, E. Shi, Town Crier: An Authenticated Data Feed for Smart Contracts, Cryptology ePrint Archive, Report 2016/168, 2016, https://eprint.iacr.org/2016/168.

[29] A. Alrawais, A. Alhothaily, C. Hu, X. Cheng, Fog computing for the internet of things: Security and privacy issues, IEEE Internet Comput. 21 (2) (2017) 34–42.

[30] K.K. Bimrah, H. Mouratidis, D. Preston, Itrust: a trust-aware ontology for information systems development, 2008.

[31] F. Corradini, F. De Angelis, F. Ippoliti, F. Marcantoni, A survey of trust management models for cloud computing, in: CLOSER, 2015, pp. 155–162.

[32] J.H. Ziegeldorf, O.G. Morchon, K. Wehrle, Privacy in the Internet of Things: threats and challenges, Secur. Commun. Netw. 7 (12) (2014) 2728–2742.

[33] P. Hu, H. Ning, T. Qiu, H. Song, Y. Wang, X. Yao, Security and privacy preservation scheme of face identification and resolution framework using fog computing in internet of things, IEEE Internet Things J. 4 (5) (2017) 1143–1155.

[34] P. Zhang, M. Zhou, G. Fortino, Security and trust issues in fog computing: A survey, Future Gener. Comput. Syst. 88 (2018) 16–27.

[35] J. Ni, K. Zhang, X. Lin, X.S. Shen, Securing fog computing for internet of things applications: Challenges and solutions, IEEE Commun. Surv. Tutor. 20 (1) (2017) 601–628.

[36] S.M. Habib, S. Ries, M. Muhlhauser, Towards a trust management system for cloud computing, in: Trust, Security and Privacy in Computing and Communications (TrustCom), 2011 IEEE 10th International Conference on, IEEE, 2011, pp. 933–939.

[37] E. Mostajeran, M.F. Khalid, M.N.M. Mydin, B.I. Ismail, H. Ong, Multifaceted trust assessment framework for container based edge computing platform, in: Fifth International Conference on Advances in Computing, Control and Networking-ACCN 2016, 2016.

[38] S.K. Prajapati, S. Changder, A. Sarkar, Trust management model for cloud computing environment, 2013, arXiv preprint arXiv:1304.5313.

[39] L. Chen, J. Xu, Socially trusted collaborative edge computing in ultra dense networks, in: Proceedings of the Second ACM/IEEE Symposium on Edge Computing, ACM, 2017, p. 9.

[40] V. Sharma, I. You, D.N.K. Jayakody, M. Atiquzzaman, Cooperative trust relaying and privacy preservation via edge-crowdsourcing in social Internet of Things, Future Gener. Comput. Syst. (2017).

[41] W. Li, H. Song, ART: An attack-resistant trust management scheme for securing vehicular ad hoc networks, IEEE Trans. Intell. Transp. Syst. 17 (4) (2016) 960–969.

[42] T. Wang, G. Zhang, M.Z.A. Bhuiyan, A. Liu, W. Jia, M. Xie, A novel trust mechanism based on fog computing in sensor–cloud system, Future Gener. Comput. Syst. (2018).

[43] R. Chen, J. Guo, F. Bao, Trust management for SOA-based IoT and its application to service composition, IEEE Trans. Serv. Comput. 9 (3) (2016) 482–495.

[44] S.A. Soleymani, A.H. Abdullah, M. Zareei, M.H. Anisi, C. Vargas-Rosales, M.K. Khan, S. Goudarzi, A secure trust model based on fuzzy logic in vehicular ad hoc networks with fog computing, IEEE Access 5 (2017) 15619–15629.

[45] D. Chen, G. Chang, D. Sun, J. Li, J. Jia, X. Wang, TRM-Iot: A trust management model based on fuzzy reputation for internet of things, Comput. Sci. Inf. Syst. 8 (4) (2011) 1207–1228.

[46] F. Mora-Gimeno, H. Mora-Mora, D. Marcos-Jorquera, B. Volckaert, A secure multi-tier mobile edge computing model for data processing offloading based on degree of trust, Sensors 18 (10) (2018) 3211.

[47] R. Di Pietro, X. Salleras, M. Signorini, E. Waisbard, A blockchain-based trust system for the internet of things, in: Proceedings of the 23nd ACM on Symposium on Access Control Models and Technologies, ACM, 2018, pp. 77–83.

[48] N. Alexopoulos, S.M. Habib, M. Mühlhäuser, Towards secure distributed trust management on a global scale: An analytical approach for applying distributed ledgers for authorization in the iot, in: Proceedings of the 2018 Workshop on IoT Security and Privacy, ACM, 2018, pp. 49–54.

[49] B. Yu, J. Wright, S. Nepal, L. Zhu, J. Liu, R. Ranjan, IoTChain: Establishing trust in the internet of things ecosystem using blockchain, IEEE Cloud Comput. 5 (2018) 12–23, http://dx.doi.org/10.1109/MCC.2018.043221010.

[50] M.T. Hammi, B. Hammi, P. Bellot, A. Serhrouchni, Bubbles of trust: A decentralized blockchain-based authentication system for iot, Comput. Secur. 78 (2018) 126–142.

[51] P. Missier, S. Bajoudah, A. Capossele, A. Gaglione, M. Nati, Mind my value: a decentralized infrastructure for fair and trusted iot data trading, in: Proceedings of the Seventh nternational Conference on the Internet of Things, ACM, 2017, p. 15.

[52] M. Wöhrer, U. Zdun, Design patterns for smart contracts in the Ethereum ecosystem, 2018.

[53] P. Kochovski, P.D. Drobintsev, V. Stankovski, Formal quality of service assurances, ranking and verification of cloud deployment options with a probabilistic model checking method, Inf. Softw. Technol. (2019) http: //dx.doi.org/10.1016/j.infsof.2019.01.003.

Abbreviations

WP	Work Package
SLA	Service Level Agreement
SLO	Service Level Objective
SMI	Service Measurement Index
ISO	International Organization for Standardization
IT	Information Technology
QoE	Quality of Experience
QoS	Quality of Service
K8S	Kubernetes
ML	Machine Learning
AI	Artificial Intelligence
SaaS	Software-as-a-service
PaaS	Platform-as-a-service
IaaS	Infrastructure-as-a-service
CPU	Central Processing Unit
GPU	Graphics Processing Unit
RAM	Random Access Memory
IoT	Internet of Things
AWS	Amazon Web Services
VM	Virtual Machine
GUI	Graphical User Interface
CLI	Command Line Interface
API	Application Programming Interface
REST	Representational State Transfer
REB	Resource Exchange Broker
DB	Database
VPN	Virtual Private Network
UML	Unified Modelling Language
DAG	Directed Acyclic Graph
NGI	Next Generation Internet
BC	Blockchain

DLT	Distributed Ledger Technology
MTBF	Mean Time Between Failures
MTTR	Mean Time To Repair
MQTT	Message Queuing Telemetry Transport
HTTP	HyperText Transfer Protocol
IDE	Integrated Development Environment
OSGI	Open Service Gateway Initiative
ETH	Ethereum
PoW	Proof of Work
BC	Blockchain