Secure and Privacy Driven Energy Data Analytics

by

Dhanya Therese Jose

A dissertation submitted in partial satisfaction of
the requirements for the degree
PHILOSOPHIAE DOCTOR (PhD)

Faculty of Science and Technology
Department of Electrical Engineering and Computer Science
November 2022

University of Stavanger
N-4036 Stavanger
NORWAY
www.uis.no

ISBN 978-82-8439-132-8
ISSN 1890-1387

PhD Thesis UiS no. 673

Dedicated to my Angels,
Gizel and Sianna

Acknowledgements

This research was carried out at the Department of Electrical Engineering and Computer Science of the University of Stavanger (UiS) in the period from June 2018 to August 2022. I acknowledge with gratitude, the financial and consulting support I received from the Research Council of Norway, as well as partners of the project Lyse, DNV GL and Statnett.

First, I would like express my sincere gratitude to my supervisor, Prof. Dr. Chunming Rong for his continuous support and guidance during my PhD study and related research. His astuteness helped me to improve the quality of my research. My gratitude also goes to my co-supervisor Dr. Antorweep Chakravorty for his support, valuable discussions and comments on the work presented in this dissertation. Also I would like to thank Dr.Martin Gilje Jaatun for his suggestions and support during my research.

I would like to thank head of the department Dr. Tom Ryen and my colleagues at UiS, Dr Mina Farmanbar, Dr. Jayachander Subiryala, Dr. Aida Mehdipourpirbazari, Dr. Nikita Karandikar, Dr.Cristina Viorica Heghedus, Dr.Albana Roci and Dr.Rituka Jaiswal for their support and friendship during the course of my research.

I would also like to thank my parents Joseph Mathew and Liby Jose, in-laws Dr. P L Jose and Dr. N S Mariya , brothers Deepu Jose and Jerin Jose, sisters Elizabeth Jose and Anjaly Thomas, nieces Minna and Hannah, nephew Luka and my friends for their support and encouragement during and beyond the period of this research.

Finally, I am grateful to my mentor, inspiration and role model, my husband Dr.Jithin Jose for his help and boundless support through out this journey. The completion of this work would not have been possible without my two little angels Gizel Marie Jose and Sianna Elizabeth Jose. Thanks for understanding your Amma, dears.

Above all, thanks to God, the Almighty, for his showers of blessings which enabled me to complete this research successfully.

Dhanya Therese Jose, November 2022

Preface

This dissertation is submitted in partial fulfillment of the requirements for the degree of Philosophiae Doctor (PhD) at the University of Stavanger, Norway. The work presented in this thesis are based on research conducted during the period June 2018 to August 2022 inclusive. Four published and one accepted research paper form the basis of this dissertation. The publications are reformatted to align with the thesis pagination. The articles are self-contained.

Abstract

Renewable resources are the main energy sources in a smart grid project. In order to ensure the smooth functioning of the smart grid, Information and Communication Technologies (ICT) need to be utilised efficiently. The objective of the SmartNEM project is to effectively utilise the technologies such as Machine Learning, Blockchain and Data Hubs for the aforementioned purpose and at the same time ensure a secured and privacy preserved solution. The data involved in smart grids require high security and it can be sensitive due to the household data which contains personal information. The individuals can be reluctant to share these data due to mistrust and to avoid unnecessary manipulation of the data they provide.

In order to overcome this it is necessary to build a trust based framework in which one could ensure data security and data privacy for the data owners to open up their data for data analysis. To achieves this we have proposed an architecture called TOTEM, Token for Controlled Computation, which integrates Blockchain and Big Data technologies. The conventional method of data analysis demands data be moved across the network to the location where the execution happens, however in the TOTEM architecture computational code will be moved to the data owner's environment where the data is located. The TOTEM is a three layer architecture (Blockchain consortium layer, Storage layer and Computational layer) with two main actors, data provider and data consumer. Data provider provides metadata of the data they own and provide resources for the execution of data. Data consumers will get an opportunity to execute their own code on the data provider´s data. For a controlled computation and to avoid malicious functions an entity called totem is introduced in the architecture. The authorised users should meet the requirements of Totem value for executing their code on the requested data. For live monitoring of the totem value throughout the run time is achieved with the components such as totem manager and updaters in the computational layer. The code must follow a specific format and will undergo preliminary checks with the TOTEM defined SDK and smart contracts deployed by the data providers in the blockchain network. The Extended TOTEM architecture is also proposed to

address the additional features when it is needed to combine the results from multiple data providers without sharing the data. This research work focused on the design of the TOTEM architecture and implementation as a proof of concept for the newly introduced components in the architecture. We have also introduced artificial intelligence in the framework to improve core features' functionality.

In the present research, the TOTEM architecture is proposed for the SmartNEM project to utilize the energy data for decision making and figure out the trends or patterns, while maintaining data privacy, data ownership, accountability and traceability. Moreover, the architecture can be extended to other domains such as health, education, etc, where data security and privacy is the key concern in sharing the data.

Contents

List of Figures

List of Papers

The following papers are included in this thesis:

- **Paper I**

  TOTEM : Token for controlled computation
  Dhanya Therese Jose, Antorweep Chakravorty, Chunming Rong
  Published in the proceedings of "10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)". IEEE, 2019.

- **Paper II**

  Distributed computational framework in TOTEM architecture enabled by blockchain
  Dhanya Therese Jose, Antorweep Chakravorty, Chunming Rong
  Published in the proceedings of "15th International Conference on Computer Science Education (ICCSE)". IEEE, 2020

- **Paper III**

  TOTEM SDK: an open toolset for token controlled computation managed by blockchain
  Behfar Behzad, Dhanya Therese Jose, Antorweep Chakravorty,

Chunming Rong
Published in the proceedings of "IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE)". IEEE, 2021

- **Paper IV**

Integrating Big Data and Blockchain to Manage Energy Smart Grid - TOTEM Framework
Dhanya Therese Jose, Jørgen Holme, Antorweep Chakravorty, Chunming Rong
Published in the journal "Blockchain: Research and Applications". Elsevier.2022

- **Paper V**

Application of Artificial Intelligence in secure decentralized computation enabled by TOTEM
Dhanya Therese Jose, Antorweep Chakravorty, Chunming Rong
Accepted in "IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE)". IEEE, 2022.

Chapter 1

Introduction

In this chapter, an introduction to the research work is presented and is structured as follows. Section 1.1 briefly explains the importance of smart grids and describes the smart community neighbourhood project. The research problem and the motivations behind the study are presented in Section 1.2. In section 1.3, the main objective and the research questions of this research are stated. The list of research articles relevant for the thesis is presented in Section 1.4 and finally, an outline of the thesis is provided in Section 1.5

1.1 Introduction to Smart Grid

In a conventional power grid system, the electricity will be generated from power stations and it will be transmitted through high-voltage transmission lines to the consumers. It is represented as a centralised one-way flow of electricity. Power stations generate electricity from natural gas, coal and nuclear energy or from renewable sources such as wind, hydro and solar power. These traditional power grids are updated in many ways such as advanced power electronics and control mechanisms [1]. Even though the traditional power grids are upgraded with advanced technologies, the system suffers electricity loss due to long distance transmission lines and distribution networks. In terms of the security and trustworthiness of the system, there are limitations. Cyber attacks or any kind of failure in the power grid may

cause power outrage over a broad area for long term. For example, a massive cyber attack happened in a Ukraine power plant in 2015 [2] resulted in power outage for hours in more than ten thousand homes and facilities. A malware called BlackEnergy installed on the control center computers was the cause of the attack. Slammer worm attack at Ohio power plant in 2003 and Stuxnet worm attack on control systems in 2010 are some among the other attacks reported. The requirement for massive power production and carbon emissions from these power generation are some other challenges for the traditional power grid system. Studies show that 40% of the overall carbon emissions in the United States are caused by power systems [3]. In order to get rid of these types of issues, modern power grid or smart grid was introduced.

A smart grid is a power transmission infrastructure that is integrated with information and communication technologies [4]. It can be represented as a bidirectional flow of electricity and communication data. Smart grid bi-directional property provides the customers and utilities an opportunity to monitor, predict, and control energy usage. Modern power grid design with principal characteristics was defined by U.S National Energy Technology, in 2007 [5]. Further in 2009, the U.S. Department of Energy restated the benefits and design features of smart grid [6]. Some of these features are described as follows:

- Customers are able to shift load, generate and store energy depending on the real-time prices and incentives. As the system is bidirectional, the customers can store the energy and sell the excess energy to the grid when the prices are high or as per the demand. Thus the system will be more efficient and can maintain a balance power supply through a demand response mechanism.

- Promoting all types of distributed energy resources such as solar, wind, geothermal, etc. with proper and flexible network architecture along with the centralised power generation. It helps to handle peak load and emergency situations. Integrating such a green resource power system contributes to social and economic benefits.

- Innovative products and services play an essential role in smart grids to achieve a cost effective and greener solution. Smart home appliances or intelligent electronic devices(IEDs) can be remotely controlled by authorised customers or providers.

- Smart grid provides power quality in terms of voltage flicker, volume and interruptions, to industrial or residential users with distinct power quality requirements.

- As the smart grid is a complicated system that controls various facilities and distributed energy resources, an efficient management mechanism is essential to optimise the asset utilisation and effortless operation and maintenance of the system. It must also reduce the investment/maintenance cost and power consumption.

- Eventually the entire smart grid operation must be volatile to interruptions such as attacks and natural disasters. It is to ensure the reliability of the power grid. Through proper communication local or national wide, the smart grid must be able to resist any attacks or damages. Sensing systems and automatic control system technologies are capable to "self-heal" the grid from unknown faults. Quick isolation of the suspicious grid component can also prevent the failure of the power supply to the nearby areas.

1.1.1 Smart Community Neighborhood

The traditional electricity infrastructure lacks proper mechanisms to handle massive and concentrated power consumption, especially with the arrival of electric vehicles (EV) and charging demands or peak over consumption [7]. In order to manage such situations, without entirely changing the structure of the power grid, smart grid projects integrate the renewable micro energy sources implemented at household or neighborhood levels. This makes the power consumers at the same time power producers as well, which gives an abstract name called prosumers. It is essential to identify potential prosumers and maintain tools to manage household or neighborhood energy

production and consumption. Therefore it is necessary to have efficient mechanisms and supporting techniques for collecting micro energy resources, maintaining community level power storage units, allowing trading energy and predicting the power consumption or production range between neighborhood houses or communities. The challenges here mentioned can be addressed in a smarter way with the help of ICT-based solutions. Since smart grids are decentralised data driven power management systems, they can contribute to the demand supply market of prosumers.

Smart Community Neighborhood (SmartNEM) project is a smart grid project which proposes a system that can utilise the advantage of advanced ICT solutions. As shown in Figure 1.1, the prosumers will produce the energy locally in households or neighborhoods with renewable energy resources and store it with the latest technologies. Prosumers can use the self-generated energy for their own requirements and excess energy produced can be sold back to the grid as per the requirement. The EV batteries can be used in households as an additional storage unit. As a result, these resources can help to balance consumption during peak hours and fluctuations, and also provide congestion relief. For ensuring a reliable and safe distribution system, operators and transmission system operators which are so called upstream suppliers demand information regarding the power consumption pattern and requirements for transmission, distribution and if needed, integrating the neighborhood resources.

The SmartNEM project mainly focuses on specific ICT areas for better smart grid functioning.

(1) **A high intensive data hub** is required as the data streams from each household in all communities and its corresponding resources will be potentially high. The data hub must be capable of providing sensible and real-time data insights.

(2) **Machine learning** can be used to predict energy consumption and production during peak hours or specific periods by households, neighborhoods, or communities.

(3) **Blockchain based technology** can be utilised to record the energy generated/traded by the prosumers and the energy stored

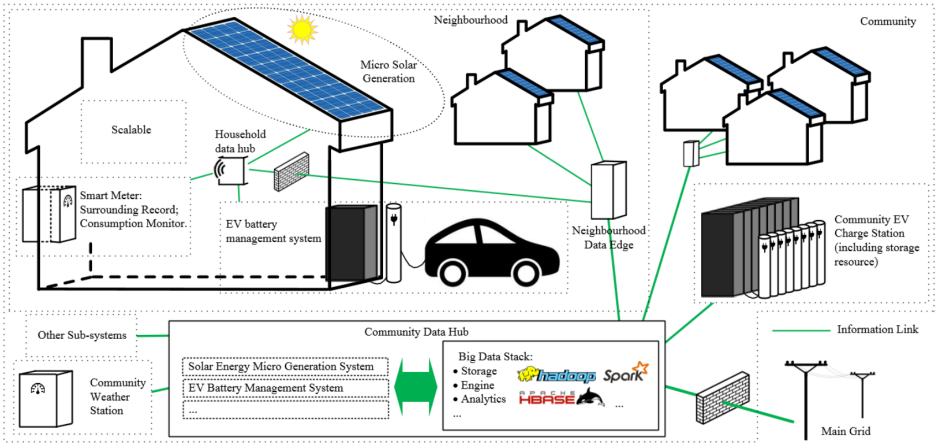


Figure 1.1: Smart Community Neighborhood (SmartNEM) [8]

in local or community storage resources. Each transaction in the blockchain will be transparent and secure. According to the National Institute of Science and Technology (NIST), blockchain is an immutable digital ledger system, which is designed in a distributed manner without a central authority [9]. In addition, digital assets cannot prove their ownership as easily as physical currency, crypto techniques are required to prove the ownership of digital assets. Asymmetric cryptography and hash function are the two techniques mainly used in Blockchain.

(4) **Data security and privacy** should be ensured in the entire process since the data from households contains sensitive data and patterns obtained from analysis of the data from the community level may be also exposed to several attacks. The trust in the system by prosumers is essential and it should maintain openness and active participation.

A self optimising and organising, secure and privacy preserving neighborhood energy management system which utilises the information-intensive data hub, blockchain and machine learning is the main purpose of the smart community neighborhood project (SmartNEM).

1.1.2 Relevance of data security and privacy in Smart grid

Data security stands for protecting the data from unauthorised users, manipulations and destruction. As per the comprehensive guideline for Smart Grid cyber security released by NIST, the main objectives of security in smart grids are availability, integrity and confidentiality. Availability ensures reliable access and use of information as per the requirement. Loss of data availability is a kind of interruption to data access and use. The integrity of data prevents modification of data by unauthorised users and systems. Confidentiality assures that access to data is restricted to authorized entities only.

Data privacy focuses on who is authorized to access the data and how the authorised can use the data after accessing it. As private data contains sensitive information, the responsible authorities must take necessary actions for the privacy protection of the data when dealing with it. Privacy handles the private data for analysis and at the same time ensures individual's privacy and their personally identifiable information. General Data Protection Regulation (GDPR) [10] is a regulation in the EU on data privacy for all within the European Union and the European Economic Area. The restrictions or essential requirements for exporting personal data to non-EU countries are also addressed in the GDPR.

Data plays a vital role in smart grid systems for obtaining energy usage patterns, peak hour consumption, and energy production from the household and community level. Data analysis will provide a clear awareness of the energy demands and it helps to improve the ability of the energy infrastructure.

1.2 Problem Description and Motivation

The conventional method of data analysis demands data transfer through the network to the destination where the actual data analysis takes place. It is network intensive and as a result, there are chances for security breaches. For example, data from households through smart meters contains private data which comes under sensitive data.

The data collected from smart meters can reveal information about energy usage in a detailed format. It will be a fine-grained meter reading per household within a short period. Daily activities and household appliance usage in each time period can be easily obtained from it and may cause serious privacy issues. In addition to this, load signatures have the potential to indicate whether you are home or not, and what routine you follow with the help of electronic device usage. Smart meters benefit to determine bills, promote energy conservation or awareness, indicate abnormal behaviour of appliances or usage of energy, illegal activities etc.

On the other hand, accessing the data by unauthorised users can lead to unethical activities, such as collecting information about famous people for gossiping and observing the pattern to know about valuable appliances and perfect time for burglary etc. Due to this reason, data owners may be reluctant to provide their data for further analysis. Not only in smart grid, but in general, due to the vulnerabilities that can occur, data owners are hesitant to provide their data for analysis in all others sectors. Thus as a common solution, it is highly recommended to have a secure and privacy-preserving system for data analytics, which can convince the data owners or organisations, to open up and allow data analysis on their own data.

The proposal of a generalised architecture for the secured data analysis will be the main objective of this thesis. Since moving the data through an external network is not secure, one approach is to move the code for analysis to where the data is available. For that, proper resources for data analysis should be there in the data owner's environment. We need to focus on two main areas to sort out the situation.

- Dealing with massive and diverse data, demands to have a proper storage mechanism and maintaining a big data analytic platform like Apache Hadoop, Spark, Storm, etc. These frameworks provide parallel computation on large datasets.

- Each and every dealing between the data users and data owners should be validated, transparent and secure enough. Blockchain technology guarantees that each transaction through it is se-

cured and tamper-proof. But blockchain has limitations to deal with large data and parallel computing.

The highlighted properties of big data analytics frameworks and blockchain technologies complement each other and motivate us to explore these features in detail to inspect possibilities for integration to achieve a secure and privacy driven data analytics framework. The architecture should inherit some of the key features mentioned above and at the same time provide strong decision making capability without compromising the security and privacy of data by utilising the technologies as shown in Figure 1.2

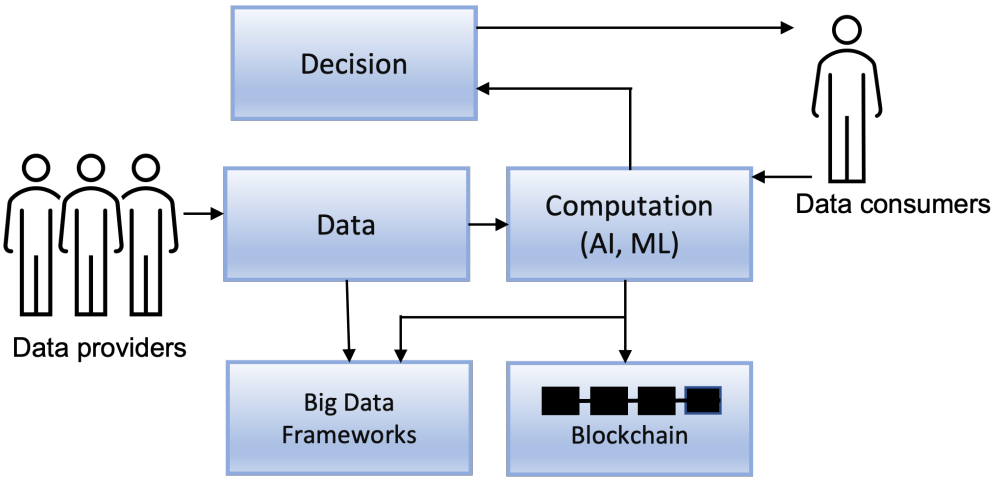


Figure 1.2: Motivation behind the research

Some of the works related to blockchain and big data framework are relevant and a detailed study was carried out as a part of the research. The SHDFS [11] is an architecture that distributed the NameNode functionalities using blockchain technology and eliminates the NameNode and secondary node concept. Another framework [12] proposed blockchain-based access control for reinforcing the security of big data platforms. In the architecture [13] uses a permisssioned Hyperledger blockchain based decentralized security system. This blockchain based access control system provides efficient access control management to asset owners on their dataset and precludes

data breaches. A scalable blockchain-based big data storage for distributed computing is known as HBasechainDB [14]. It was built on the Hadoop ecosystem, which inherits adequate big data processing and decentralization and immutability for the HBase database over blockchain. The decentralised management of private data, resolution of digital property, IoT communication etc with blockchain and big data systems were explained in detail in a literature review [15]. In this paper, blockchain and big data systems are integrated for different purposes. However, it is relevant to design a framework that can effectively utilise the benefits of blockchain and big data frameworks for secure data analytics.

1.3 Research Objective and Questions

The fundamental aim of this research is to design an architecture, which allows authorised users to analyse the dataset available in the data owner's environment, without moving the data across the network. Thus to obtain a system that is capable of protecting user's energy usage data, sensitive private information and providing a better approach for data collection, storage and processing.

Data privacy is essential for each and every one. The data from households will provide the entire pattern of usage of household appliances and routinely followed by the people. It can be utilised in two ways as mentioned in Section 1.2. One is for improving the efficiency of the grid by obtaining the demand and real time consumption of energy. On the other hand, it can be used for unethical purposes as well. Since private data is involved in these data streams produced from smart meters, the data owners may be reluctant to install and share their data with the responsible authorities as well. The main reason is the fear of data breaches. Thus it is important to ensure data privacy and to utilise the data ethically for the benefit of both the data owner and the system. It is needed to achieve a proper balance between data utility and privacy.

Proper records need to be maintained to verify for what purpose data is taken and how the data is used. Once the data is accessed, the responsibility for the personal data collected is obligatory and how

to handle the data acquiesce with existing consent is essential. From the original source of data, the entire change and flow of data should be traceable. It is also relevant to know which all users requested to access the data for analysis and what kind of information is taken out from the dataset. Thus accountability and traceability of the data are considered and it is essential to figure out a feasible mechanism.

The data owner should get full control of their own data and it makes it easier for them to trust and open up their dataset. It is their responsibility to manage data from the origin of data to the consumption for analysis stage. Rules and restrictions for usage directives/analysis and defining authorised users to access data also need to be ensured and controlled by data owners.

Blockchain technology maintains ledgers that contain blocks that record each and every transaction through it and big data analytic framework allows storage mechanism and processing of the data. Investigation of the features of big data analytics framework, blockchain and smart contracts that can be adapted as a part of the design with the above mentioned requirements are crucial and shows a solid direction to focus on.

Based on the objectives of the study, we formulated the following research questions.

(1) RQ1: What kind of mechanisms will achieve a balance between data utility and data privacy?

(2) RQ2: How to achieve accountability and traceability of the data?

(3) RQ3: What are the requirements for ensuring data ownership?

The approaches developed in this work would be useful to address challenges in peer to peer transaction systems for prosumer communities, power companies and storage providers.

1.4 Research publications

The research questions listed in the Section 1.3 were examined and solutions for the questions were answered with the research articles,

which are organised in five papers. All five papers are included in the thesis. The connection between research questions and papers is shown in Figure 1.3.

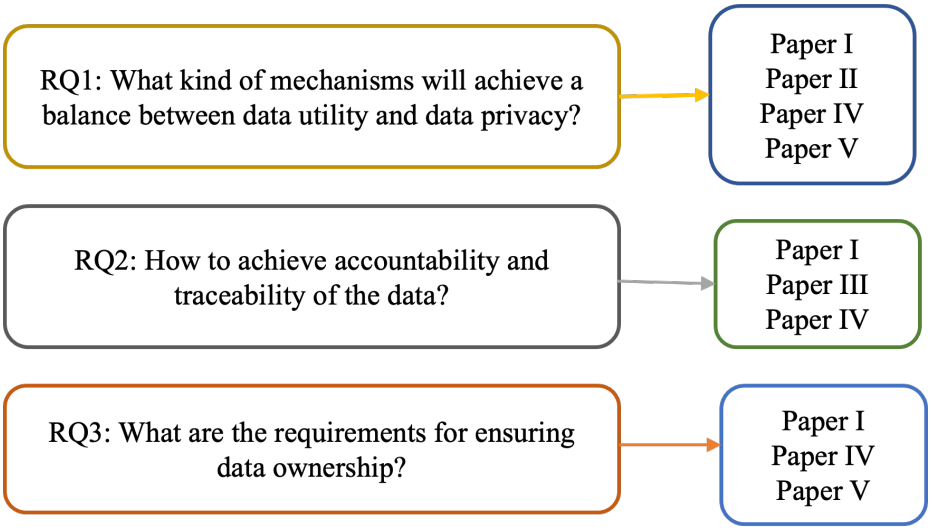


Figure 1.3: Relation between appended papers and research questions

Paper I proposed a framework TOTEM: Token for controlled computation [16] for secure and privacy driven data analytics. It integrates two technologies such as blockchain and big data analytics framework to achieve this. An entity called totem is introduced for the controlled computation in the framework. Paper II [17] is a proof of concept paper for the newly introduced components in the TOTEM framework, for controlled computation. It demonstrated the functioning of these components. Paper III [18] introduced the TOTEM defined SDKs design and architecture, implementation and demonstrate the SDK with basic rules, format and scenarios. Paper IV [7] proposes an extended TOTEM framework, which is capable of handling multi-provider scenarios compared to the single data provider framework of TOTEM. Paper V [19] shows the application of machine learning algorithms on the TOTEM framework to improve one of the core features of the framework and integrate federated learning for advanced learning scenarios.

The core part of the entire thesis is Paper I and all other papers

are linked to this paper. The connection between all the five papers is as given in Figure 1.4.

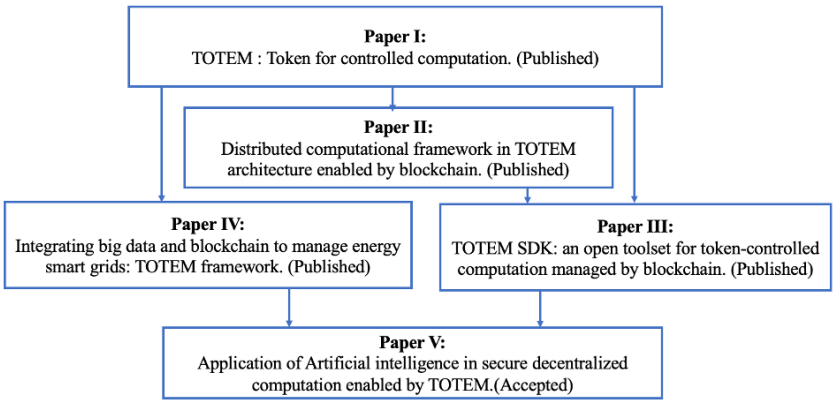


Figure 1.4: Connection between the Papers

A brief introduction of the research publications included in this thesis is given below.

- Paper I [16] "TOTEM : Token for controlled computation. " was published in the proceedings of "10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)". IEEE, 2019.

  This paper proposed a novel framework called TOTEM stands for "Token for controlled computation". This framework that integrates blockchain technologies and big data framework got patented on US Patent No.: US11,121,874 B2 in 2021, "Method for analyzing data using a blockchain, a data provider and a data customer therefor" [20]. TOTEM is a three layer architecture consisting of a blockchain consortium, a computation layer and a storage layer. The two main actors are the data provider who owns the data and the data consumers/users who get the opportunity to apply their computational code to the available data.

- Paper II [17] "Distributed computational framework in TOTEM architecture enabled by blockchain" was published in the pro-

ceedings of "15th International Conference on Computer Science and Education (ICCSE)". IEEE, 2020.

This paper mainly focuses on the TOTEM framework's computational layer. The computation layer located in the data owner's environment needs to monitor the value of the totem entity after each and every set of executions. A totem manager and updators are introduced in the TOTEM framework to achieve this. A proof of concept is presented in this paper. The implementation and testing of the components are taken place in the Hadoop environment. The totem manager is integrated with the master node and updators with the slave nodes for testing.

- Paper III [18] "TOTEM SDK: an open toolset for token controlled computation managed by blockchain." was published in the proceedings of "IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE)". IEEE, 2021

  TOTEM framework demands a TOTEM defined SDK in the architecture. The SDK with a set of rules and specified formats for the data consumers to write their code and submit for further execution in the data owner's environment. The TOTEM defined SDK designed to i. avoid malicious functions by restricting some specific functions and ii. to calculate totem value according to the complexity of the user code. The implementation of SDK with basic rules and calculation of the estimated totem value is carried out within this work.

- Paper IV [7] "Integrating big data and blockchain to manage energy smart grids—TOTEM framework" was published in the journal "Blockchain: Research and Applications". Elsevier.2022

  TOTEM is for analysing the data in a secure and privacy preserving manner. How to adapt the framework for Smart grid projects is explained and demonstrated in this paper. Also TOTEM framework in Paper I shows how a data consumer can execute their own code on dataset available in a data provider's environment. The possibility of analysing two data providers dataset and obtaining the combined result from two/multiple

data providers with the user submitted code is explained in this paper.

- Paper V [19] "Application of Artificial intelligence in secure decentralised computation enabled by TOTEM" is accepted in "IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE)". IEEE, 2022.

  A proposal for applying machine learning to obtain a better and more accurate totem estimated value is presented in this paper. The lack of data is a limitation for experiments thus proposing a way to collect data for training. Also introduced federated learning on the TOTEM framework and its applications in this work.

1.5 Thesis Outline

The remaining part of this thesis is organized as follows. Chapter 2 presents a background of the technologies and methodologies used in this thesis. Chapter 3 shows the main contributions from each paper presented in this thesis and Chapter 4 concludes the thesis and discusses the future directions. The five research papers that are part of this work are available at the end of the thesis.

Chapter 2

Background

This chapter presents the theoretical base for the papers included in the thesis. First of all, the concept of the Blockchain was briefly described with the benefits of smart contracts and an introduction to Hyperledger, which is permissioned blockchain and its features. After that a brief discussion on the big data systems and specifically about Hadoop and its MapReduce framework. Eventually, a brief description of Machine learning and federated learning is provided in this section.

2.1 Blockchain Technology

The Blockchain is a distributed ledger, that efficiently records each and every transaction through it [21]. Each of these transactions is transparent, immutable and tramper-proof. Unlike any centralised system, blockchain is a decentralised system with no central node to control or manage the entire system, but enables secure peer to peer communication through asymmetric encryption. For storing transactions in the blockchain, secure hash functions are used and each individual node can verify the transactions. Haber and Stornetta [22] used as a secure hash function to timestamp digital documents first in 1991 and later in 1992 Mekle tree was used to time stamp several documents into a single block. The same principle was used in Bitcoin [23] which is a public or permissionless blockchain introduced

by Satoshi Nakamoto in 2009.

The verified transactions will be added to a block and will be appended to an existing chain. Depending on the type of the blockchain this can be done by any of the nodes in the network. The consensus algorithm acts as an agreement to avoid conflict between the nodes on adding the same transaction blocks. By consensus algorithm, the valid node to append the new block will be chosen. Proof-based algorithm and voting-based algorithm are the two generally used algorithms [24].

Blockchain can be classified into two; public and private blockchains. In public or permissioned blockchains, the participants are anonymous and have no restrictions to join the network for the verification process [25]. Ethereum is a public blockchain. In private or permissioned blockchains, only authorised participants can join the network and selected nodes are responsible for the verification process. Sometimes it is restricted to nodes within an organisation or a consortium. Hyperledger is an example of a private blockchain.

2.1.1 Smart contracts

Smart contracts are programs that will be executed in a decentralized manner where a third party is not involved. It was proposed by Nick Szabo [26]. Smart contracts can be enabled in blockchain in a way that each transaction in blockchain will take place only if the conditions in the smart contracts are satisfied. The chances of deploying smart contracts that demand high computation in public blockchains are high due to the anonymous users in the network [23]. Since all the users in the network can participate in the validation process and if the deployed smart contract is with high execution time or infinite loop will result in large delays and denial of service attacks, which may lead to major issues in the network. To limit such complexity of computations certain mechanisms are introduced, for example, the 'gas' concept in Ethereum blockchain [27]. Using this concept, Ethereum limits operational complexity and restricts unrealistic or additional delays in the network.

2.1.2 Hyperledger

Hyperledger started as an open-source project in 2015 hosted by Linux Foundation and other companies, to enhance blockchain for enterprises. Hyperledger Fabric is a permissioned blockchain [28] that executes distributed applications written in programming languages such as Go, Java and Node.js. It tracks the execution sequence and adds the records into a replicated ledger data structure securely.

The three phases in the architecture are; the execution phase, ordering phase and validation phase. The two main parts of the architecture are chaincode and endorsement policy. Chaincode (smart contract) implements the application logic and performs during the execution phase. Some special chaincodes are deployed for managing or maintaining the parameter of the blockchain which are commonly known as system chaincodes. An endorsement policy takes action in the validation phase. It is a static library for transaction validation that can only be parameterized by the chaincodes. Authorised administrators are permitted to modify endorsement policies with the help of system management functions.

Hyperledger Fabric consists of a network of nodes. Membership Service Providers (MSP) provide an identity to each and every node participant in the network. The roles that nodes in the network are responsible, can be described as follows. When a client made a transaction proposal, nodes support the execution phase and broadcast the transaction for the ordering phase. Peers are responsible for transaction proposal execution and validating it. All peers maintain a ledger and only endorsing peers or endorsers specified in the policy are allowed to execute the corresponding transaction proposal. Ordering Service Nodes (OSN) or orderers are the nodes that perform ordering service.

2.2 Big Data

Big data refers to large datasets that are impossible to process with traditional methods or mechanisms. These large data cannot be interpreted, collected, and processed efficiently [29]. Big data feature is a set of V´s, which stands for volume, variety, velocity, veracity,

variability etc, [30] that represents the size of generated and stored data, nature of data, speed of data generated, data quality and unpredictability of data flow respectively [31]. Big data analysis refers to the mechanism used to analyse these large data sets. Hadoop [32], Spark[33], Strom [34] and MangoDB [35] are some of the approved big data analysis frameworks available.

2.2.1 Hadoop

Hadoop is an open-source framework that contains a cluster of computers for distributed processing of large datasets. The history of Hadoop starts in 2002, when Doug Cutting and Mike Cafarell worked on Apache Nutch project [36] which aimed to build a search engine system that can index one billion pages. Later they figured out that it is expensive to build and maintain such a system. So they were working to get a feasible solution that can solve the storage and processing of large data at a reasonable cost. In 2003 and 2004 Google published papers where the former describes the architecture of Google´s distributed file system(GFS) that solves storage of large datasets and latter introduce the MapReduce technique which solves the processing of large datasets. Hadoop is an Apache Software Foundation project for storing large dataset and processing it. [37]. The three main components of Hadoop are: Hadoop Distributed File System (HDFS), MapReduce and Yet Another Resource Negotiator(YARN). MapReduce and HDFS are briefly explained in the coming subsection, since it is relevant to our proposed architecture TOTEM. Hadoop is a master-slave architecture, where the master controls the slave nodes and slave nodes responsible for data processing.

2.2.1.1 MapReduce

MapReduce is a programming model for executing distributed computing. It is written in Java. The MapReduce algorithm consists of two main functions; Map and Reduce. The input data will be given to the map function and the output will be in a tuple form, a key/value pair. This key/value pair will be the input for the reduce function. According to the unique key value, reduce function will

group the tuples and combine them to obtain the output. The input, intermediate and final results of both functions are stored in files. A JobTracker and TaskTrackers are in the framework for managing the distributed parallel processing.[38]. The JobTracker is in the master node and the TaskTracker is in each and every slave node. JobTracker is responsible for checking the availability of resources, allocating resources as per the demands and scheduling tasks to the TaskTrackers. TaskTracker will compute the tasks, which are scheduled. A heartbeat signal which is basically status information will be sent to the JobTracker by every TaskTracker at regular intervals of time. It indicates that the corresponding node is active. If a heartbeat is not received from any particular node which a time period, then the JobTracker will reschedule the incomplete task assigned for that node to the next available TaskTracker. When a JobTracker is down then the entire process will be halted.

2.2.1.2 Hadoop distributed file system

Hadoop distributed file system (HDFS) maintains a master-slave architecture. The master or NameNode keeps the metadata of the file system. The actual files which are divided into fixed size blocks are stored in the slaves or DataNodes. The NameNode has the responsibility to map the blocks to the respective DataNode and DataNodes are responsible for the read-write operation in the file system and the creation, deletion and replication of blocks according to the instructions given by the NameNode. The secondary NameNode keeps the checkpoints of the file system metadata present in the NameNode. It is not a backup for the primary NameNode, and hence the NameNode cannot be replaced with the secondary NameNode.

2.3 Artificial Intelligence

Artificial Intelligence stands for the process of composing machines as intelligent as humans through proper learning and developing problem solving skills by algorithms artificially. Machine learning is a subset of artificial intelligence that deals with computational algorithms and

is designed to reproduce human intelligence by learning from the available dataset [39].

2.3.1 Machine learning

The principles in machine learning are adapted from various branches such as computer science, probability and statistics and artificial intelligence. Machine learning enables the machine to automatically learn from data, improve performance from past experiences, and make predictions with machine learning algorithms. The data is fed to these algorithms to train them, and on the basis of training, they build the model and perform a specific task. Machine learning can be applied in day-to-day life for various sections and fields such as entertainment, finance and medical applications.

Machine learning algorithms help to solve different business problems such us regression, classification, forecasting, clustering, etc. Based on the methods and way of learning, machine learning is divided into mainly four types, which are supervised machine learning, unsupervised machine learning, semi-supervised machine learning and reinforcement learning. Supervised learning is based on supervision [40] and train the machines using the labelled dataset and the machine predicts the output for the coming input data whereas in unsupervised learning there is no need for supervision and the machine is trained using the unlabeled dataset and the input sample alone is given for the learning process. Semi-supervised learning is a machine learning technique that uses large number of unlabeled data and less number of labeled data to train a model. In other words, for this process, both supervised and unsupervised method is required. Reinforcement learning works on a feedback-based process, in which an AI agent (A software component) automatically explore its surrounding by hitting and trail, taking action, learning from experiences, and improving its performance. In reinforcement learning, there is no labelled data like supervised learning, and agents learn from their experiences only.

Classification and regression are two types of supervised learning problems. Clustering technique comes under the category of unsupervised learning. Text document classifier will come under the category semi-supervised learning. Positive reinforcement learning

and negative reinforcement learning are also two types of reinforcement learning methods. In supervised learning regression algorithm is recommended for solving regression problems in which there is a linear relationship between input and output variables. Linear regression and Logistic regression are two such algorithms. Classification algorithm is recommended for solving the classification problems in which the output variable is categorical, and it is applied to test data to figure out which particular category it belongs to is called classification. Decision tree, Random forest, Support Vector Machines (SVM) and K-nearest neighbour are the algorithms for classification problems.

The limitations of conventional machine learning appear when it needs to handle large amount of raw data as input [41]. Human intervention, domain expertise and feature extraction are required to transform the raw data into a feature vector to input into the system to obtain or detect the classification category accordingly. Representation learning is a method that can take raw data as input and convert it into the corresponding form required for classification automatically. Deep learning is a representation learning method with multiple levels for representation. Simplified non-linear functions present in the design will transform from one level to the next higher level. Deep learning adopts artificial neural networks to execute complex computations on a large volume of data. Convolutional Neural Networks (CNNs), Long Short Term Memory Networks(LSTMs) and Recurrent Neural Networks (RNNs) are some of the popular deep learning algorithms.

2.3.2 Federated learning

In 2016, Google proposed the architecture of federated learning [42]. In federated learning, the available local data in edge devices or servers will be trained separately with the learning algorithm and the model thus obtained from each device will be aggregated to produce a global model. It ensures data privacy, since there is no data exchange between the devices and is not centralised.

Horizontal federated learning, vertical federated learning and federated transfer learning [43] are the three categories of federated

learning. Horizontal federated learning can be applied when the features of datasets are the same and overlapping. It selects the portion of data with the same features but different users for training by splitting the dataset horizontally. Sample size increases when the number of users increases, whereas vertical federated learning uses when the datasets with different features need to be trained and combine with the model for a global model. Federated transfer learning is a vertical federated learning enforced with an already trained model on a quite similar dataset, utilised for different problems. Federated computation is a MapReduce function for decentralized data that ensures privacy preserving aggregation. Federated averaging, FedAvg is the first federated learning algorithm developed by Google. FedProx, FedMa and FedOpt are some other algorithms commonly used.

Chapter 3

Contributions

In this chapter, the contributions based on the five papers included in this thesis are presented. First, we will discuss the main highlights and connections between the papers. The rest of the sections will summarise each paper included in this thesis.

3.1 Overview

This dissertation addresses the need for a new framework to facilitate a secure method for energy data analytics while preserving the privacy of each user in the network. Even though we have focused on the energy data, this solution is a global one and thus it can be recognised as a method for secure and privacy driven data analysis. The papers and highlights included in this thesis are:

Paper I [16] proposed a novel architecture that allows computation to be taken to the data, rather than moving the data to where the computation takes place. It is a three-layered architecture that includes a blockchain network and big data analytics tool with two main actors data owners, who owns the data and data consumers, who want to do computation on the data available to owners. An entity called totem is introduced to control the computational complexity or any malicious functions in the opcode submitted for analysis. A totem manager and updaters were also introduced in the computational layer of the architecture to coordinate the computation of the submitted

code until it finishes or the totem value exhausts. Each layer and new component were explained in the paper.

Paper II [17] implements the proof of concept of the computational layer in the TOTEM architecture proposed in Paper I. It will show the detailed workflow of the customized computational framework and implement it with the newly introduced components such as totem manager and updaters. The demonstration will show how a sample code execution and corresponding used totem calculation works in the framework. Paper II paper also describes the TOTEM defined SDK mentioned in Paper I. Hadoop cluster and the totem manager and updaters were the main components of the implementation on the computational layer.

Paper III [18] focuses on the TOTEM defined SDK based on Paper I and Paper II. It describes the architecture of TOTEM defined SDK and explains each layer of the SDK. This paper discloses how the SDK analyses the submitted codes with various layers in it and responds to the submissions. The implementation of TOTEM defined SDK with the basic set of rules and format is done in this work. The demonstration with sample codes was also performed in this study.

Paper IV [7] shows how we can adapt the TOTEM architecture proposed in Paper I to the smart community neighborhood project. An extended TOTEM architecture is proposed as a solution when the data consumer demands a combined result from multiple data providers as a part of data analysis. In the implementation, chaincode in the Hyperledger Fabric is used to manage the access to a remote resource and to the provisional computational resources as Docker containers that form the Hadoop cluster. The Hadoop cluster is responsible to do the required computation in an isolated environment with remote resources.

Paper V [19] introduced applications of artificial intelligence in the TOTEM architecture. We have mainly focused on two areas, 1) machine learning algorithm in TOTEM defined SDK for predicting the required or estimated totem value for a particular code to execute and 2) federated learning to obtain a global model from multiple providers present in the network. This paper is thus linked to Paper I, Paper III and Paper IV.

The contributions in each paper based on the research questions

and the technologies used are shown in Figure 3.1.

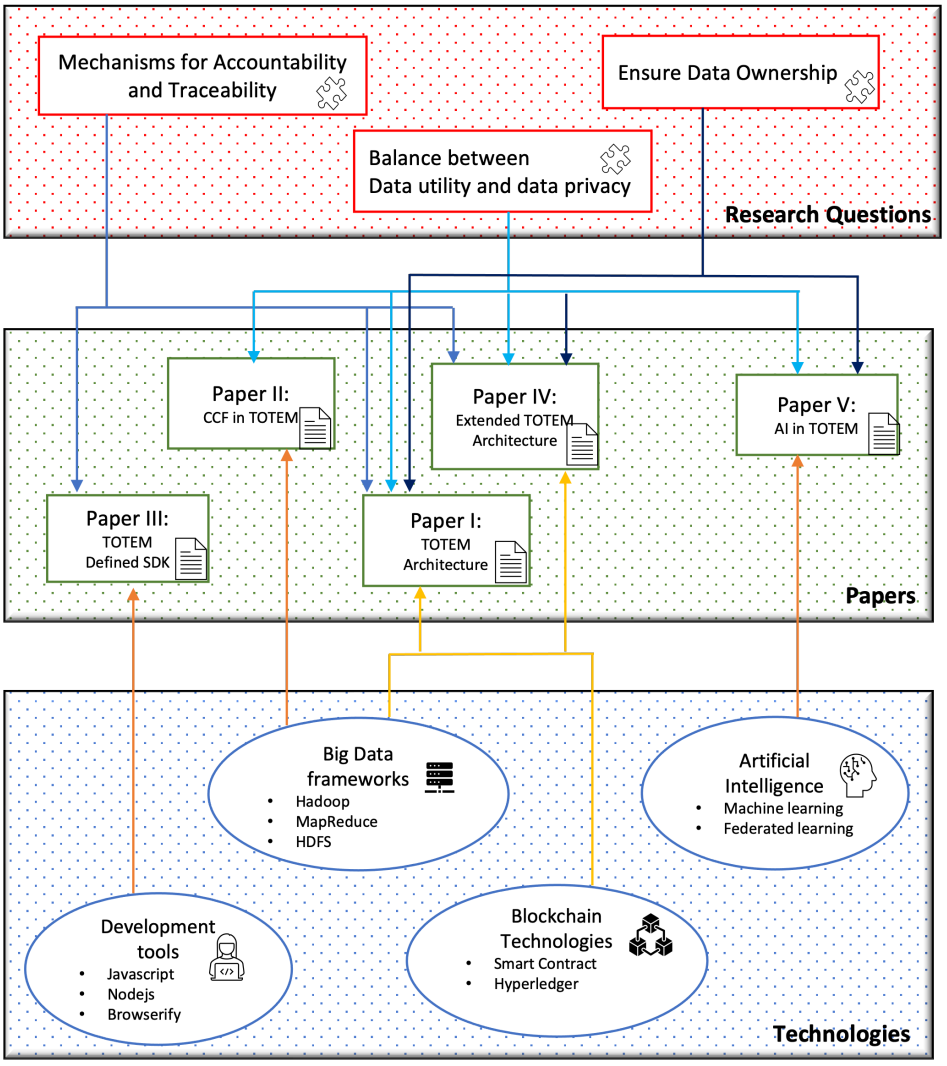


Figure 3.1: Contribution Summary

It shows which research questions are addressed in each of the papers and the technologies used to solve the problem statement. The papers are well connected since all the papers are based on the first paper, Paper I [16]. The relation between the papers is given in Figure 1.4 in chapter 1. The connection arrows from **Technologies** to

Papers shown in the Figure 3.1 discloses only the primarily focused technology in each of the papers. Papers I and IV talk about the TOTEM architecture, where the integration of two technologies is the objective, thus big data framework and Blockchain technology are the primary focus. Paper II aimed at the computational layer in TOTEM architecture which used the Hadoop framework for implementation and demonstration purposes. Paper III concentrated on the design and implementation of TOTEM defined SDK, which is again a part of the TOTEM architecture, thus required technology will be the development tools. Finally in Paper V, the technologies highlighted are machine learning techniques and federated learning. In Section **??**, a detailed description of how the research questions were answered with the papers is presented.

3.2 Paper I

TOTEM : Token for controlled computation. Integrating Blockchain with Big Data. [16]

This paper was published in the proceedings of "10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)". IEEE, 2019.

Nowadays, each one of us is dealing with large data around us. These diversified data streams keep on increasing exponentially. Therefore an authentic mechanism is required for the storage and process of these data with proper security and privacy measures. As we all are aware, in traditional data analysis methods the required data will be moved across the network for data analysis. In terms of security, while transferring such a large volume of data, these conventional methods are highly demanding. The data owners or data providers may be cautious and uncertain to provide their data in this situation. But these data from the data owners play a vital role in various useful trend identification and problem solving. At the same time it should be handled with appropriate methods to ensure data security and as well as it should not reveal the sensitive data associated with the dataset. In order to achieve this, the best option is to avoid

data transfer across any external networks. Big data frameworks and Blockchain technologies are two technologies, where their properties complement each other and can contribute a solution to the problem statement mentioned. Blockchain guarantees each transaction through it is secured and tamper-proof and Big data frameworks such as the Hadoop framework allows computation on large dataset efficiently.

A novel technology for sending computation to data is proposed in this paper. To achieve this, Blockchain technology is integrated with big data systems. We used permissioned private blockchain Hyperledger Fabric and Hadoop framework for the proposed architecture. The layers of TOTEM architecture are blockchain consortium, storage layer and computation layer as shown in Figure 3.2. The two main actors in the network are data providers or data owners, who owns the data and data consumer or data user who gets an opportunity to compute their own code on the data provider's dataset. Data providers are responsible to publish the corresponding metadata of their dataset and provide resources for computation according to the request received. Data providers in the network together agree and deploy smart contracts for the effortless functioning of the system. Smart contracts preliminary check the code submitted by the data consumer to avoid malicious codes. We have introduced a new entity called totem for controlled computation. Once the code is satisfactory and the data consumer is eligible with enough totem value in their account the code will be submitted for actual execution in the data provider's environment.

When the code is executed in the data providers environment, after each opcode execution the totem value in the account will be deducted corresponding to the complexity of the code executed until then. The execution continues until obtaining the final results or if the totem value exhausts. For coordinating the totem value calculation, a totem manager in the master node and totem updaters in each slave node of the Hadoop framework is introduced. The computational layer is also known as customized computational framework. The totem estimator table will maintain the value of the totem, depending on the data type and what kind of operation it demands. A detailed workflow on

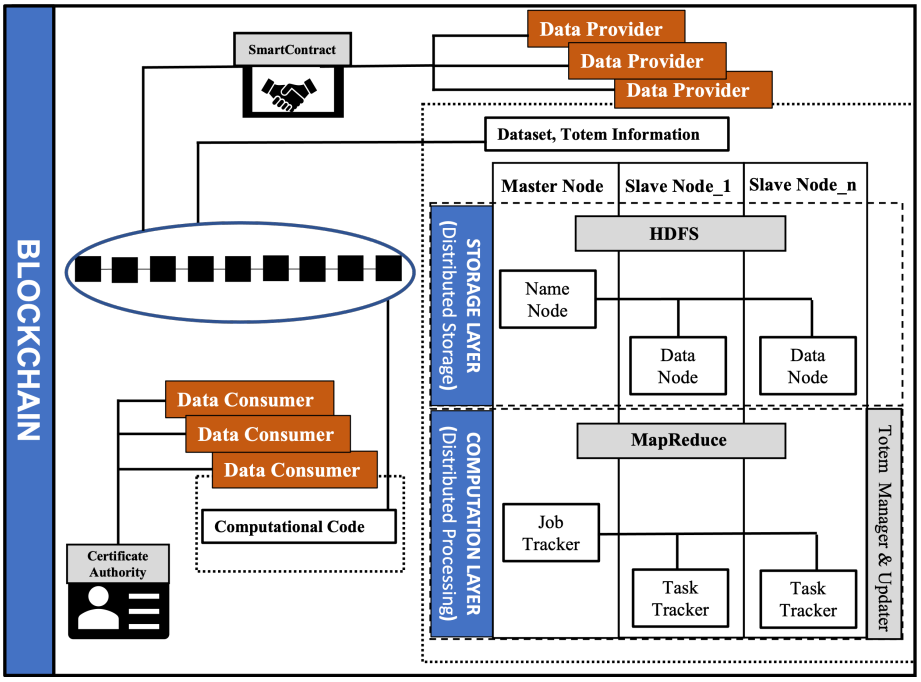


Figure 3.2: Totem Architecture [16]

the customized computational framework and sample calculation of totem value according to the provided code is also explained in the paper.

Newly introduced components such as totem manager and updaters are required to integrate with the current Hadoop system or corresponding big data systems accordingly. The code submitted through the framework should have a specific format and need to be pre-checked before the actual execution. Thus a TOTEM defined SDK is required for data consumers to write the code. This entire system results in a new method that enables the data providers to keep their own data secure and at the same time to be analyzed by the authorized data consumers through a proper secure mechanism. The transactions involved in the network such as analysis request, code pre-check, and continuous monitoring in between the execution will be recorded through the blockchain technologies. It helps to trace

each and every step that took place in the system. The data owner need not be reluctant anymore to provide the data, since the data will not be manipulated by any source since the computation itself will be controlled by the data providers themselves. As mentioned in the paper this project is unique and allows various organizations to open their data centers and create disruptive business models.

3.3 Paper II

Distributed computational framework in TOTEM architecture enabled by blockchain [17]

This paper was published in the proceedings of "15th International Conference on Computer Science and Education (ICCSE)". IEEE, 2020.

The proposed "TOTEM: Token for controlled computation" framework has the capability to bring changes in the traditional way of data analysis, by integrating the technologies such as big data frameworks and blockchain technologies. The authorised users are allowed to submit their code for execution on the data provider's dataset. The authorised users will be assigned totem values in their accounts according to the requirements. The customized computational framework in the architecture will execute the computations in a controlled manner with the help of the totem entity. The customized computational part contains a master-slave architecture with a totem manager and updaters in the master node and the slave nodes respectively. These components will coordinate the computation of the user code in the framework. Hence it is important to demonstrate that these newly introduced components in the customized computational framework are practically possible. In this paper, as a first step towards the implementation of the proposed framework, the proof of concept for the newly introduced totem manager and updater entities will be integrated with the big data systems [17].

The workflow of the TOTEM architecture which includes the functioning of Customized Computational Framework (CCF) is described

in [17] with the help of a flowchart. It will start with data consumer's enrollment to the blockchain, view and request for computation, code submission, preliminary check for code and totem value, actual computation with the particular data and finally publishing the result, if it exits gracefully. The TOTEM defined SDK is also introduced in this paper. The SDK will have a set of rules that are defined to meet the requirements of the TOTEM architecture. The rules and format will provide guidelines to data consumers for writing the codes. The TOTEM defined SDK is responsible to validate the data consumer's code and provide an estimated totem value required to execute the code. Once the totem requirements satisfy the SDK will pass the opcodes as an array of operations to the computational layer.

As a proof for the TOTEM customized computational framework concept, the main objective is the functioning of the totem manager and updaters. For the implementation, a Hadoop environment with a master node and two slave nodes is used in the computational framework. We have assumed that the execution of the submitted code will be carried out in the slave nodes. After each opcode execution happens the totem updaters in slave nodes will be notified. Then each totem update will inform the totem manager about the used totem and that will be deducted from the current totem value by the totem manager. It will continue until the totem value exhausts or when the computation finishes.

The implementation of the totem manager and updaters on the Hadoop environment was successfully incorporated. It is a part of the TOTEM architecture, thus we need to focus on the entire framework functioning. A basic demonstration of how customized computational framework calculates and updates the usage of the TOTEM value during the computation is presented in the paper. SDK and blockchain network should also be implemented as per the requirement for the fulfillment of the TOTEM architecture. In the present paper, there are a few assumptions made in order to make the framework functional, especially the array of operations given as input from SDK, which will be addressed in detail in the following paper.

3.4 Paper III

TOTEM SDK: an open toolset for token controlled computation managed by blockchain.[18]

This paper was published in the proceedings of "IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE)". IEEE, 2021

In order to submit the data consumer´s code for computation in a specific format and incorporate a mechanism to avoid the malicious functions in the code, the TOTEM framework required a TOTEM defined SDK. In this paper III, the main objective is to present the architectural design of the TOTEM defined SDK with three layers and basic functionalities involved in it. Also, we have implemented the Software Development Kit (SDK) for the TOTEM architecture with basic functionality and tested it with sample codes.

As mentioned in the motivation section, the SDK plays a vital role in the architecture of TOTEM by providing facilities for data consumers to submit their codes in a predefined format. The estimated totem value required to compute the code submitted should also be calculated in the SDK according to the complexity of the code submitted and the metadata received from the data provider. If the total totem available in the data consumers account is sufficient enough to proceed compared with the estimated cost, the SDK outputs a formatted code needed for the execution section. This code is an array of opcodes, which will be used as the input for the Customised Computational Framework(CCF).

This three layered architecture contains an SDK(main) layer, a controller layer and a handler layer. The SDK layer is the primary layer through which the user code and rules for the SDK are taken. The two components of the main layer are Line Reader Component (LRC) and Rule Processor Component (RPC). The Controller layer wraps all the components, each of which handles the control of a specific command type defined in the rules block. Labeled Type Controller (LTC) and Control Statement Controller (CSC) are samples of controlling

components in the layer. The handler's layer receives requests to handle execution or throw exceptions on mathematical computations. The defined rules of TOTEM SDK are given in a table in the paper. Also, the error functions and global variables associated with SDK are given in tables in the paper. The pseudo codes for rule processing and code submission are presented.

This paper discusses the relevance of having an extensible, well-structured, and easy-to-use SDK. The implemented SDK is not the final version but aimed to show the potential functionalities regarding sample predefined rules. The result will be an array of opcodes which can be the input for the computational layer of the TOTEM architecture. It will also estimate the totem value required for each specific code execution according to the complexity of the user submitted code.

3.5 Paper IV

Integrating big data and blockchain to manage energy smart grids—TOTEM framework [7]

This paper was published in the journal "Blockchain: Research and Applications". Elsevier.2022

Smart Community Neighborhood project (EnergiX project) is a smart grid project where the conventional end prosumers (consumers and producers) produce the energy from local sources and store it through eco-friendly mechanisms. It allows the end users to meet their own requirements with the energy produced and they can sell the excess back to the grid as per the requirement. Proper mechanisms are needed to coordinate the micro-energy sources, community-level power storage units, predict production and consumption, and allow the trading of energy between households, neighborhoods, and communities. The ICT-based solutions help to address the challenges that occur in this. The TOTEM architecture which can be defined as a data-driven decentralised framework can contribute to this system. This framework can be used in the smart community neighborhood

for enabling data hubs from different communities to open up their database for allowing data analysis from different communities or users without moving their data. Paper IV explains how to utilise TOTEM for the EnergiX project and also shows the scope of an extended version of the TOTEM which is dealing with the multi-provider architecture.

In this paper, the TOTEM architecture is explained with two scenarios related to the smart grid project. The first one is with a single data consumer (e.g: researcher) requesting a single data provider (e.g: C1DH: Community neighborhood-1 with a Data Hub) for data analysis on the data provider´s available dataset. It will work with the TOTEM workflow explained in [16]. When the data consumer wants to get a combined result from multiple data providers (e.g: C1DH and C2DH) the providers must communicate and both providers have to accept the request for data analysis in order to proceed. The totem value must be continuously monitored and updated by both providers. An extended TOTEM architecture is proposed in order to address the second scenario. We assumed that the opcode from the data consumer, R is submitted to data providers C1DH and C2DH through the TOTEM architecture network. The primary aim is to obtain a combined result from the data providers without sharing the individual data between the data providers.

The proposed extended TOTEM architecture solution consists of methods to allow each individual data provider to execute the given code, and then store the result in Hyperledger Fabric. It is achieved here by provisioning computational infrastructure with Docker containers in the data provider´s environment. The containers act as the Hadoop cluster on which the actual execution of the submitted code takes place. It is provisioned with the help of Ansible [44], which is open-source software that automates the process involved in IT infrastructure and application deployment.

A one-time code (OTC) from the blockchain will be given to the authorised data consumers. The obtained OTC along with the corresponding public key will be sent across to the data provider's resource

to authenticate and gain access. Once the authentication process completes, the data consumer is allowed to push Ansible commands to deploy the necessary infrastructure needed for running the opcode. The playbooks is responsible to provide the infrastructure, executing the computational code, and storing the result in Hyperledger Fabric. In the proposed architecture we recommend private data collections (PDCs) for storing the results.

For implementing the system, the following specific scenario is considered. Let us consider a Data consumer, R residing in Stavanger who wants to get a combined statistical result from two community data hubs, C1DH and C2DH, which reside in Spain and the Netherlands respectively. The system will allow each participant to comply with their respective region rules and permit authorized data consumers to compute their own code. The first stage is to set up the system on a single cluster residing in one region, followed by a demonstration of a distributed multi-cluster environment spanning two different regions. Azure [45] is a cloud service and it offers plenty of services, including the Azure Kubernetes Service (AKS). AKS offers a fully managed Kubernetes service and can be scaled up easily when needed. Hence AKS is utilised to provision multi-node Kubernetes cluster. The two methods implemented are:

- Method 1: Deploying Hyperledger Fabric on a single Kubernetes Cluster
- Method 2: Deploying Hyperledger Fabric network in a distributed cross-cluster environment.

The paper discussed on how the TOTEM architecture can be adapted to a smart community neighborhood project (EnergiX project) for data analysis in a secure manner. The extended version of the TOTEM architecture is also proposed as a solution for obtaining combined results from multiple data providers as requested by a data consumer. We have implemented the architecture as a part of the proof of concept. Chaincode in the Hyperledger Fabric is used to manage access to a remote resource in the implementation. Further

discussion on the improvements that can be done to the architecture for better working of the architecture is also presented.

3.6 Paper V

Application of Artificial intelligence in secure decentralised computation enabled by TOTEM [19]

This paper is accepted in "IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE)". IEEE, 2022.

The TOTEM defined SDK checks the format and rules defined and also estimate the totem value required for the execution of a particular code. For simple codes to execute, it is easy to predict the values while considering each opcode present in it. But when it comes to dealing with code execution on a given dataset the run time code complexity depends on the operands and the size and type of the dataset. Then we need a more accurate totem value estimation. How to utilise machine learning as a solution for the described problem and how the required dataset is collected are discussed in the paper. The application and advantages of incorporating federated learning in the TOTEM architecture are also discussed in this paper.

The paper opens up the TOTEM architecture in two directions with respect to the application of artificial intelligence. The relevant features of the user submitted code which follows the format and rules defined by the SDK can be extracted in a specified format to apply the prediction algorithm model. The model will be updated after each new code is evaluated and obtain the complexity. Also, the value of the totem can be calculated with the complexity obtained from the prediction. After each prediction, the dataset needs to be updated as well.

In the multi-provider TOTEM architecture which is the extended version of the TOTEM architecture proposed in paper IV, federated learning can be applied in order to address some of the challenges in the architecture. The dataset that is available in each data provider's

environment needs to be trained in order to obtain the trained model. Federated learning can be applied in this scenario to train the local model with locally available data and combine those local models to obtain a global model. Hence without sharing the dataset a global model is obtained. The TOTEM manages to calculate the run-time complexities and totem requirements for the entire functioning of the system. Federated learning can be applied to the TOTEM architecture in two ways.

- 1.Cross-silo federated learning where multiple data providers are involved in the process.
- 2. Cross-device federated learning where data users and data provider(s) are involved.

The paper discussed and highlighted two main directions and applications of artificial intelligence in TOTEM architecture for improving the features. One is to implement machine learning, training the data collected from previous estimations to estimate the totem values required for executing a specific code. By obtaining accurate totem value, a flawless execution of user code can be gained and also increase the trust in the architecture. Limitations are there in terms of available data for the initial prediction of the totem value. Thus we proposed the conventional estimation of the totem value for initial times and further Machine learning can be implemented as the system matures and when the architecture has enough data to predict the totem values. Another interesting direction is to apply federated learning on the data provider's dataset for obtaining a global model. However further detailed examination and implementation are needed as a scope for future work.

Chapter 4

Conclusions and Future Work

This chapter concludes the findings from the present research work. The conclusions are drawn by answering the research questions formulated in chapter 1, with the findings from the publications as the base. This chapter also discusses the future direction of the research and applications of the TOTEM architecture in various sectors.

4.1 Conclusions

The thesis has investigated the following research questions.

RQ 1. What kind of mechanisms will achieve a balance between data utility and data privacy?
Papers I, II, IV and V provide a solution for this research question with the help of the TOTEM architecture which was proposed in Paper I. The TOTEM architecture allows data consumers to analyse the data for decision making or finding the trends or patterns or proposing a better solution for the efficient working of a system. The data available in the data provider's environment will not be moved across the network for any computation and it will be handled only by the owner of the data. The computational code which will be submitted by the data consumer undergoes a preliminary check and is then executed with the data provider's full control. The TOTEM thus provides an opportunity to utilise the data and at the same time,

data privacy will be ensured as the data provider can decide on the privacy requirements of the available dataset. Paper II discusses the implementation of the computational layer of the TOTEM architecture, which is maintained by the data provider. In the computational layer, the data will be actually executed according to the code submitted by an authorised user. Continuous monitoring for a controlled computation is achieved. Paper IV is an extension of the TOTEM architecture that deals with providing combined results from multiple data providers. How to obtain the results without compromising the privacy of each data provider is addressed in this paper. Paper V explained how to utilise the data for obtaining machine learning global models for decision making while maintaining data privacy with federated learning along with the TOTEM architecture.

RQ 2. How to achieve accountability and traceability of the data?
Blockchain ensures a transparent and tamper-proof record of each transaction through it. In our TOTEM architecture, each and every request for the analysis of data will be processed with the decision from smart contracts which are deployed by the data provider's collective effort. Also during the execution of computational code, live monitoring of the totem value is managed by the totem manager and updaters. After each set of execution, it will be again checked with the smart contracts for more transparency between the data provider and data consumer. Thus accountability and traceability can be ensured with the mechanism proposed in the paper I and which is an extended version in paper IV. The totem estimate value calculation is also incorporated in the TOTEM defined SDK in paper III which controls the computation.

RQ 3. What are the requirements for ensuring data ownership?
In the TOTEM architecture, the data is owned by the data owner and will not be moved across the network by any means. Only the metadata will be displayed for the data consumer, to view and request for analysis. The data provider/data owner environment will have the resources to analyse the data. Thus the data is fully controlled by the data owner. The data owner has full access to the transactions made on the data through blockchain and will remain the sole

owner throughout the process. In order to avoid malicious functions that can cause data manipulation or denial of service with infinite loops, the computational code submitted by the data consumer in a specific format will be pre-checked before the execution. Papers I and IV explained the mechanisms used with a single provider and multiple providers respectively. Paper IV shows the method that can be applied if we need to combine the results from two data providers without sharing the data between the providers. Paper V discussed the application of privacy preserved artificial intelligence on the TOTEM architecture with federated learning.

The TOTEM architecture is a framework that can ensure secure and privacy preserved data analytics. Specifically, in paper IV we have stated how the energy data can be analyzed through the architecture. Most of the electricity providers nowadays use smart meters which enables them to handle the large volume of usage data which can be further used to study the usage patterns of various households and propose smart solutions. However, these kinds of data are not available to any external parties due to privacy concerns. The TOTEM framework is the way to go for such situations, which ensures maximum utilisation of the data and at the same time enables privacy and security.

The architecture is presented as a global solution for secured data analytics for any domain such as health, education, oil and gas etc. The framework is patented [20] and hence it proves the novelty and shows the relevance of the framework we proposed.

4.2 Future Work

In the present research work, the TOTEM architecture is proposed and some of the essential components such as Customised Computational Framework (CCF) and TOTEM defined SDK with basic features are implemented. However, the full implementation is not in the scope of the present research. For the same reason the performance analysis of the system is also not included in the present

study. In order to set up the full framework and make it completely work, it is needed to add all the required functionalities, and rules to perform the system integration. As a first step towards commercialisation of the TOTEM framework, the framework got patented on US Patent No.: US11,121,874 B2 in 2021, "Method for analyzing data using a blockchain, a data provider and a data customer therefor" [20].

Even though the present research was focused on the energy data, the new architecture benefits all business models which make use of large organizational data for decision making. There are many, but only a few are listed here.

Health domain: In the health industry there are many instances where the government wants to have key data from the hospitals for evaluating the medical situation and further planning the actions at a national level. However, it is of great concern for private hospitals to hand over such confidential information to external networks. In such a case, the TOTEM architecture enables the hospitals to have a full track of the data without moving the data out of the hospital datacentres, at the same time letting the government handle the data within their premises.

Renewable Energy Industry: In a wind farm set up there are hundred to two hundred wind turbines working under different environmental conditions. The operational behaviour of a wind turbine is highly nonlinear and a global database for wind farm operation will benefit designers to optimize the efficiency of future wind farms. However, due to confidentiality, such data is only available to the consortium that owns the wind farm and its partners. A secured and privacy preserved data analysis framework will open up these data for vast industries and accelerate the green energy shift at a faster pace.

Education: Educational sector needed continuous diagnosis and corrections in order to reform the policies best for the students. This is achieved these days by individual surveys and studies at various universities and other educational institutions. The process is too slow and it requires a lot of nondisclosure agreements with the sur-

veyors and the institutions. The outcome of these surveys will be then studied by an expert committee that advises the government on educational reformations. The entire process can take many years and by the time the survey details are obsolete. Our present framework is a solution that can help the institutions to share their data with the surveyors and enable the decision making process efficient and faster. Machine learning algorithms linked to the data survey can even enable the decision making smart and accurate.

References

[1] Damir Novosel Michael I. Henderson and Mariesa L. Crow. "Electric Power Grid Modernization Trends,Challenges, and Opportunities." In: (November 2017).

[2] "Cyber-attack against Ukrainian critical infrastructure, Alert (IR-ALERT-H-16-056-01) The Industrial Control Systems Cyber Emergency Response Team (ICS-CERT), Department of Homeland Security, Washington, DC https://www.cisa.gov/uscert/ics/alerts/IR-ALERT-H-16-056-01." In: (2016).

[3] "NaturalGas.org, "Natural gas and the environment," available at: http://www.naturalgas.org/environment/naturalgas.asp." In: (2010).

[4] Jing Liu, Yang Xiao, Shuhui Li, Wei Liang, and CL Philip Chen. "Cyber security and privacy issues in smart grids." In: *IEEE Communications surveys & tutorials* 14.4 (2012), pp. 981–997.

[5] "U.S. NETL, "A systems view of the modern grid," White Paper,available at: http://www.smartgrid.gov/white papers." In: (2007).

[6] "U.S. DOE, "Smart grid system report," White Paper, available at: http://www.oe.energy.gov/SGSRMain 090707 lowres.pdf." In: (2009).

[7] Dhanya Therese Jose, Jørgen Holme, Antorweep Chakravorty, and Chunming Rong. "Integrating big data and blockchain to manage energy smart grids—TOTEM framework." In: *Blockchain: Research and Applications* 3.3 (2022), p. 100081. ISSN: 2096-7209. DOI: https://doi.org/10.1016/j.bcra.2022.100081. URL: https://www.sciencedirect.com/science/article/pii/S2096720922000227.

[8] "https://www.uis.no/en/research/data-centered-and-secure-computing-dscomputing." In: (2017).

[9] Mingli Wu, Kun Wang, Xiaoqin Cai, Song Guo, Minyi Guo, and Chunming Rong. "A Comprehensive Survey of Blockchain: From Theory to IoT Applications and Beyond." In: *IEEE Internet of Things Journal* 6.5 (2019), pp. 8114–8154. DOI: 10.1109/JIOT.2019.2922538.

[10] "GDPR https://gdpr-info.eu/." In: (2020).

[11] Deepa S Kumar and M Abdul Rahman. "Simplified HDFS architecture with blockchain distribution of metadata." In: *International Journal of Applied Engineering Research* 12.21 (2017), pp. 11374–11382.

[12] A Outchakoucht Jp Leroy H Es-Samaali, Nn Van, and R Nakagawa T Tanouchi S Kodama. "A Blockchain-based Access Control for Big Data." In: *Journal of Computer Networks and Communications* 5 (2017), pp. 137–147.

[13] Uchi Ugobame Uchibeke, Kevin A Schneider, Sara Hosseinzadeh Kassani, and Ralph Deters. "Blockchain access control ecosystem for big data security." In: (2018), pp. 1373–1378.

[14] Manuj Subhankar Sahoo and Pallav Kumar Baruah. "HBasechain DB-a scalable blockchain framework on hadoop ecosystem." In: (2018), pp. 18–29.

[15] Elena Karafiloski and Anastas Mishev. "Blockchain solutions for big data challenges: A literature review." In: (2017), pp. 763–768. DOI: 10.1109/EUROCON.2017.8011213.

[16] Dhanya Therese Jose, Antorweep Chakravorty, and Chunming Rong. "TOTEM : Token for controlled computation: Integrating Blockchain with Big Data." In: (2019), pp. 1–7. DOI: 10.1109/ICCCNT45670.2019.8944855.

[17] Dhanya Therese Jose, Antorweep Chakravorty, and Chunming Rong. "Distributed computational framework in TOTEM architecture enabled by blockchain." In: (2020), pp. 83–88. DOI: 10.1109/ICCSE49874.2020.9201683.

[18] Behfar Behzad, Dhanya Therese Jose, Antorweep Chakravorty, and Chunming Rong. "TOTEM SDK: an open toolset for token controlled computation managed by blockchain." In: (2021), pp. 1–8. DOI: 10.1109/CSDE53843.2021.9718489.

[19] Dhanya Therese Jose, Chunming Rong, and Antorweep Chakravorty. "Application of Artificial intelligence in secure decentralised computation enabled by TOTEM." In: *IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE).IEEE* (2022).

[20] Dhanya Therese Jose, Antorweep Chakravorty, and Chunming Rong. *Method for analyzing data using a blockchain, a data provider and a data customer therefor*. US Patent 11,121,874. Sept. 2021.

[21] Karim R Lakhani and M Iansiti. "The truth about blockchain." In: *Harvard Business Review* 95.1 (2017), pp. 119–127.

[22] S Haber and WS Stornetta. "How to Time-Stamp a Digital Document, Menezes AJ, Vanstone SA (eds) Advances in Cryptology-CRYPTO'90. CRYPTO 1990." In: *Lecture Notes in Computer Science* 537 (1991).

[23] Satoshi Nakamoto. "Bitcoin: A peer-to-peer electronic cash system Bitcoin: A Peer-to-Peer Electronic Cash System." In: *Bitcoin. org. Disponible en https://bitcoin. org/en/bitcoin-paper* (2009).

[24] Giang-Truong Nguyen and Kyungbaek Kim. "A survey about consensus algorithms used in blockchain." In: *Journal of Information processing systems* 14.1 (2018), pp. 101–128.

[25] Gareth W Peters and Efstathios Panayi. "Understanding modern banking ledgers through blockchain technologies: Future of transaction processing and smart contracts on the internet of money." In: (2016), pp. 239–278.

[26] Nick Szabo et al. "Smart contracts." In: (1994).

[27] Vitalik Buterin et al. "A next-generation smart contract and decentralized application platform." In: *white paper* 3.37 (2014), pp. 2–1.

[28] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, Srinivasan Muralidharan, Chet Murthy, Binh Nguyen, Manish Sethi, Gari Singh, Keith Smith, Alessandro Sorniotti, Chrysoula Stathakopoulou, Marko Vukolić, Sharon Weed Cocco, and Jason Yellick. "Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains." In: EuroSys '18 (2018). DOI: 10.1145/3190508.3190538. URL: https://doi.org/10.1145/3190508.3190538.

[29] Min Chen, Shiwen Mao, and Yunhao Liu. "Big data: A survey." In: *Mobile networks and applications* 19.2 (2014), pp. 171–209.

[30] Judith Hurwitz, Alan Nugent, Fern Halper, Marcia Kaufman, et al. "Big data for dummies." In: 336 (2013).

[31] Parth Chandarana and M Vijayalakshmi. "Big data analytics frameworks." In: (2014), pp. 430–434.

[32] Jeffrey Dean and Sanjay Ghemawat. "MapReduce: a flexible data processing tool." In: *Communications of the ACM* 53.1 (2010), pp. 72–77.

[33] Salman Salloum, Ruslan Dautov, Xiaojun Chen, Patrick Xiaogang Peng, and Joshua Zhexue Huang. "Big data analytics on Apache Spark." In: *International Journal of Data Science and Analytics* 1.3 (2016), pp. 145–164.

[34] Muhammad Hussain Iqbal, Tariq Rahim Soomro, et al. "Big data analysis: Apache storm perspective." In: *International journal of computer trends and technology* 19.1 (2015), pp. 9–14.

[35] Shakuntala Gupta Edward and Navin Sabharwal. "Mongodb architecture." In: (2015), pp. 95–157.

[36] Richa Vasuja, Ayesha Bhandralia, and Kanika Chuchra. "Daemons of Hadoop: An Overview." In: *International Journal of Engineering Research and Technology* (2018), pp. 2278–0181.

[37] Dhruba Borthakur. "The hadoop distributed file system: Architecture and design." In: *Hadoop Project Website* 11.2007 (2007), p. 21.

[38] Vinod Kumar Vavilapalli, Arun C Murthy, Chris Douglas, Sharad Agarwal, Mahadev Konar, Robert Evans, Thomas Graves, Jason Lowe, Hitesh Shah, Siddharth Seth, et al. "Apache hadoop yarn: Yet another resource negotiator." In: (2013), pp. 1–16.

[39] Issam El Naqa and Martin J Murphy. "What is machine learning?" In: (2015), pp. 3–11.

[40] Rich Caruana and Alexandru Niculescu-Mizil. "An empirical comparison of supervised learning algorithms." In: (2006), pp. 161–168.

[41] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." In: *nature* 521.7553 (2015), pp. 436–444.

[42] Jakub Konečn, H Brendan McMahan, Daniel Ramage, and Peter Richtárik. "Federated optimization: Distributed machine learning for on-device intelligence." In: *arXiv:1610.02527* (2016).

[43] Qiang Yang, Yang Liu, Yong Cheng, Yan Kang, Tianjian Chen, and Han Yu. "Federated learning." In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 13.3 (2019), pp. 1–207.

[44] "Ansible https://www.ansible.com/overview/it-automation." In: *Ansible* (2022).

[45] "Azure https://azure.microsoft.com/en-us/." In: *Azure* (2022).

Paper I:
TOTEM : Token for controlled computation

TOTEM : Token for controlled computation

Dhanya Therese Jose [1], **Antorweep Chakravorty**[1], **Chunming Rong**[1]

[1] Department of Electrical Engineering and Computer Science, University of Stavanger

Abstract:

Conventional methods for data analysis demand data be moved across networks for analysis. Along with it being highly network intensive, data owners are also reluctant to provide their data due to security and privacy breaches. In order to overcome these issues, we proposed TOTEM: Token for controlled computation, that combines both blockchain technologies and big data systems. The objective is to allow computation to be taken to the data, rather than moving the data to the computation. A third-party user uses the TOTEM SDK to create a MapReduce code for computation, which will be executed within the data owner's environment. This computational system prevents the execution of any malicious functions in the user code, by putting constraints on computational operations. A pre-defined totem will be assigned to authorised users, based on their computational needs. A smart contract performs pre-checks on user-submitted code and associated totem value, by using a totem estimator to determine the required totem for executing the given code. A Totem manager and updater are introduced to coordinate computation of user code until it exits gracefully, or the assigned totem gets exhausted. The TOTEM project ensures data security by allowing organisations to open their data centers and allow disruptive business models.

1 Introduction

In this present world, we need to deal with large and diverse data streams that keeps increasing exponentially. As a result, we need to handle the storage mechanism, security and privacy issues of these data. Conventional methods for data analysis demand that the data is moved across networks for analysis. However, transferring such large datasets becomes highly demanding. Moreover, many of the data providers are reluctant to furnish their data, due to privacy and security issues. One approach is to move the computational code to the data, without transferring the data to an external network. Hadoop framework allows computation on large datasets through parallel computing. Blockchain technology guarantees that each transaction through it is secured and tamper-proof. On the other hand, blockchain is limited while handling large data and parallel computations. However, the properties of these two technologies can complement each other which would enable a new way of computing upon large datasets without the need for moving the data across the network.

In this paper, we propose a novel architecture combining the Hyperledger Fabric blockchain and the Apache Hadoop framework, in which the computation is taken to the data, rather than the conventional way of moving the data to the computation. This architecture ensures data privacy and security, as third-party computations are executed within the environment of the data owners. In order to prevent the execution of any malicious functions in the user code, a new entity called TOTEM: Token for controlled computation, is introduced. The totem is a computational system, that puts constraints on computational operations. It limits the number of executable operation codes (opcodes) in a computational code, by allowing users to associate predefined totem values with their code. This computational code would be written in a custom MapReduce SDK specific to the TOTEM project. A monitoring system keeps track of opcodes performed and used totem values. A gating system prevents further opcodes in a computational code once the associated totem values gets exhausted.

The rest of the paper is structured as follows: Section 2, provides an overview of Blockchain technology, Smart contracts, Big data

systems, Hyperledger Fabric and Hadoop. Section 3 discusses the existing works related to the integration of blockchain and big data systems. The proposed architecture is explained in Section 4. A detailed explanation on major components of the architecture is also provided. Finally, Section 5 concludes the paper.

2 Background

2.1 Blockchain Technology

The blockchain is an open distributed ledger, which records transactions very efficiently [1]. The transactions recorded in blockchain are tamper-proof and transparent to all users in the network. It is a peer-to-peer communication between nodes, therefore no central node controls the entire network. Each node individually verifies all transactions directly. Cryptographically secure hash functions are implemented to store the transactions in the blockchain. In 1991, Haber and Stornetta used the cryptographically secured chain of blocks to time stamp digital documents [2] and in 1992, they used Merkle tree for timestamping of several documents into one block. However, this concept got attention in 2009, when Satoshi Nakamoto, introduced Bitcoin [3]. Bitcoin is a completely decentralized, peer-to-peer permissionless blockchain.

Blockchain can be classified into two types, namely public and private blockchains [4]. The public blockchain is permissionless blockchains where the participants are anonymous and there are no restrictions in joining the network for the verification process. Bitcoin and Ethereum [5] are examples of the public or permissionless blockchain. In private or permissioned blockchain, permission to join the network is restricted to users within an organization or group of organizations. Selected nodes by the blockchain consortium can participate in the verification process. Hyperledger Fabric[1] and Ripple[2] are examples for permissioned or private blockchain. As blockchains are decentralized, when a transaction is proposed, its validity can be

[1] `https://www.hyperledger.org/`
[2] `https://ripple.com/`

verified by any node in the network. These nodes add the transactions to a block and append it to the existing chain. However, there is a chance that more than one node can come up with a new block to append to the blockchain at the same time. In order to avoid this situation, an agreement should be made between the nodes about the node that is chosen to append a new block. This agreement is called the consensus algorithm which can be either proof based or voting based [6].

2.2 Smart Contracts

One of the biggest advantages of blockchain is that it can enable smart contracts. The concept of the smart contract was first proposed by Nick Szabo [7]. Blockchain-based smart contracts can be any kind of computer program that will be executed in a decentralized manner, i.e., without a third party. Each transaction in a blockchain will occur only if the conditions in the smart contracts are met. In public blockchain, as it is anonymous, there is a chance that anyone can deploy smart contracts which requires high computation. Particularly in Proof of Work (PoW) consensus algorithm [3], as all users in the network participate in validation, if the smart contract requires high execution time or it contains an infinite loop, large delays might be added to the network. This kind of denial-of-service (DOS) attacks could be catastrophic to the whole network. Therefore, it is important to limit the complexity of computations in smart contracts. For instance, Ethereum introduced the 'gas' concept to tie up with execution complexity of smart contract to financial limitation [5]. To deploy a smart contract to the network, the user needs sufficient amount of gas. When each operation in a smart contract executes in the network, it uses the available gas in the user´s account. If a smart contract exhausts the available gas of users, it stops executing. This way, Ethereum introduces limits on operational complexity and thereby preventing unrealistic or additional delays to the network.

2.3 Big Data System

Big data represents the datasets that are very large to be efficiently interpreted, collected, managed, and processed, using traditional mechanisms [8]. The main features of big data are volume, variety, velocity and veracity [9]. These features denote the size of generated and stored data, type and nature of the data, the speed at which the data is generated and processed and the data quality and the data value respectively [10]. Big data analytics stands for the method or strategy of analyzing large volumes of data. Some of the popular frameworks for big data analytics are Hadoop [11] [12], Spark[3], MongoDB[4], Strom[5], Cassandra[6], Neo4j[7], etc. Among these Hadoop is one of the leading open source frameworks, which can run on-premises or in the cloud. However, each framework has its own advantages and disadvantages.

2.4 Hyperledger Fabric

Hyperledger Fabric is a private and permissioned blockchain [13] [14]. Members of this network are enrolled through a trusted Membership Service Provider (MSP). It supports different MSPs and ledger data can be stored in multiple formats. Channels in Hyperledger Fabric allow a group of participants to create a separate ledger of transactions. If a group of participants forms a channel, then only those participants will have copies of the ledger of that channel. A ledger contains two components called world state and transaction log. The world state represents the state of the ledger at a given point of time. The transaction log is the update history for the world state. Smart contracts in Hyperledger Fabric are written in chaincode and it can be invoked by an external application when it needs to interact with the ledger. Mostly the chaincode interacts with the world state and not the transaction log. The chaincode can be implemented in Go or Node.js programming languages.

[3] `https://spark.apache.org/`
[4] `https://www.mongodb.com/`
[5] `http://storm.apache.org/`
[6] `http://cassandra.apache.org/`
[7] `https://neo4j.com/`

2.5 Hadoop

Hadoop is a framework for distributed processing of large datasets with clusters of computers. Hadoop is an open-source implementation based on a programming model called MapReduce [11]. Hadoop framework consists of two main layers such as the Hadoop Distributed File System (HDFS) [12] and distributed processing (MapReduce).

HDFS has a master-slave architecture, where the master or the NameNode maintains the file system metadata. Files will split into fixed-size blocks and are stored in the slave or DataNodes. DataNodes will periodically send heartbeats to the NameNode to indicate the node is active. Mapping of blocks to the DataNodes is determined by the NameNode. DataNodes are responsible for read-write operations in the file system. Based on the instructions given by the NameNode, DataNodes are also responsible for the creation, deletion and replication of blocks. Secondary NameNode in HDFS is the node which keeps checkpoints of the file system metadata on the NameNode. In the MapReduce distributed processing framework, there are two functions/tasks that are performed: map and reduce. The map function takes input data and converts it into granular structures. The output of the map function will be tuples with a key/value pairs. Reduce function takes the output from a map function as inputs and combines and groups the tuples into a set with a unique key value. Input and output of both functions will be stored in the file system. The MapReduce framework contains a single JobTracker and TaskTrackers [15]. The master or JobTracker will, monitor resource availability, allocate resources and schedule tasks for the slaves or TaskTrackers. The slaves or TaskTrackers compute the tasks given by the JobTracker, and provide task status information to the master at regular intervals. If one of the TaskTrackers goes down, the JobTracker will reschedule the failed tasks to the next available TaskTracker and if the JobTracker itself goes down the entire process will halt.

3 Related Work

In [16], various possibilities of using blockchain on big data systems such as decentralized management for private data, IoT communication, resolution of digital property, and public institutions were discussed. A blockchain-based access control framework for reinforcing the security of big data platforms is proposed in [17]. The framework achieved objectives such as user-driven, transparency, lightweightness, fine-granularity, pseudonymity and unlinkability. However additional critical issues were emerged while adopting blockchain technology to handle access control functions. The SHDFS architecture [18] for big data storage eliminates the concept of NameNode and secondary NameNode. The functionalities of the NameNode being distributed using blockchain technology. Metadata creation and blockchain placement were also implemented and tested in a cluster of nodes.

A blockchain access control ecosystem that gives asset owners complete guarantee to effectively manage access control of large datasets and prevent data breaches is proposed in [19]. The architecture uses a decentralized security system based on the private and permissioned Hyperledger blockchain. The challenges associated with traditional and centralized access control is solved by blockchain technology. The blockchain ensures the data transparency, traceability, secure data sharing, auditability and data self-sovereignty for the owner. HBasechainDB [20], is a scalable blockchain-based big data storage for distributed computing. It adds decentralization and immutability to the HBase database through blockchain. As HBasechainDB was built on the Hadoop ecosystem, it inherits efficient big data processing. For organizations which have Hadoop ecosystem based business logic, HBasechainDB makes it easy to accommodate blockchain. In [21], the common security problems associated with Kerberos [22] are discussed. Especially the authentication mechanism using Kerberos makes big data systems exposed to many security risks and vulnerabilities. Therefore, it recommends to utilise the advantages of blockchain and hardens the security systems, including distributed authentication and no single point failure of the big data system.

In the above-mentioned literature, integration of blockchain and big data system is proposed or implemented for various purposes.

However, it is important to establish a framework which can effectively utilise the advantages of both the technologies. In the proposed framework we ensure the security and privacy of data to the data owner and provide a controlled computation facility for third-parties to execute their own code on the data.

4 Proposed Architecture

A novel technology for shifting computation to data is proposed. This approach will integrate blockchain technologies with big data systems. The architecture is shown in Figure 1. integrates Hyperledger Fabric blockchain and the Hadoop framework.

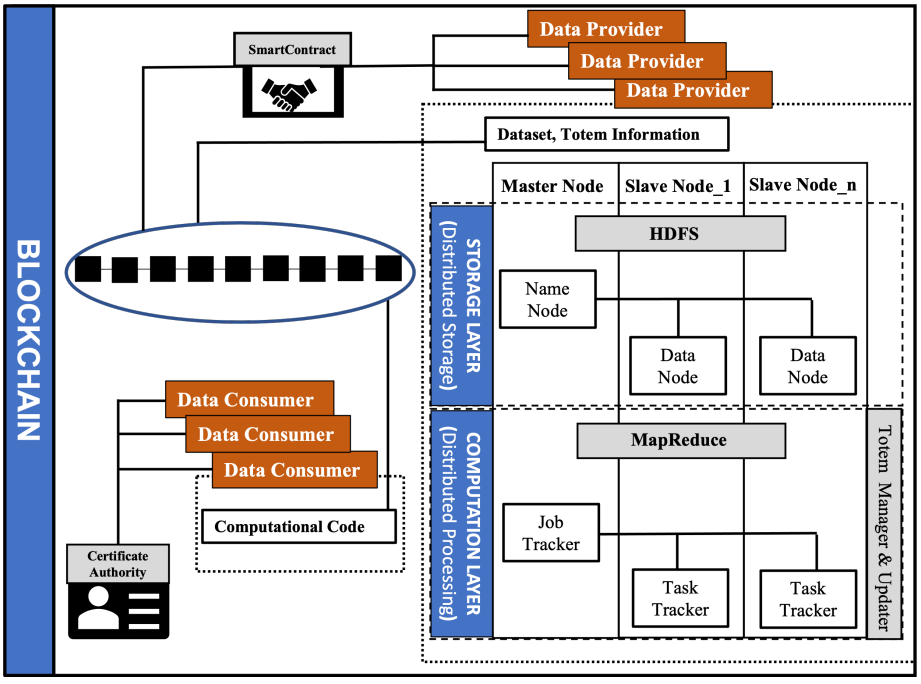


Figure 1: Proposed architecture

It consists of a blockchain consortium, storage layer and a computation layer. The major actors in the blockchain consortium are data providers and data consumers. Data providers own the data and publishes the meta-data regarding the dataset to the blockchain

and connects their big data resources for computation. Authorized data consumers will get the opportunity to execute their own code on the available dataset. Data providers are responsible to deploy the smart contract which evaluates the code submitted by the data consumer. The smart contract preliminary monitors the malicious functions or infinite loops in the consumer code and limits computational complexity. The monitoring must be standardized and as blockchain consortium contains multiple data providers it should be a collective effort from all the data providers. The proper validation of the submitted code is a challenge and we propose a new concept called token for controlled computation (totem), for limiting the computational complexity and preventing malicious functions in the code. Additionally, the architecture also contains a storage layer that represents the Hadoop distributed file system, where the actual data is stored. The computation layer of the architecture consists of the MapReduce framework for distributed processing. Finally, a totem manager and updater is introduced, that manages the usage of totem according to the execution of each opcode.

4.1 TOTEM :Token for controlled computation

Totem is an entity which controls the computational complexity or malicious functions of the opcode given by the data consumer and thereby prevents delays on the network. Totem in the present architecture is similar to the gas concept in Ethereum [7]. A predefined totem will be assigned to the authorized consumer based on their computational needs, along with the access key. During the execution time, depending on the computational complexity of the consumer´s computational code, the value of totem assigned to them reduces. The computation will continue until it exits gracefully or when the totem value exhausts. The usage of the totem will be monitored continuously during the execution.

The workflow of the proposed architecture is shown in Figure 2. The workflow shows the entire process starting from enrolling a data consumer to finally obtaining relevant results after the computation of their code. The various steps in the workflow are:

1) An authorized data consumer which can be an entity, or an

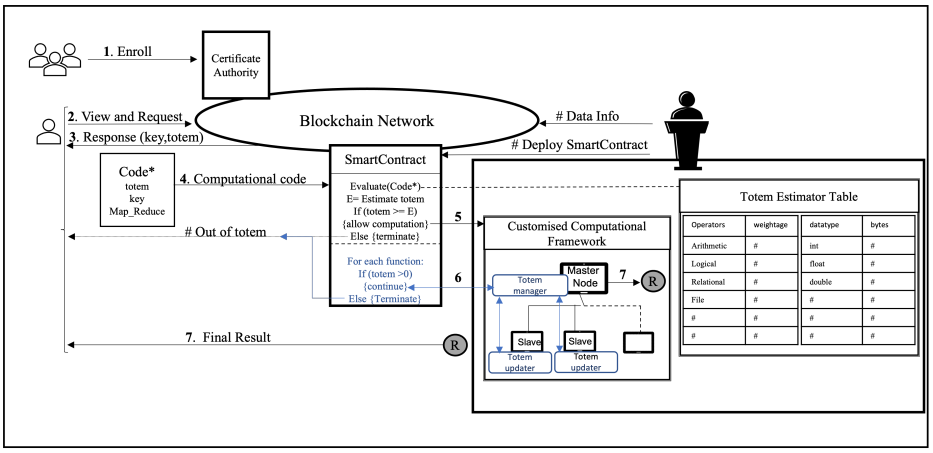


Figure 2: Workflow of proposed architecture

organization can perform computation on an available dataset under the blockchain consortium. The data consumer is authorized by registering or enrolling to the blockchain consortium. (2) When the data consumer is authenticated, he can view and request the corresponding meta-data and the computation facility provided by the data providers. Based on metadata and their computational requirements, a data consumer can estimate their totem needs. Depending on the network, the consumer could acquire the required totem from the blockchain consortium. (3) If the request is valid, the data consumer will get a response which contains the key to access the data and the totem for computation. (4) The data consumer will provide the computational code which he wants to perform on the requested data, along with the provided key and acquired totem. It is required that the submitted computational code must adhere to custom MapReduce SDK for the TOTEM project. The smart contract which was deployed by the data provider will perform a preliminary check of the code. It evaluates whether the provided totem satisfies the computational needs of the provided code. The smart contract uses this information and a totem estimator table to find the estimate of required totem value. A detailed description of the totem estimator table is given in Section 4.3. If the requirement is less than or equal to the available totem, it will allow the computation

on the data.

(5) If the preliminary check passes, the actual computation will take place in a Customized Computational Framework (CCF). As the name indicates, it is customized and resembles the Hadoop architecture. In addition to the master and slave nodes, the new computational framework contains a totem manager and a totem updater. A detailed description of the CCF is given in Section 4.2. For each opcode execution, the slave node will report the performed opcode to the totem updater in the slave node, which in turns send a response to the totem manager in the master node. The totem manager will calculate the available balance based on the reports from all the updaters and send the status back to all the updaters. When the updater gets the response from the manager, it will send a signal to the slave node for performing the next opcode. This will repeat until the map or reduce function execution completes. Once a chain of map or reduce function is executed in the slave nodes, totem updaters will update to the totem manager. In addition to that, whenever the available totem becomes zero, the totem manager will immediately report an "Out of totem" status to the blockchain and send a signal to master node to immediately stop the whole execution. (6) After the execution of each map or reduce function, the totem manager sends the information regarding the used totem for that set of opcodes to the blockchain network as a transaction. Following, the smart contract which is already deployed in blockchain consortium will check whether the consumer still has sufficient totem for further computation. If it has enough balance, then control goes to the master node to perform further functions. (7) If the required computation on the particular dataset finishes or totem is exhausted, the corresponding result will be available for the data consumer.

The two main components of the data provider in the proposed architecture are the customized computational framework and totem estimator table. A detailed explanation of these two components and the smart contract for preliminary check is given below.

4.2 Customized Computational Framework

Data provider provides the data and also a platform for computation. The Customized Computational Framework (CCF) used in the present architecture resembles the Hadoop architecture, which contains the master and slave nodes. Apart from that, in the present architecture, we are extending a totem manager in the master node and a totem updater in slave nodes.

The workflow diagram for the Customized Computational Framework is shown in Figure 3. After the preliminary check for the estimate of totem required for the execution of the requested code, the control goes to the CCF Master node. The master node which is responsible to assign the task for the slave nodes starts to send the required map/reduce functions and the data.

The various steps in the CCF are explained below, (1.1) From blockchain, available totem and the estimated totem to perform the entire task will be given to the totem manager. The totem manager is responsible to calculate the total usage of the totem in between the executions and give instructions to master node whether to continue the execution further or not. (1.2) Simultaneously, the actual computational code to be performed on the requested dataset will be sent to the master node. In the workflow, the code contains three map functions M1, M2 and M3, and two reduce functions R1 and R2. These functions must be in a custom MapReduce format, which is specified in the TOTEM project. (2.1) The master node will give instructions to all the slave nodes, to perform the first map function M1. (2.2) Simultaneously, the master node will inform the totem manager that the instruction to perform the first function is given to all slave nodes. (3.1) Totem manager sends the available totem information to all the totem updaters. (3.2) The totem updaters will have the information regarding the function to perform, M1 and the datatype of the dataset from the respective slave nodes where the updater is associated with.

(4) Totem updater will estimate the totem required to perform the function M1, on the dataset. If the estimate is within the available totem scope, step 5 else step 15. (5) An instruction is sent to the corresponding slave node to start the execution of the function. (6)

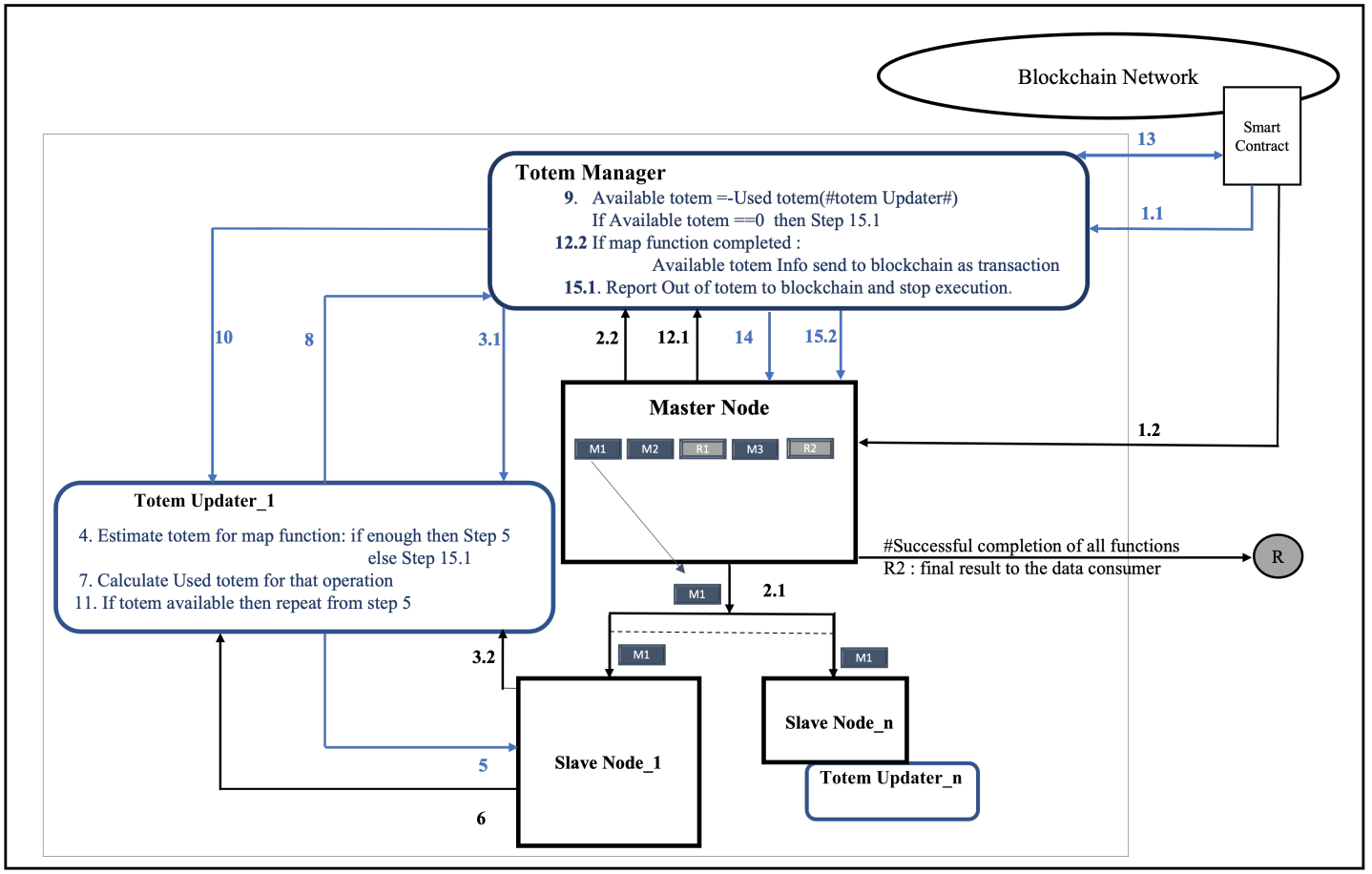


Figure 3: Customised Computational Framework

A function contains, a set of operators, which need to be executed. Operators can be of any type such as arithmetic, logical, relational, assignment, bitwise, etc. After preforming each opcode, the slave node will inform the updater. (7) The updater will calculate the used totem for performing that opcode on the given datatype. (8) The totem used for current opcode will be updated to the totem manager. (9) In the totem manager, the updaters update the used totem and calculate the available balance. If the available totem becomes zero, the control goes to step 15, which is to stop the execution immediately. (10) After calculating the available totem, it will be reported to the totem updaters.

(11) If the totem updater receives the signal from totem manager, which indicates that there is enough totem for further execution, these updaters will again send a signal to the corresponding slave nodes to continue the execution. It then repeats step 5 to step 11 until the completion of that function or termination due to lack of totem. (12.1) The master node signals the totem manager that the execution of that particular function completed successfully. (12.2) Simultaneously, the totem manager updates the blockchain with the available totem. (13) While sending the available totem as a transaction to the blockchain, the smart contract will double check if the totem balance is empty or not. If not, the control goes back to totem manager and continues from step 14, else data consumer will get the information "Out of Totem" and hence the connection is closed. (14) Otherwise, it will signal the master node to release the next function in the code. It can be a map or a reduce function. Here in the workflow, it is M2, map function. Thus, the entire process will repeat from step 2.1 to step 13, the same as for the M1 function. (15.1) When the available totem is empty, the following two actions will be taken place simultaneously. Send an update to blockchain: "Out of Totem" and (15.2) send a signal to master node to stop the entire execution immediately. If the final result from last reduce function, R2, is available, the result will be published to the data consumer through the blockchain network and the connection will be closed.

4.3 Totem Estimator Table

Totem is an entity which controls the computational complexity of the opcode given by the data consumer. There should be some standard totem value for each type of opcodes to be performed. These values should be the same for all the data providers present in the blockchain consortium. The table which describes the totem requirement for performing each opcode is known as the Totem Estimator Table. The smart contract, which is executed before the required computation, will evaluate an estimate of the totem according to the submitted code by the user. For this estimation, the smart contract uses the information in the totem estimator table. This estimate of the totem is a function of various opcodes in the code and data types. The estimator table should include all the supported opcodes on the data. For example, the addition of two values with datatype double would require more totem compared to int datatype. The datatype eventually shows the bytes required to store the particular data. Hence, it can be represented as a function of operator and bytes required.

The set of opcodes is given as O:

$$\{o_1, o_2, o_3, \ldots o_o\},$$

which include the arithmetic, logical, relational, assignment, bitwise, etc. Each operator in O has a weightage, W:

$$\{w_1, w_2, w_3, \ldots w_o\}$$

corresponding to the computational complexity, which is further used for totem estimation. The set of supported datatypes is given as D:

$$\{d_1, d_2, d_3, \ldots d_d\}$$

and the corresponding bytes required for each data type as B:

$$\{b_1, b_2, b_3, \ldots b_d\}$$

.

A general formula for estimating totem required can be written as:

$$\text{Estimated totem} = \sum_{i=1ton} w(o_i) * b(d_i)$$

Table 1: DATASET

Date	Cid	Bill
25/02/2019	01	100
25/02/2019	02	200
25/02/2019	03	150
26/02/2019	02	100
26/02/2019	03	50
27/02/2019	02	100

where, n represents the number of opcodes in the computational function,

$$o_i \in 0 \quad \text{and} \quad d_i \in D.$$

4.4 Smart contract: Eval(code*)

Eval(code*) on the smart contract performs a preliminary check to find the estimate of totem required for the given code by the data consumer. A sample calculation is shown below to demonstrate the actual execution of Eval(code*) according to the equation defined for estimated totem. The dataset in Table 1 has three columns which contain the date, customer id, Cid with int datatype and bill amount (Bill) with the double datatype.

The code in Figure 4 contains a map and reduce function. The reduce function should have the input as a (key, [value]) pair, key represent each unique Cid and value represent an array of all Bills corresponding to each Cid. Reduce function here is, to sum up, all elements in each array.

Function	*Pseudo code*
Map	*f() {find (this.C_id, this.Bill)}*
Reduce	*f(key,values) {array.sum(values)}*

Figure 4: SAMPLE TOTEM ESTIMATOR TABLE

The totem estimator table is shown in Figure 5, which gives information regarding the weightage of each operator. It includes all the valid operators that can be used during execution. It also contains the number of bytes required for each data type.

Totem Estimator Table

Operator		*Datatype*	
Operator	Weightage	Datatype	Bytes
Arithmetic:+	2	int	4
Assignment:#	1	double	8
Relational:#	1	date	3
Read	1	#	#
#	#	#	#

Figure 5: SAMPLE TOTEM ESTIMATOR TABLE

The estimation determines the required totem which is demonstrated in the Eval(Code*) table in Figure 6. According to the example, map function will read each row in the dataset, i.e., a total of 6 rows. The weightage of the read operator and the datatypes involved in a row determines the required totem for that particular statement to execute. Then it will have each tuple as (Cid, Bill), that requires only read operations per row.

The output of the map function will be sorted and shuffled by the Hadoop MapReduce framework. The input of reducer function will be a tuple (key, [value]) pair.

Reading each input requires same effort as mentioned in the map function. After each read, the sum of Bills related to each Cid will be executed. Cid: 1 has only one Bill and thus no '+' operator required. Cid: 2 has three Bills and thus two '+' operator are required. Therefore, the required totem will be twice the weightage of addition operation times bytes required to store the datatype of Bill (double). Finally, Cid: 3 has only two bills, hence one '+' operator . The sum

Eval(Code*)						
Map function	*n*	*Operator*	$w(o_j)$	*Data type* (d_l)	$b(d_l)$	$w(o_j)$* $b(d_l)$
6 rows: Repeat (n=1,2, 3,) 6 times	1	Read a row	1	int, double, date	3+4+ 8	15
n=1 to 6	6*15					90
Reduce function						
	7	Read first row	1	int, double	4+8	12
	8	Read second row	1	int, double	4+ 3*8	28
	9, 10	Arithmetic: '+' (three elements)	2, 2	double	8, 8	32
	11	Read third row	1	int, double	4+ 2*8	20
	12	Arithmetic: '+' (two elements)	2	double	8	16
n=7 to12						108
$\sum_{i=1 \text{ to } n} w(o_i) \times b(d_i) = 90+108= 198$						

Figure 6: SAMPLE TOTEM ESTIMATOR TABLE

of required totem for each operator mentioned here will give the total totem required to execute the entire code*.

5 Conclusion

In the presented paper, a new architecture for secured and privacy driven big data analytics is proposed. The new architecture combines both blockchain technologies and big data systems. The architecture contains blockchain consortium, storage layer and computational layer. It also aims to create custom SDK for creating MapReduce based computation code specific to the platform. It enables the data provider to keep the data secure and at the same time to be analyzed by the authorized data consumers. The computation takes place where the data is located, rather than moving the data to the code. A new entity called totem is introduced for preventing any malicious functions in the computational code. The architecture uses a totem estimator table for estimating the required totem for executing a given code. In the computation framework, the totem value is continuously monitored in between executions. The execution continues until the computation is over or totem gets exhausted. This TOTEM project is unique as it allows multiple organizations to open their data centers and create disruptive business models. In addition to that, the proposed architecture would be realized and investigated as an open source project.

References

[1] Karim R Lakhani and M Iansiti. "The truth about blockchain." In: *Harvard Business Review* 95.1 (2017), pp. 119–127.

[2] S Haber and WS Stornetta. "How to Time-Stamp a Digital Document, Menezes AJ, Vanstone SA (eds) Advances in Cryptology-CRYPTO'90. CRYPTO 1990." In: *Lecture Notes in Computer Science* 537 (1991).

[3] Satoshi Nakamoto. “Bitcoin: A peer-to-peer electronic cash system Bitcoin: A Peer-to-Peer Electronic Cash System.” In: *Bitcoin. org. Disponible en https://bitcoin. org/en/bitcoin-paper* (2009).

[4] Gareth W Peters and Efstathios Panayi. “Understanding modern banking ledgers through blockchain technologies: Future of transaction processing and smart contracts on the internet of money.” In: (2016), pp. 239–278.

[5] Vitalik Buterin et al. “A next-generation smart contract and decentralized application platform.” In: *white paper* 3.37 (2014), pp. 2–1.

[6] Giang-Truong Nguyen and Kyungbaek Kim. “A survey about consensus algorithms used in blockchain.” In: *Journal of Information processing systems* 14.1 (2018), pp. 101–128.

[7] Nick Szabo et al. “Smart contracts.” In: (1994).

[8] Min Chen, Shiwen Mao, and Yunhao Liu. “Big data: A survey.” In: *Mobile networks and applications* 19.2 (2014), pp. 171–209.

[9] Judith Hurwitz, Alan Nugent, Fern Halper, Marcia Kaufman, et al. “Big data for dummies.” In: 336 (2013).

[10] Parth Chandarana and M Vijayalakshmi. “Big data analytics frameworks.” In: (2014), pp. 430–434.

[11] Jeffrey Dean and Sanjay Ghemawat. “MapReduce: a flexible data processing tool.” In: *Communications of the ACM* 53.1 (2010), pp. 72–77.

[12] Dhruba Borthakur. “The hadoop distributed file system: Architecture and design.” In: *Hadoop Project Website* 11.2007 (2007), p. 21.

[13] “Hyperledger Fabric Documentation Release 1.4.” In: *Hyperledger* (2019).

[14] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, Srinivasan Muralidharan, Chet Murthy, Binh Nguyen, Manish Sethi, Gari Singh, Keith Smith, Alessandro Sorniotti, Chrysoula Stathakopoulou, Marko Vukolić, Sharon Weed Cocco, and Jason Yellick. "Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains." In: EuroSys '18 (2018). DOI: 10.1145/3190508.3190538. URL: https://doi.org/10.1145/3190508.3190538.

[15] Vinod Kumar Vavilapalli, Arun C Murthy, Chris Douglas, Sharad Agarwal, Mahadev Konar, Robert Evans, Thomas Graves, Jason Lowe, Hitesh Shah, Siddharth Seth, et al. "Apache hadoop yarn: Yet another resource negotiator." In: (2013), pp. 1–16.

[16] Elena Karafiloski and Anastas Mishev. "Blockchain solutions for big data challenges: A literature review." In: (2017), pp. 763–768. DOI: 10.1109/EUROCON.2017.8011213.

[17] A Outchakoucht Jp Leroy H Es-Samaali, Nn Van, and R Nakagawa T Tanouchi S Kodama. "A Blockchain-based Access Control for Big Data." In: *Journal of Computer Networks and Communications* 5 (2017), pp. 137–147.

[18] Deepa S Kumar and M Abdul Rahman. "Simplified HDFS architecture with blockchain distribution of metadata." In: *International Journal of Applied Engineering Research* 12.21 (2017), pp. 11374–11382.

[19] Uchi Ugobame Uchibeke, Kevin A Schneider, Sara Hosseinzadeh Kassani, and Ralph Deters. "Blockchain access control ecosystem for big data security." In: (2018), pp. 1373–1378.

[20] Manuj Subhankar Sahoo and Pallav Kumar Baruah. "HBasechain DB-a scalable blockchain framework on hadoop ecosystem." In: (2018), pp. 18–29.

[21] Mithun Kankal and Pramod Patil. "An adaptive authentication based on blockchain for bigdata hadoop framework." In: *Int J Eng Tech* 5 (2019), pp. 89–94.

[22] Tom White. *Hadoop: The definitive guide.* " O'Reilly Media, Inc.", 2012.

Paper II: Distributed computational framework in TOTEM architecture enabled by blockchain

Distributed computational framework in TOTEM architecture enabled by blockchain

Dhanya Therese Jose [1], **Antorweep Chakravorty**[1], **Chunming Rong**[1]

[1] Department of Electrical Engineering and Computer Science, University of Stavanger

Abstract:
TOTEM: Token for controlled computation is a newly proposed framework that integrates blockchain with big data systems. This framework allows users to send the computational code towards the data and analyse instead of the conventional method of data analysis which required data to be sent across the network. The framework provides a TOTEM defined SDK for users to code according to the standard format and rules defined. TOTEM execute the user code within the data owner's environment. Proper mechanisms are there to manage and prevent the malicious functions that may occur in the user code by defining a proper set of rules and putting constraints on the user computational code by introducing totem value. Authorised users are assigned with a predefined totem value and when the user submits the code the smart contract will estimate totem value for performing the user-submitted code. If the totem value is sufficient for the execution it will pass the code to the computational part of the framework. The customized computational part of the framework consists of a big data system such as Apache Hadoop and a new system that have a totem manager and updaters to coordinate the controlled computation of the user code. In this paper, we discuss and demonstrate the proof of concept for the customized computational part of the framework.

1 Introduction

TOTEM: Token for controlled computation [1] is a proposed framework, which changes the conventional methods of data analysis by integrating blockchain with big data systems. In the proposed framework instead of data, computational code will be sent across the network and computation will take place in the data owner´s environment itself. TOTEM allows authorized users to submit computational codes through a TOTEM defined SDK. Depends on the computational needs a totem value will be assigned to authorized user´s computational code. Malicious functions in the submitted code will be detected and the execution of such functions will be prevented by putting constraints on computational operations by the totem value. A smart contract validates the user code and using totem estimator table it will estimate the totem value required to execute the user-submitted code. The actual computation happens on the customized computational part of the framework. The customized computational part contains a master-slave architecture. In addition to that, it contains a totem manager and updaters in the master node and the slave nodes respectively, which coordinate the computation of the user code until the entire execution completes, or when the assigned totem for the particular user gets exhausted. In this paper, we aim at taking the first step towards the implementation of the proposed framework by presenting the proof of concept for the newly introduced totem manager and updaters part which will be integrated with the big data systems.

The rest of the paper is structured as follows: An overview of Blockchain technologies and Big Data systems are provided in Section 2. The relevant related works are discussed in Section 3. Section 4 discusses the overview of the TOTEM framework, which integrates blockchain and big data systems. The detailed explanation of the customized computational part of the framework is also given in this section. A proof for the concept with the methodology used and implementation details are given in Section 5. Future work and conclusion are given in Section 6 and 7, respectively.

2 Background

TOTEM framework consists of two main components, blockchain and big data systems. Blockchain allows the authorized users to perform there required analyses on the data owner's data, without moving the data across the network. An overview of the technologies used in the framework is given below.

2.1 Blockchain Technology

The blockchain is an open distributed ledger, that efficiently records transactions [2]. It allows peer-to-peer communication between nodes, instead of a central node that controls the entire network. In 2009, Satoshi Nakamoto introduced Bitcoin [3], which is a decentralized and permissionless blockchain. Blockchain records are tamperproof and transactions are transparent to all users in the network. Transactions in blockchain are stored by using cryptographically secure hash functions. Nodes can individually verify these transactions. Blockchain can be mainly classified into two categories, such as private and public blockchains. Public blockchains are permissionless as the name indicates, there are no restrictions in joining the network for the verification process and also the participants are anonymous. Examples of the public or permissionless blockchain are Bitcoin and Ethereum [4]. In the other side private or permissioned blockchain, required permission for joining the network. It is restricted to authorized users within an organization or group of organizations. The verification process is executed by selected nodes in the blockchain consortium. Examples for permissioned or private blockchain are Hyperledger Fabric [5] and Ripple [6]. The validity of a transaction in blockchain can be verified by any node in the network. After verification, these nodes can add this transaction to a block and append to the blockchain. There is a possibility of more than one node can come up with such new blocks. To avoid this, a consensus algorithm is introduced. Consensus algorithm will help nodes to agree on which node will append the block next. Consensus algorithms are mainly proof based or voting based [7].

2.2 Smart Contracts

Smart contract concept was first proposed by Nick Szabo [8]. Blockchain can enable smart contract, which can be any kind of computer program. Blockchain-based smart contract will be executed in a decentralized manner. Transitions in blockchain will occur only if the conditions in smart contract satisfy. In public blockchains anyone can deploy smart contracts, thus it demands high computation [3]. Measures are taken to control this kind of high computational requirements. For instance, 'gas' concept in Ethereum blockchain is to tie up with smart contract execution complexity to financial limitation [4]. It will also prevent additional or unrealistic delays in the network.

2.3 Big Data System

Very large datasets that cannot efficiently interpret, collect, manage and process using traditional mechanism is referred to as big data [9]. Main features of big data are variety volume, velocity and veracity [10] which deals with nature of data, size of generated and stored data, speed of data generated and data quality respectively [11]. The strategy for analysing these large volumes of data or big data can be denoted as big data analysis. Among the frameworks used for big data analytics, Hadoop [12] [13], MangoDB [14], Spark [15] and Strom [16] are quite popular.

2.4 Hadoop

Hadoop is a framework that manages a cluster of computers for distributed processing of large datasets. It is an open-source distributed processing where implementing is based on MapReduce [12] programming model. Hadoop Distributed File system also known as HDFS [13] is a layer in the Hadoop framework.

Hadoop distributed file system maintains a master-slave architecture. Master or NameNode contains file system metadata. The files are divided into fixed-size blocks and stored in the slave or DataNodes. A heartbeat signal is periodically sending to NameNode from DataNodes in order to indicate that the node is still alive and active.

NameNode has the responsibility to map the blocks to the respective DataNode. In the other hand DataNodes are responsible for read-write operation in the file system and also creation, deletion and replication of blocks according to the instructions given by NameNode. The node that keeps checkpoints of the file system metadata on the NameNode is known as the secondary NameNode. MapReduce model has mainly two types of tasks such as a map and reduce functions or tasks.

Input data of map function will further convert into granular structures and the output contains a key/value pair which will be in a tuple format. The output of map function will be taken as the input of following reduce function. Reduce function will combine and groups the tuples according to the unique key value. A file system is required because the input and outputs of these function are stored in files. To manage and track the distributed processing a JobTracker and multiple TaskTracker are there in the framework [17]. JobTracker in the master node will monitor the resource availability, allocates the resources and thus schedule the tasks for the TaskTracker. TaskTracker in the slave node will compute the tasks according to the task scheduled by JobTracker. At a regular interval of time TaskTracker will provide the status information to the JobTracker, which is referred to as the heartbeat signal. If there is no heartbeat from a TaskTracker for a period of time it assumes that the particular TaskTracker goes down and JobTracker will reschedule the failed task to the next available TaskTracker. The entire process will get halt if the JobTracker goes down. Figure 1 below shows the basic master-slave architecture.

3 Related Work

Many proposals and implementations are there related to the integration of blockchain and big data. For reinforcing the security of big data, a blockchain-based access control framework is proposed in [18]. Transparency, fine-granularity and pseudonymity are some of the objectives achieved by this framework. It also mentioned about the additional critical issues observed when adopting blockchain tech-

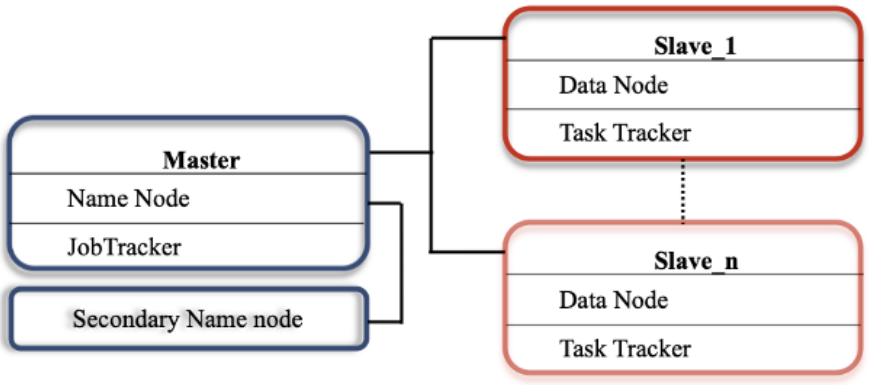


Figure 1: Master slave architecture

nologies for handling the access controls functionality. Possibilities of blockchain technologies on big data systems are mentioned in [19], which shows how to utilise decentralized management for private data, resolution of digital property, public sector institutions and for IoT communication. In SHDFS architecture [20] the possibilities of using blockchain technologies as NameNode and thus eliminates the requirement of secondary nodes. A guarantee to effectively manage the access control of large dataset and protect data from data breaches by using blockchain-based access control ecosystem is mentioned in [21]. As blockchain ensures data transparency, traceability, secure data sharing and data self-sovereignty for the data owner, it solves issues associated with traditional centralized access control. By adding immutability and decentralization to HBase database through blockchain a scalable blockchain-based big data storage for distributed com putting is introduced in HBasechainDB [22]. Organizations that have Hadoop based business logic can easily this since it inherits the efficient big data processing.

However, TOTEM: Token for controlled computation [1] is a framework which can effectively utilise the advantages of both blockchain and big data systems. “This framework ensures the security and privacy of data to the data owner and provide a controlled computation facility for third-parties to execute their own code on the data.”

4 TOTEM Architecture

TOTEM architecture which is proposed in the TOTEM: Token for controlled computation [1] is shown in Figure 2. The architecture consists of three major layers such as blockchain consortium, storage layer and computational layer. Data consumers and data providers are the two actors connected through the blockchain consortium. According to the GDPR [23], data consumer and data providers are a data processor and data controller respectively.

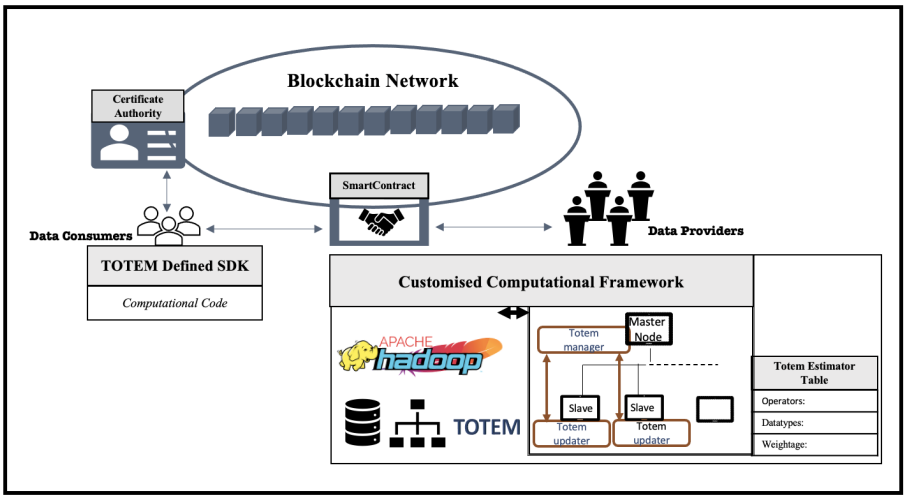


Figure 2: Proposed architecture

The data and the resources for computation are owned by the data providers. Authorized data consumers can view this metadata and given an opportunity to execute their own code on the requested dataset. Responsibilities of data providers are:

(1) Deploy the smart contract which evaluates the code submitted by the data consumer. As blockchain consortium contains multiple data providers, this smart contract should be a collective effort from all the data providers.

(2) Publish the meta-data of the dataset they own, to the blockchain.

(3) Execute the data consumer pre-checked code on the requested dataset and provide the results.

(4) After each operation, check the used totem value for that particular operation by totem manager and updater.

(5) If the totem value is not exhausted before the final execution, provide the final result to the data consumer.

Authorized users submit their computational code using the TOTEM defined SDK and it will be preliminarily monitored by a smart contract which is deployed by the data provider. It also checks for any malicious functions or infinite loops in the consumer code and limits the computational complexity. This monitoring must be a standardized procedure for all the data providers.

In order to prevent malicious functions in the submitted code and limit the computational complexity, the concept TOTEM: Token for controlled computation is introduced. It thereby provides a proper validation for the computational code submitted and prevents any delay on the network. As similar to the gas concept in Ethereum [9], a pre-defined totem value will be assigned for each authorized user. Based on the computational need of the submitted code, the pre-defined totem varies. During the actual execution, depends on the computations of each operation, the totem value reduces. The computation will continue until the final result is produced or when the totem value exhausts. We also introduced a totem manager and updaters, in order to continuously monitor the usage of totem during the entire execution.

The architecture is shown in Figure 2 contains a storage layer and computational layer which is represented by the Hadoop distributed file system, and a MapReduce framework for distributed processing, respectively. Since the actual execution will take place in the distributed processing part, the accurate totem value usage can be calculated from there. MapReduce framework contains the master-slave model where master assigns tasks for slave nodes and slave nodes to execute the task. After the execution of each opcode, the totem used for that particular opcode should be reported to the master node from slave nodes and the master node should have the

functionality to parallelly collect, calculate and inform each slave nodes about the remaining totem value. The computational part mentioned here contains the existing Hadoop MapReduce framework in addition to the totem manager and updater which communicate and exchange the status of the totem value. In this paper, the proof for the newly added components of the computational part of the framework is presented.

The workflow of the TOTEM architecture is shown in the flowchart in Figure 3, provided, the data provider/ data owner is already in the blockchain network and deployed the smart contract and provided the metadata of the dataset they own. The flowchart shows the entire workflow.

4.1 TOTEM Defined SDK

Totem defined SDK will have a set of rules which is designed and defined in order to meet the requirement of the TOTEM architecture. According to the designed rules and format, the user will get guidelines on how to write the code using the SDK. The SDK will validate the user code. In addition to that, it also has the responsibility to provide a rough estimate of the totem required for that particular user code. If the user has enough totem, it will pass the user code for real execution in the customized computational framework. While passing the user code, the SDK will convert these opcodes into an array of operations. This array will be stored and used when the actual execution of the user code occurs in order to calculate the totem used.

4.2 Customized Computational Framework

When an authorized user requests and submit the computational code, it will go through the aforementioned preliminary check. If all the requirements are satisfied and the user have enough totem value, the code will be further executed on the data providers' computational framework. Hence one of the responsibilities of the data provider is to provide a platform for computation. Big data computational platforms such as Hadoop, Spark and Strom can be used for the computational part. But in order to manage the controlled usage

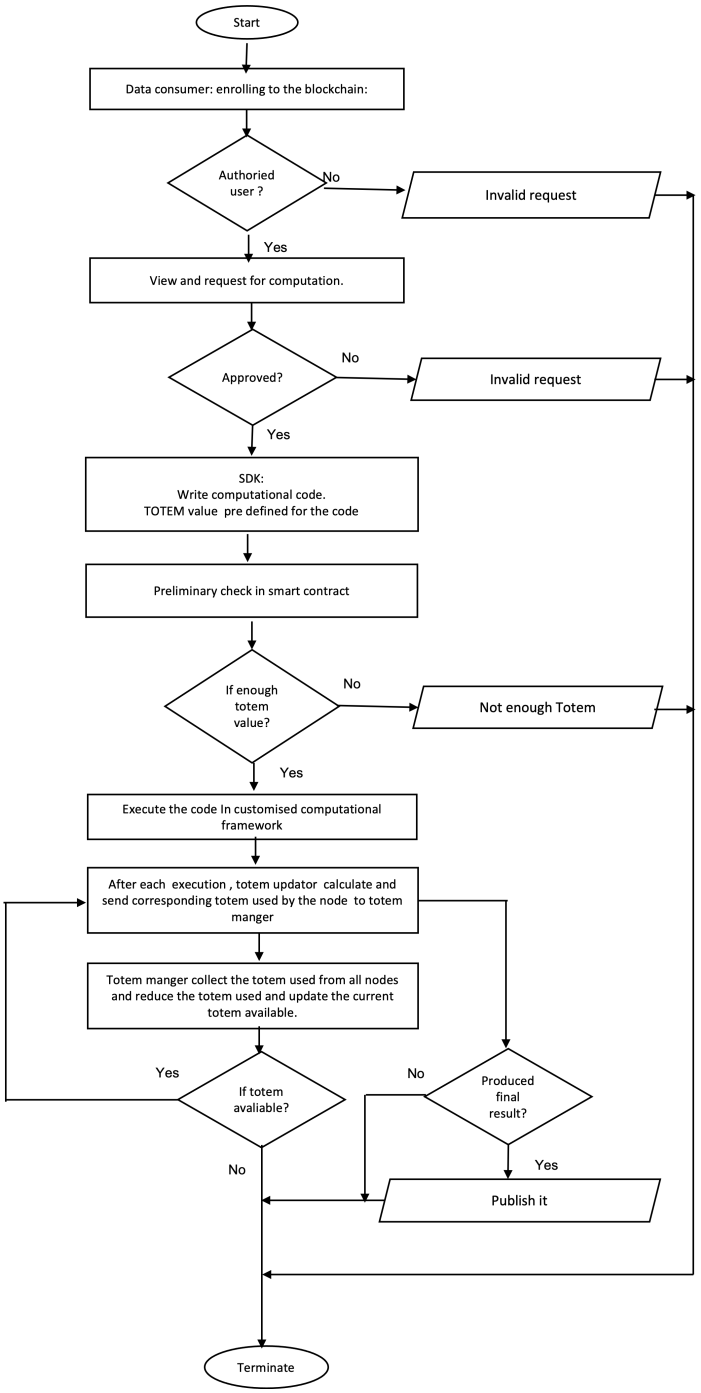


Figure 3: Workflow of TOTEM architecture

of computational resources, we need an additional component called totem manager/updater. According to the TOTEM architecture, the existing big data computational framework with an additional totem manager/updater is called Customized Computational Framework (CCF). Hadoop MapReduce framework is used in this present work to demonstrate the CCF part. The Hadoop framework consists of master-slave architecture. Master node divides the task and gives it to the slave nodes, where slave nodes will execute the tasks and return the results. Here the actual execution happens in the slave nodes/DataNodes. For every single operation executing in each node, the totem value corresponding to that operation should reduce from the total totem value. This totem value corresponding to each operation should be reported by the totem updaters in each slave nodes to the totem manager in the master node. Each time after receiving totem value, totem manager will reduce that amount from the available totem and check if the totem value is greater than'0'. If totem value is still available, it will reply the balance amount to each totem updaters otherwise, it will immediately stop the entire execution.

5 Proof of concept for the Customized Computational Framework

The novel part of the newly proposed framework is the totem manager and updater, which exactly control the computation. Hence it is relevant to show the practical side of this concept. A detailed explanation of the methodology and environment set up is discussed in this section.

5.1 Methodology

In the framework, the totem manager and updaters need to communicate after the execution of each opcode. Since the computational framework already has the Hadoop setup, the nodes are connected and can communicate between the master/NameNode and slave/-DataNodes. However, the totem required to execute each opcode

varies. It depends on the data type of the operands and the operation performed. The table which contains the details regarding the totem required to execute each operation and datatypes is known as the totem estimator table. This estimator table is maintained by the data providers. i.e., the data owners are responsible to fix the totem required for each set of operations. It should be standardized.

A predefined totem is provided to each data consumer. After each set of operations, the totem manager needs to provide the current status of the totem value. This value will be checked and saved in the blockchain as a transaction. If the current value is greater than zero, then the next set of operations will be carried out. Since parallel computation is taking place in big data systems, from each node we need to collect the information about the totem consumed for each operation. In order to do this, totem updaters are introduced in each and every DataNode. After each operation, depends on the opcode, corresponding totem value will be calculated with the help of totem estimator table and the results are updated to totem manager. This will be the main responsibility of the totem updaters.

The methodology used in the CCF for controlled computation is shown in Figure 4. The user code submitted by the data user through TOTEM defined SDK will be given to the NameNode. An array of opcodes corresponding to the user code and the available totem assigned for that particular data user will be provided to the totem manager. The DataNodes have access to the totem estimator tables. Once the request for the computation is accepted, totem updater from the DataNodes confirms the connection to totem manager in NameNode. Then the array of opcodes corresponding to the user code which is currently going to be executed in the DataNode will be passed to the totem updater. For each element/opcode in the array, totem updater will calculate the totem used while executing that opcode. This calculated totem value for that opcode will be then passed to the totem manager. The totem manager then reduce that value from the available totem value and if the totem is not exhausted it will pass a message to continue execution.

If the totem value is exhausted, it will inform the master node and immediately halt all executions in the slave nodes.

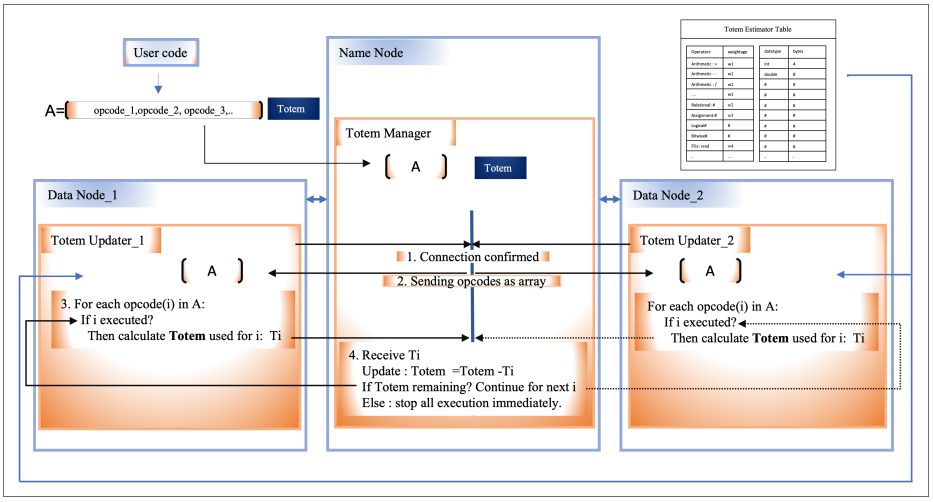


Figure 4: CCF Methodology

5.2 Experimental setup and implementation

In the present paper, the Hadoop environment is used as the computational framework. Three nodes are used, one as a master node and two slave nodes. In both the slave nodes, totem updaters and in the master node totem manager is implemented. As a proof for the TOTEM customized computational framework concept, the focus is on the functioning of the totem manager and updaters. The actual execution of the user code is not considered in this paper. Experimental set up as follows:

For the implementation we used three nodes with the following configuration:

- One node with Intel(R) Xeon(R) CPU E5-2640 v4 CPU 2.40 GHz*4 processor and 16 GB of memory, with Ubuntu 16.04.6 OS.
- Two nodes with Intel(R) Xeon(R) CPU E5-2640 v4 CPU 2.40 GHz*2 processor and 8 GB of memory, with Ubuntu 16.04.6 OS.
- In each of these nodes Hadoop version: hadoop-2.7.3 runs where one as master and 2 as slave nodes

- In three nodes Python 2.7.12 is installed.

In the present setup, it is assumed that the execution of the code will be done in each slave nodes and after each opcode execution totem updater will be notified. The following steps are implemented. The totem manager in the master node will read the file that contains the array of opcodes converted from the user code and send to both totem updaters in the slave nodes of the Hadoop environment. The totem updaters already have access to the estimator table, the file which contains the unit of totem required per datatypes. The totem updaters after receiving the array of opcodes, each element in the array will be passed through a function to calculate the required totem for that particular opcode. This will be then sent to the manager to update the current totem value and if it is still available, a continue signal is sent back. This process will continue until totem value is exhausted or the execution stops immediately or the entire execution finishes without exceeding totem value. This part is successfully implemented in order to show the viability of the concept.

A sample test done with a predefined array of opcodes is shown below in Figure 5. It shows the connection start point of all three nodes and the progress. Eventually, the final part after all the execution is done, it will display the available totem balance.

6 Future Work

In the present paper, it is demonstrated how to deal with the usage of totem entity, which depends on the computational complexity of the user code. The user code will analyse the data from a single data source. However, it is not shown how to utilise data from multiple data source and deal with the usage of totem entity in the TOTEM architecture. Hence the architecture needs to be further extended to handle or utilise multiple datasets from multiple organization without transferring the data across the network. Also, the paper does not discuss how the GDPR is addressed in the existing TOTEM architecture and its impact on the architecture.

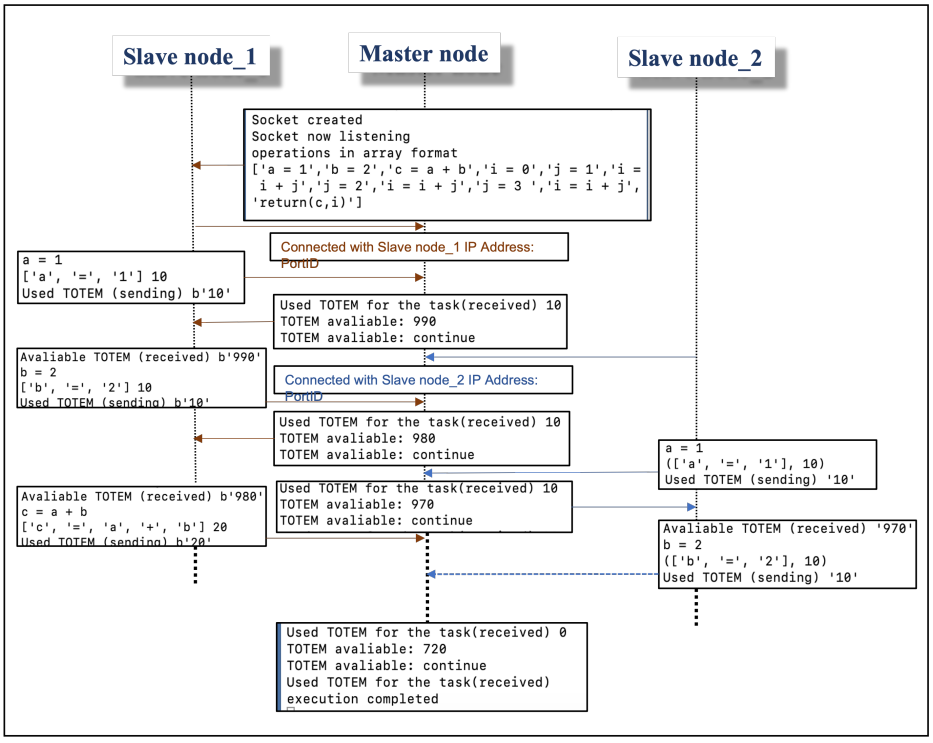


Figure 5: Execution flow

7 Conclusion

In this paper it is discussed about an architecture known as TOTEM: Token for controlled computation, which is a solution for secured and privacy driven big data analytics through blockchain technologies. An entity called totem is considered as the unit of computation, is mentioned in the architecture. Instead of moving the data across the network for analysis, the user code for analysing the owner´s data will be given to the owner's environment through a TOTEM defined SDK. According to the computational complexity of the user code, the predefined totem value gets consumed during the real execution. This part is designed in the customized computational part of the architecture. The methodology or approach for this totem management is explained in the paper and implemented the core parts, totem manager and updater. The implementation was a proof

for the concept to demonstrate how the customized computational framework deal with the usage of totem value during the computation.

References

[1] Dhanya Therese Jose, Antorweep Chakravorty, and Chunming Rong. "TOTEM : Token for controlled computation: Integrating Blockchain with Big Data." In: (2019), pp. 1–7. DOI: 10.1109/ICCCNT45670.2019.8944855.

[2] Karim R Lakhani and M Iansiti. "The truth about blockchain." In: *Harvard Business Review* 95.1 (2017), pp. 119–127.

[3] Satoshi Nakamoto. "Bitcoin: A peer-to-peer electronic cash system Bitcoin: A Peer-to-Peer Electronic Cash System." In: *Bitcoin. org. Disponible en https://bitcoin. org/en/bitcoin-paper* (2009).

[4] Vitalik Buterin et al. "A next-generation smart contract and decentralized application platform." In: *white paper* 3.37 (2014), pp. 2–1.

[5] "Hyperledger Fabric Documentation Release 1.4." In: *Hyperledger* (2019).

[6] "Ripplee https://ripple.com/." In: (2020).

[7] Giang-Truong Nguyen and Kyungbaek Kim. "A survey about consensus algorithms used in blockchain." In: *Journal of Information processing systems* 14.1 (2018), pp. 101–128.

[8] Nick Szabo et al. "Smart contracts." In: (1994).

[9] Min Chen, Shiwen Mao, and Yunhao Liu. "Big data: A survey." In: *Mobile networks and applications* 19.2 (2014), pp. 171–209.

[10] Judith Hurwitz, Alan Nugent, Fern Halper, Marcia Kaufman, et al. "Big data for dummies." In: 336 (2013).

[11] Parth Chandarana and M Vijayalakshmi. "Big data analytics frameworks." In: (2014), pp. 430–434.

[12] Jeffrey Dean and Sanjay Ghemawat. “MapReduce: a flexible data processing tool.” In: *Communications of the ACM* 53.1 (2010), pp. 72–77.

[13] Dhruba Borthakur. “The hadoop distributed file system: Architecture and design.” In: *Hadoop Project Website* 11.2007 (2007), p. 21.

[14] “Mongodb https://www.mongodb.com/.” In: (2020).

[15] “Spark https://spark.apache.org/.” In: (2020).

[16] “Strom http://storm.apache.org/.” In: (2020).

[17] Vinod Kumar Vavilapalli, Arun C Murthy, Chris Douglas, Sharad Agarwal, Mahadev Konar, Robert Evans, Thomas Graves, Jason Lowe, Hitesh Shah, Siddharth Seth, et al. “Apache hadoop yarn: Yet another resource negotiator.” In: (2013), pp. 1–16.

[18] A Outchakoucht Jp Leroy H Es-Samaali, Nn Van, and R Nakagawa T Tanouchi S Kodama. “A Blockchain-based Access Control for Big Data.” In: *Journal of Computer Networks and Communications* 5 (2017), pp. 137–147.

[19] Elena Karafiloski and Anastas Mishev. “Blockchain solutions for big data challenges: A literature review.” In: (2017), pp. 763–768. DOI: 10.1109/EUROCON.2017.8011213.

[20] Deepa S Kumar and M Abdul Rahman. “Simplified HDFS architecture with blockchain distribution of metadata.” In: *International Journal of Applied Engineering Research* 12.21 (2017), pp. 11374–11382.

[21] Uchi Ugobame Uchibeke, Kevin A Schneider, Sara Hosseinzadeh Kassani, and Ralph Deters. “Blockchain access control ecosystem for big data security.” In: (2018), pp. 1373–1378.

[22] Manuj Subhankar Sahoo and Pallav Kumar Baruah. “HBasechain DB-a scalable blockchain framework on hadoop ecosystem.” In: (2018), pp. 18–29.

[23] “GDPR https://gdpr-info.eu/.” In: (2020).

Paper III:
TOTEM SDK: an open toolset for token controlled computation managed by blockchain

TOTEM SDK: an open toolset for token controlled computation managed by blockchain

Behfar Behzad[1], **Dhanya Therese Jose** [1], **Antorweep Chakravorty**[1], **Chunming Rong**[1]

[1] Department of Electrical Engineering and Computer Science, University of Stavanger

Abstract:
As a part of using the Internet, surging over the past decade, a wide range of data has been generated at a breakneck pace from various sources such as social media, banking sectors and governments. Many organizations have changed their work culture and adopted big data analytic to gain various benefits from the data being produced. Nevertheless, sharing this big data was always a tremendous challenge for scientists and engineers as it involves a large volume and sensitive data, which cannot be handled by the conventional way of data analysis. TOTEM: Token for controlled computation, accounts for an innovative concept that integrates both blockchain technologies and big data systems and uses their advantages to present a better, secure and more cost-effective solution for both data owners and data consumers. The TOTEM architecture (US Patent No.: US11,121,874 B2) aims to overcome security and privacy breaches and prevent moving large datasets across the network for analysis. Totem is an entity used in TOTEM architecture for putting constraints on computational operations. Authorised users in the network are allowed to write their own code in a specific format through a TOTEM defined SDK for analysing the data provided by the data owner. The SDK, along with the deployed smart contracts in the network, form a pre monitoring system that keeps track of totem entity value associated with each users' submitted codes using an estimator table. In this article, we will focus on the layers of this TOTEM defined SDK and further explain how the SDK interacts with the code, analyze it within the layers and finally how it responds to it.

1 Introduction

In this modern era the role of technology becoming multi-fold in all sectors, it demands to deal with large and diverse data in various industries that keeps increasing exponentially. Nevertheless, conventional methods for data analysis are not privacy preserved and secure enough as they demand moving data across the network from the data owners environment for the purpose of data analysis. Here we can see the importance of a framework for secure and privacy preserved data analysis known as TOTEM: Token for controlled computation [1] [2]. This patented framework is capable of configuring and handling the storage and computational mechanism while ensuring the security and privacy of the data. The architecture of TOTEM allows the computational code for analysing the dataset to move towards the data owner´s environment instead of the conventional method of data analysis. TOTEM is an approach that allows moving the computational code to the data without any concern regarding the data disclosure to external networks. TOTEM employs the properties of big data analytic tools such as the Hadoop framework, for its parallel computation on large datasets, and Blockchain technology for ensuring each transaction secured and tamper-proof, thus it enables a new way of computation upon large datasets securely and effectively. In order to submit the code for computation in a specific format and implement a mechanism for monitoring the malicious functions which may occur in the submitted code the framework introduced TOTEM defined SDK. In the present work, we focused on the TOTEM defined SDK architecture layers and basic functionalities involves in it. The main objective of this paper is an implementation of a Software Development Kit (SDK) for the TOTEM architecture integrating the Hyperledger Fabric blockchain and the big data framework, its architectural design process and the methodologies employed to develop the SDK.

The rest of the paper is arranged as follows. The background of the work and prior works are there in Section 2 and Section 3 correspondingly. In Section 4 we have given a detailed description of the TOTEM defined SDK architecture. Its shows the three layers in the architecture and explained the functionalities of each layer.

Implementation and results obtained is shown in Section 5. Finally in Section 6 we concluded and suggested future works and improvements required in this current work.

2 Background

Since we are discussing about an architecture that integrates blockchain technologies and big data frameworks, it is relevant to go through the basic concept of both the technologies. Blockchain and big data are two the leading technologies that have grabbed a great deal of attention. Both are expected to reshape the way businesses are done across all kinds of industries in the years ahead. Further in this section an overview of these technologies used in TOTEM architecture as well as the necessary tools used to implement the TOTEM defined SDK is given.

2.1 Blockchain Technology

Blockchain is an open distributed ledger that allows multiple parties to add and store transactions efficiently and permanently into it. In 1991, S. Haber and W. Stornetta described cryptographically secured chains of blocks to make the document timestamps tamper-proof [3], and used the Merkle tree to allow timestamping of several documents into one block. In Blockchain, data can be embedded in digital code and be recorded in a transparent manner. This method makes Blockchain protected against revision, tampering, and deletion. It is, in fact, a Point to Point (P2P) communication among all nodes in the chain; therefore, no central node can monitor or control the whole network. Blockchain can be mainly classified into two types such as public and private blockchains [4]. The public or permissionless blockchain, signifying that the participants are anonymous and no restriction for joining the network [5] whereas in private or permissioned blockchain, prior approval is necessary and participants required permission to join the network. Hyperledger Fabric [6] belongs to private blockchains.

Smart contracts account for one of the promising uses of Blockchain

that has exploited this technology's potential even beyond cryptocurrencies. In [7], Nick Szabo, cryptographer and lawyer, was the first person who coined the term in the 1990s. He suggested translating the contract terms into computer codes and added them into the software or hardware. This suggestion not only makes the contract automated and self-executed, but it also minimizes the possibility of accidental exceptions and fraudulent transactions between parties [8]. Without third party involvement, this low-level computer code deployed on the network monitors all transactions so that each transaction in the blockchain is executed only if they meet all the terms and requirements in the smart contracts.

2.2 Big Data System

Big data refers to large, unstructured, and heterogeneous datasets which are beyond the capabilities of standard analytical methods in data management in an acceptable amount of time [9]. Data management here means that it starts from collecting, storing, processing and finally visualizing the data. Various methods, tools or strategies and frameworks are developed to analyze significant volumes of data. Among those frameworks such as Hadoop [10], Spark and Storm are more popular. Hadoop is an Apache project that began in 2008 and accounts for one of the leading frameworks in distributed processing of big datasets with clusters of computers. Hadoop is an open-source framework that works on a programming model approach, namely MapReduce to process and generate large datasets [8]. Hadoop principally consists of a two-layer structure, designed to improve the performance of handling I/O requests [11]. These layers are called the Hadoop Distributed File System (HDFS) and MapReduce (distributed processing).

HDFS is an implementation that is similar to Google File System (GFS) and provides a scalable Distributed File System (DFS) to record big files on distributed machines reliably and effectively. HDFS has a master/slave architecture where the NameNode is the master with several DataNodes being slaves [10]. The NameNode or the master is mainly responsible for providing physical space to record massive data files sent by the HDFS client. DataNodes or slaves, on the other

hand, are used to store HDFS client data files after files being split into fixed-sized blocks. MapReduce is a programming model for processing and producing large datasets. MapReduce is a highly efficient framework for large scale data analysis [8]. MapReduce distributed processing framework consists of two principle functions or tasks that are executed: map and reduce. Users specify a map function that takes input and generates a set of intermediate key-value pairs as output [12]. In the reduce function, the output from the map function sorted in the file system which is the intermediate key-value pairs are merged and grouped by key. MapReduce framework is based on two components such as JobTrackers and TaskTrackers. JobTrackers manage resources and schedules task for the TaskTrackers.

2.3 JavaScript

JavaScript (JS) is an imperative, object-oriented language first announced in 1995 by Netscape as an "easy-to-use object scripting language designed for creating live online applications that link together objects and resources on both clients and servers" [13]. Since then, JavaScript(JS) has become the standard for front-end scripting and even one of the most dominant languages in the software industry. Unlike other popular traditional languages such as Java and C, JavaScript does not allow encapsulation by using classes or structured programming to maximize flexibility. JavaScript prosperity is undeniable to the extend that, based on Google's report as a data point, it has been used in 97 out of 100 most popular websites and web applications [14]. In JS, objects can be sent over the web as raw strings that can be dynamically parsed and easily executed by the receivers, and its APIs support by all modern browsers [15]. Less load on the server and speed are two of the distinct advantages of using JS in software or application development over the other programming languages.

2.4 Node.js

Node.js is an open-source, server-side JavaScript runtime environment that runs JS scripts outside of the web browser. Node simplifies

the creation of multi-functional web servers as well as networking tools by using JavaScript. It provides a collection of modules such as file system I/O, data streams, networking protocols, including HyperText Transfer Protocol (HTTP), Domain Name System (DNS), and Transmission Control Protocol (TCP)/ User Datagram Protocol (UDP), etc.[16]. Node.js is primarily used to bring event-driven programming to web servers, owing to make the server development fast in JS. It allows developers to build scalable servers without the use of threading as it used to be traditionally, but by using an event-driven programming model that uses call-back functions to indicate that a task is completed [16].

3 Prior Work : TOTEM framework

In TOTEM architecture [1], the blockchain technologies and the big data framework play a complementary role to solve many challenges accompanied by traditional methods such as inaccessible data due to the data owner reluctant to reveal their data, security breaches and transferring of large datasets over the network. TOTEM aimed at bringing the computational code to data to avoid the drawbacks associated with the traditional methods for data analysis, in particular with higher bandwidth demands and security breaches. As discussed in the background, the Hadoop framework allows parallel computation on large datasets through its Map Reduce programming model. The parallel execution is performed through a parallel map over input data in the first place. Then, the intermediate data are grouped in a key-value pair, as needed for the reduce phase. Reduce function is carried out ultimately to produce the output data [12]. On the other hand, Blockchain is restricted by block size and creation frequency; consequently, it is incapable of handling large datasets or parallel processing, but it ensures that each transaction through the blockchain network is secure and tamper-proof. TOTEM combines the strength of these two technologies, which can complement each other, to develop a new approach. In this section, we will go through a brief description of TOTEM architecture layers and two major actors in the network.

TOTEM architecture has three fundamental layers such as blockchain consortium, storage layer, and computational layer. For users according to their role in the blockchain network they can submit code for computation, obtain results, deploy smart contract, publish the metadata of their own available dataset, etc. All the transactions or actions through the blockchain network is recorded as transactions that can be verified later. Totem entity act as a token in the system, for the controlled computation or to put constraints for the computational complexity of the submitted code by the user through TOTEM defined SDK. Smart contact deployed in the network will do the preliminary check on the validity of the submitted code. The storage layer is where the actual dataset is stored. It will be in the data owners environment itself. The computational layer will handle the actual execution of the submitted code by the user on the data owners dataset. The computational layer contains resources for the parallel computation that it also resides in the data owner environment. The two actors in the network are data provider and data consumer. Data provider is the one who owns and providers facility to analyse their dataset. Data consumer is the one who requests and writes computational code for the analyses of the available dataset. The actions or major responsibility of two actors in the network is explained in the coming subsections.

3.1 Data Consumer

Data consumer will join the network for analysing particular dataset if it is available. This process includes actions such as writing the code in a specific format and submitting it through a TOTEM defined SDK. A detailed description of SDK architecture and its workflow is given in Section 4. There are certain formalities to go through for reaching this opportunity. In this subsection, we will go through the various steps needed to pass for reaching this goal. We need to assume that the smart contract is deployed in the network as a collective effort from all the data providers joined in the network and also the metadata about the datasets from all the data providers is available and accessible in the blockchain network.

The workflow begins with the process of enrollment and gets autho-

rized in the blockchain network as in Figure 1. The data Consumers can gain access to the datasets, provided under the blockchain consortium once they are authorized users in the network.

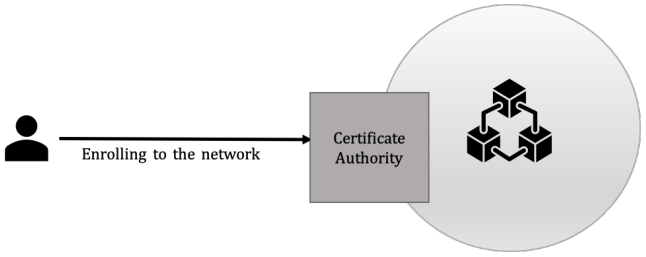


Figure 1: Enrollment in Blockchain consortium

An authorised data consumer will be allowed to view the metadata and request for the computing facility provided by the data providers for analysing the selected dataset. The request will be processed and if they are authorised to use the facilities from that requested data provider then the request will be accepted. This process will also check for the totem value balance of the data consumer as well. Each data consumer will have a totem account balance and this totem value is essential for further computation. However, acquiring or buying totem from the blockchain consortium is not discussed in [1] since it is highly dependent on the underlying network architecture. In Figure 2, proceedings of data consumer including view/request and the response for the request is shown. Once the request from the data consumer is accepted the computational code will be submitted for execution on the requested data. The computational code must correspond to the MapReduce pattern of the SDK defined for the TOTEM. Further in this, the smart contract which was deployed by the data providers will perform a preliminary evaluation on the consumer code. It processes the submitted code and determines whether there are sufficient totems provided by the data consumer regarding the computational needs of the submitted code.

Totem Estimator Table (TET) accounts for a table that describes standard totem values to perform each of the defined opcodes [1]. As mentioned earlier, the smart contract is executed before the computation of the consumer code to estimate the required totem

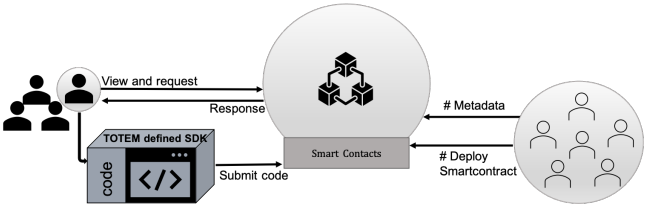


Figure 2: Submitting Computational code

and then check the data consumer's balance. This estimation of the submitted code is carried out by using the information within the TET.

3.2 Data Provider

Further steps of the TOTEM architecture workflow proceed in the data providers' side as shown in Figure 3. When the consumer code is allowed for actual computation after preliminary check, this computation will be carried out in a Customized Computational Framework (CCF), which resembles the architecture of Hadoop with some modifications. The CCF contains two components such as totem manager in the master node and also totem updaters in each slave nodes for the continuous monitoring of totem current value. Following each execution of the given opcodes, totem updaters will calculate the totem value for that particular opcode performed in the corresponding slave node and report to the totem manager resides in the master node, which is responsible for the calculation of available totem balance. All the totem updaters will update the used totem value like this and the totem manager will calculate the balance totem value accordingly. After each set of opcodes performed in the nodes, the master node will update the available totem value to the network as a transaction.

This cycle will repeat either until the completion of executing a chain of map/reduce function or the available totem exhausts and the system reaches the "Out of totem" status. In case the prior happens, updaters send the final report to the totem manager. However, if the latter takes place, the totem manager will immediately

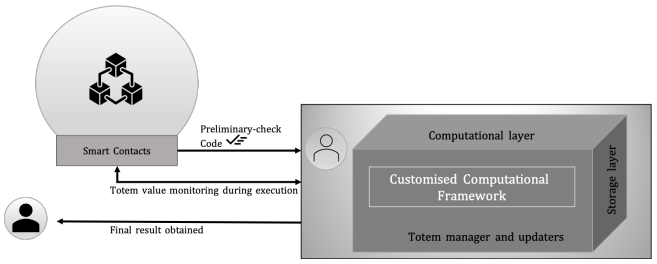


Figure 3: Data Provider Environment

signal the master node to halt the execution. When the execution of the map/reduce the function has been completed, the manager sends the final amount of totem consumed during the execution of the blockchain network. For this purpose, the manager creates a transaction, which invokes the smart contract that has already been deployed to the blockchain consortium. Finally, when the computation on the specified dataset completes or there is not enough totem for the rest of the execution, the currently obtained result will be published to the data consumer.

4 TOTEM defined SDK

As discussed in previous sections, the SDK plays a vital role in the architecture of TOTEM. It is aimed at enabling the data consumers to submit their computational codes in the data providers' side in a specific format. Moreover, the SDK produces an estimated value of required totem that consumer's code might cost. Ultimately, the SDK outputs a formatted code needed for the execution section. This shows the relevance of having an extensible, well-structured, and easy-to-use SDK. The TOTEM defined SDK presented here, however, is not the final version but we aimed to show the potential functionalities regarding some predefined rules. This section introduces the highlights and describes the different layers within the architecture of the defined SDK to give a better understanding of the implementation in Section 5.

4.1 Specifications of TOTEM defined SDK

TOTEM defined SDK provides a flexible software development kit for coding, testing and submitting computational codes of data consumers over the network, within the frame work of TOTEM architecture. Some of the highlights about TOTEM defined SDK is as below:

• Modular Design Pattern – TOTEM SDK is designed from the beginning to confirm Modular Design Pattern (MDP), allowing for wrapping a set of variables and procedures together in a single scope. The module interface is, in fact, a piece of program that specifies which other pieces it relies on and which functionality it provides for other modules to use. By limiting the way that modules interact with each other, the whole system can be seen as a LEGO, where pieces interact through well-defined connectors called dependencies [17]. From all the benefits that MDP brings in favor of interdependent codebase, the following three are more important in the TOTEM SDK: Maintainability, Namespacing and Reusability. The modular pattern makes it simple to improve and update single modules and reuse them in new required places in the project. It also helps to avoid "namespacing pollution" where completely unrelated code shares global variables by providing private space for variables [18].

• Maintainability and extensibility – The core architecture of the presented SDK has been made to be maintainable and extensible as there are still opportunities for further enhancements. This architecture was done through the modular pattern. A well-designed module strives to reduce the dependencies on different parts of the codebase as much as possible so that it can grow and improve independently. Therefore, either adding a new module to the architecture or updating a single module would be much easier while the module is disassociated from the other pieces of code [18].

• The TOTEM SDK presented in this paper is implemented in fully vanilla JavaScript to simplify Document Object Model (DOM). Using JS makes TOTEM SDK very efficient to deploy in web browsers regardless of operating systems and multiple hardware platforms [19]. The source code is put on Github publicly and leverage the whole community and interested developers to improve the SDK for better performance and flexibility.

4.2 Architecture of TOTEM defined SDK

As stated earlier, the TOTEM defined SDK is primarily used for creating a formatted code from the user's submitted code as well as estimating its computational cost, all according to the defined rules. This analysis of the user's code and the estimation of the required totem are taken place within a three-layered architecture, consists of SDK layer, Controllers layer, and Handlers layer. Figure 4 depicts these three layers and shows the workflow, describing how its comprising components are linked to each other in the architecture.

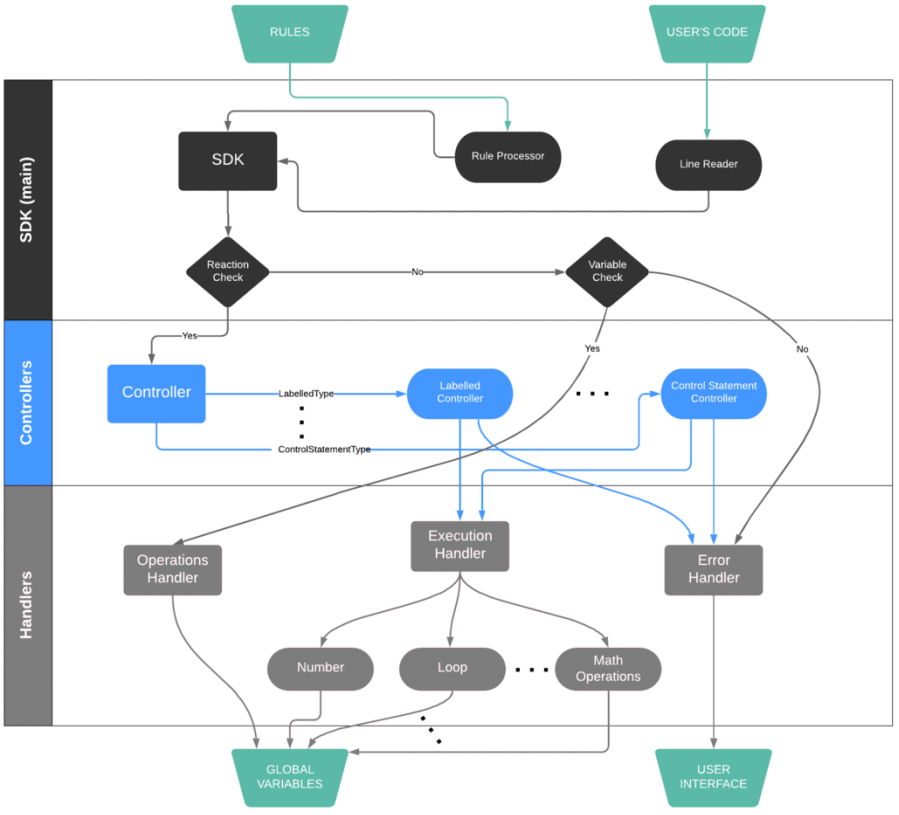


Figure 4: Workflow of TOTEM defined SDK

4.2.1 SDK layer

The SDK layer accounts for the primary layer of the whole design, where all the rules and the user inputs are taken. Each line of the user input goes to the next layer if it complies with the rules. As illustrated in Figure 4, Line Reader Component (LRC) is one of the main components in the current layer in which the process of reading user input lines is carried out. Another critical component in the SDK layer is the Rule Processor Component (RPC). RPC is the only component within the whole architecture which reads, processes and extracts the proper way of handling various commands based on the defined rules, prior to any other action. Consequently, Rules is the first block of scripts that the SDK takes. This action is necessary as all the further actions occurring in the SDK layer are determined based on these rules. Each line of code goes through LRC and will get splitted into two parts, called command part and assignment part. For the command part, if the command exists in the Rules set, the LRC will read the proper reaction to that specific command and rest will be done in the Controller layer, after conducting a check through the Reaction Checker (RC).

On the other hand, if it is not in the rule set, but just an other operation or mathematical equation on an already defined variable, it will be identified in RC and it will be given to Variable Checker (VC). The VC can ignore the Controller Layer and directs both the command and the assignment parts directly to the Operation Handler Component (OHC) in the Handler layer. This happens if the variable checker condition is satisfied. Otherwise, Error Handler Component (EHC) will take control and throws the related exception.

4.2.2 Controllers layer

The next layer in the TOTEM SDK architecture is the Controllers layer. This intermediate layer between the SDK layer and the Handler layer plays a crucial role in enabling the user's submitted code to the following global variables stored after the evaluation in the final step. On a broader level, the Controller wraps all the components, each of which handles the control of a specific command type defined in the rules block. Table 1 indicates an example of rules defined in the

rules block.

Table 1: An example of defined rules in the TOTEM SDK

CMD	**Reaction**	**CMD type**	**Domain**
Int	HandleNumber()	Labelled	Set of Z
Bool	HandleBoolean()	Labelled	Set of $\beta = \{0, 1\}$
Float	HandleNumber()	Labelled	Set of Q
For	HandleForLoop()	Control Statement	Labelled Type Vars
If	HandleIf()	Conditional Statement	Labelled Type Vars

As it is indicated in Table 1, multiple command types might lay in the rules file. Nevertheless, for simplicity and to follow the modular pattern of the design, the Controller redirects the taken commands to their respective components. All further operations on each command type will take place in these components. Labelled Type Controller (LTC) and Control Statement Controller (CSC) are two examples of such controlling components, shown in Figure 4. Following some necessary operations on the Controller inputs, such as naming convention check, splitting the assignment chunk for more evaluation, error check, etc., type controlling components consider where exactly to proceed in the subsequent layer, which is Handlers layer. If the input is error free, then the respective reaction function determined in the "Reaction" column of Table 1 will be invoked. By contradiction, if errors detected within the type controlling component, it invokes a related error in the Error Handler Component (EHC) to show on the User Interface (UI).

4.2.3 Handlers layer

The final layer in the architecture is Handlers layer, which receives requests to handle either execution, throw exceptions, or mathemati-

cal computations. The handlers layer is predominantly responsible for providing functions to the requesting components within the Controllers layer. However, in a situation that a new line of command is detected as an operation on an already defined variable, that line of command within the SDK layer will be sent directly to the Operations Handler Component (OHC) to perform the operations in the Handlers layer. The handlers layer consists of three major components in which several functions are defined. In the operations handler, all allowed operations on existing variables are managed. OHC handles logical operations, relational assignments, and mathematical operations such as addition, subtraction, multiplication and power. The error functions, on the other hand are wrapped up in the EHC. Such a component that handles all the functions, not only makes it more feasible to add, change or remove the functions in the future versions, but it also helps to avoid redundancies by calling interchangeable functions in different places of the SDK. Table 2 reveals various error functions defined so far in the existing version of the SDK.

Table 2: Error functions defined in the EHC

Function	**Call Site**	**Description**
CommandError()	SDK	Command is not defined
HandleError()	LTC	No defined handler to compile the command
NamingError()	LTC	Name is against the naming convention
CTRLStatementError()	CSC	An error occurred during the control statement execution
OperationError()	Execution Handler	An error occurred during the operation functions

Execution handler accounts for another handler components with several functionalities, each of which serves to execute the output of controller components. After Controllers layer, the input line of codes prepared for the execution stage. Depending on the call site of these

functions, different functions could be called. Each of these functions handles at least one type of command defined in the rules file. At the end, the outcomes from the execution will be stored in several global variables for the use of smart contracts in the blockchain consortium. Table 3 shows three global variables as well as their descriptions defined in the existing version of TOTEM defined SDK.

Table 3: Global variables of the defined SDK

Global Variable	**Description**
ruleLines	A global array of map values, each representing one total row of the rules defined in rules.csv
userDefined_vars	A global object to keep track of user-defined variables and their updated values
TOTEM_operators	A global array to keep all operators and assignments within the data consumer's code – used to estimate the required TOTEM
mapped_executions	A global array to store the resulted mapped executions format of the data consumer's code - used as an input in CCF

As discussed earlier, the objective of our TOTEM defined SDK is not to output the final results of the submitted code as it will be done in the Customised Computational Framework (CCF) of TOTEM architecture [20]. Nevertheless, having the updated values of variables is necessary before reading the next line. The reason is that the next lines of the user's submitted code might be dependant on those variables defined earlier. As a result, considering such a global object of most updated variables and their values, which is accessible through the modular pattern in other layers, particularly in the SDK layer, is crucial.

5 Implementation

The SDK is implemented in a way that data consumers are allowed to submit their computational codes and receive reactions. It is also assumed that users have already gone through the authorization process which is outside the scope of this paper. For this purpose, a simple Nodejs [21] application designed to display a realistic interaction among hypothetical users and the SDK in modern browsers. This includes both receiving the consumers' computational codes and showing feedback to them, e.g. exceptions and results. Within the application, SDK is bundled as a JavaScript file using a package called Browserify [22]. Browserify is a module loader library, keeping all modules of the TOTEM defined SDK architecture into a single neat JavaScript file; since browsers still do not support that.

5.1 Main Functionalities

All command rules and corresponding reactions are defined in a separate file within the data provider's site referred to as rules.csv, which should be fetched in SDK before taking the submitted code and estimating the number of required totem for its execution. This is done through an asynchronous request initiated by the browser called AJAX (Asynchronous JavaScript and XML). Processing rules starts when the AJAX callback has been successfully done. It reads each rule and stores all as an array of map values into a global variable known as ruleLine for use in further computations.

Script 1 describes rule processing procedure statements in pseudo-code. The reason for representing each rule as a map value is its constant time algorithm *O(1)* to extract values using their paired keys, rather than tracking single values over an array of commands (*O(n)*). Furthermore, it is ordered, iterable and accepts any type of data as key.

Line reading is the consequent fundamental procedure implemented in the TOTEM defined SDK. It enables the user's submitted lines of code separately and in succession to the lower layers of the defined SDK (Controllers and Handlers layers), if a reaction is set out to each. Each line of code is broken down into two parts; command

Algorithm 1 Rule processing

```
1: procedure ProcessRule (rules)
2:     allTextLines ← an array of splitted rules
3:     headers ← an array of splitted allTextLines[0]
4:     for i ∈ 0, . . . , length(allTextLines) do
5:         data ← allTextLines(i) splitted by comma
6:         if length (data) = length (headers) then
7:             ruleMap ← an empty map.
8:             for j ∈ 0, . . . , length(headers) do
9:                 ruleMap ← [headers(j), data(i)]
10:            end for
11:            Global variable ruleLines ← ruleMap
12:        end if
13:    end for
14: end procedure
```

and assignment. Then, based on the correlative reaction to each command, the procedure routes each line to either the controller or handler set of functions.

Script 2 reveals how the line reading procedure operates and routes the data consumer's computational code line by line to the successive layers. Routing to corresponding functions within the Controller or Handler layers is handled by a local procedure called ***getRuleValues***. In Script 2 line 4 it shows a semantic concept of extracting related information from the command obtained in line 2. The procedure has two arguments; first to take the row and second to determine which column of the Table 1 should be taken by iterating through the rules global variable (*ruleLines*). Ultimately, it returns the reaction procedure defined within the Controller layer for further execution. ***isVar*** checks if is no known command or reaction for the command part of line 2.

Event listener for the code submission target is other key functionality in TOTEM SDK. Event Listeners are common methods in JavaScript which set up a callback function that will be called whenever the specific event is delivered to the action target. This procedure, which is described in Script 3 in pseudo-code, is executed

Algorithm 2 Line reading

```
1: procedure LineReader (line of code)
2:     command ← command part of the line
3:     assignment ← rest of the code line
4:     reaction ← getRuleValues(command, "REACTION")
5:     isVar ← true, if command is known; Otherwise false
6:     if reaction exists and defined then
7:         Go to Controller
8:     else
9:         if isVar == True then
10:            Go to Execution Handler
11:        else
12:            Go to Error Handler
13:        end if
14:    end if
15: end procedure
```

before the line reading function operates. It is triggered when users submit the codes.

The callback function takes the entire submitted code, separates it line by line and stores each line as an element of a pre-defined local array (*codeValueLines*). The reason is to deliver each command separately in one piece to the lineReader function. However, some commands such as control statements are more than a single line of code. Therefore, the consecutive lines related to a single command should all be considered as one. This will be managed by the first if statement. In the second if statement it checks whether the next line of command is an initial splitted line or a block of lines belongs to a sole command. Then, it calls the lineReader function with the proper argument.

5.2 Execution and Testing

To verify that the developed SDK can analyze the submitted code and produce the expected results, a simple testing scenario is conducted. Script 4 illustrates a simple code snippet in potential users' codes

Algorithm 3 Listener for code submission

```
1: procedure CALLBACK FUNCTION IN EVENTLISTENER
2:     userCodeValue ← user's submitted code
3:     codeValueLines ← array of userCodeValue lines
4:     loopLines ← an empty array
5:     statementLine ← an empty string
6:     for each line in codeValueLines do
7:         if command is of Control Statement type then
8:             loopLines ← all loop lines
9:             Delete all pushed lines into loopLines
10:            statementLine ← concatenated loopLine lines
11:            statementLine ←→ deleted lines in 9
12:        end if
13:        if statementLine is empty then
14:            Execute lineReader (line)
15:        else
16:            Execute lineReader (statementLine)
17:        end if
18:    end for
19: end procedure
```

with basic commands, within which an Integer and a Float number values change inside a For loop without any error.

Algorithm 4 A simple code snippet as testing scenario

```
int a = 1, b = 2
int c = add (a, b)
float d = 4.1
for i = 1; i ≤ 3; i++ do
    c = add (c, i)
    d = sub (d, 0.1)
end for
```

Figure 5 displays the printed outputs of **LineReader()** as well as **userDefined_vars** global variable in the browser console, at the end of the execution in Script 4. As shown, the SDK detected the type

and extracted the corresponding reaction to each block of command, after being processed in the **LineReader()**. Moreover, the value of each defined variable is stored at **userDefined_vars** and gets updated by each command block execution. These data are the input of the Controllers layer.

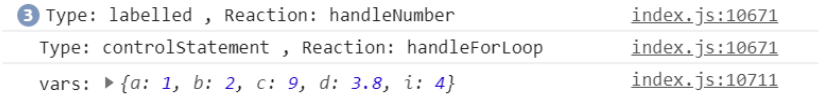


Figure 5: Controllers layer input data and final values of variables

Before termination, two more console logs are implemented to show the **TOTEM_operators** and **mapped_executions** global variables. Figure 6 shows the final value of these variables, storing all the assignments and operators within the user's code. This information is then used to estimate the amount of totem required to perform the computation using a pre-defined totem estimator table and execute the code on the computation part.

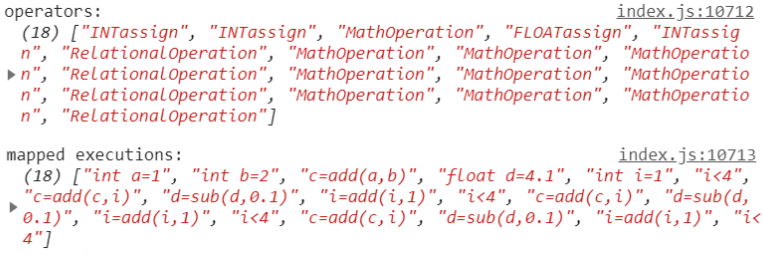


Figure 6: Totem_operators and mapped_executions global arrays after the execution in the testing scenario

Mapped_executions is an array with each element representing a single execution in the submitted code, further used in Customized Computational Framework (CCF) execution [20].

6 Conclusion

The main objective of this paper is the implementation of TOTEM defined SDK that focused on the TOTEM language format and rules.

The multi-layer architecture of TOTEM defined SDK is introduced and it has been implemented. An operational scenario emulating data consumers' computational codes, have been tested and verifies in the Implementation section. However this test has been conducted only to demonstrate the stability of the SDK framework, even though there is still room for the extension by adding new operations and allow other data types in the rules.csv file.

In future work, this TOTEM defined SDK should be integrated with the CCF implementation [20], upon a blockchain network to form the TOTEM architecture [1]. Future studies could also aim to replicate results in a broader scope such as creating a sole compiler that can be installed separately rather than interpreting codes deployed into the browser.

References

[1] Dhanya Therese Jose, Antorweep Chakravorty, and Chunming Rong. "TOTEM : Token for controlled computation: Integrating Blockchain with Big Data." In: (2019), pp. 1–7. DOI: 10.1109/ICCCNT45670.2019.8944855.

[2] Dhanya Therese Jose, Antorweep Chakravorty, and Chunming Rong. *Method for analyzing data using a blockchain, a data provider and a data customer therefor*. US Patent 11,121,874. Sept. 2021.

[3] S Haber and WS Stornetta. "How to Time-Stamp a Digital Document, Menezes AJ, Vanstone SA (eds) Advances in Cryptology-CRYPTO'90. CRYPTO 1990." In: *Lecture Notes in Computer Science* 537 (1991).

[4] Karim R Lakhani and M Iansiti. "The truth about blockchain." In: *Harvard Business Review* 95.1 (2017), pp. 119–127.

[5] Satoshi Nakamoto. "Bitcoin: A peer-to-peer electronic cash system Bitcoin: A Peer-to-Peer Electronic Cash System." In: *Bitcoin. org. Disponible en https://bitcoin. org/en/bitcoin-paper* (2009).

[6] Gareth W Peters and Efstathios Panayi. “Understanding modern banking ledgers through blockchain technologies: Future of transaction processing and smart contracts on the internet of money.” In: (2016), pp. 239–278.

[7] Nick Szabo et al. “Smart contracts.” In: (1994).

[8] Jeffrey Dean and Sanjay Ghemawat. “MapReduce: a flexible data processing tool.” In: *Communications of the ACM* 53.1 (2010), pp. 72–77.

[9] T Ramalingeswara Rao, Pabitra Mitra, Ravindara Bhatt, and Adrijit Goswami. “The big data system, components, tools, and technologies: a survey.” In: *Knowledge and Information Systems* 60.3 (2019), pp. 1165–1245.

[10] Wissem Inoubli, Sabeur Aridhi, Haithem Mezni, Mondher Maddouri, and Engelbert Mephu Nguifo. “An experimental survey on big data frameworks.” In: *Future Generation Computer Systems* 86 (2018), pp. 546–564.

[11] Xiayu Hua, Hao Wu, Zheng Li, and Shangping Ren. “Enhancing throughput of the Hadoop Distributed File System for interaction-intensive tasks.” In: *Journal of Parallel and Distributed Computing* 74.8 (2014), pp. 2770–2779.

[12] “ElR ancho. MapReduce explained. https : / / medium . com /@francescomandru / mapreduce-explained-45a858c5ac1d. 2019.” In: *mapreduce* (2019).

[13] Simon Holm Jensen, Anders Møller, and Peter Thiemann. “Type analysis for JavaScript.” In: *International Static Analysis Symposium*. Springer. 2009, pp. 238–255.

[14] Gregor Richards, Sylvain Lebresne, Brian Burg, and Jan Vitek. “An analysis of the dynamic behavior of JavaScript programs.” In: *Proceedings of the 31st ACM SIGPLAN Conference on Programming Language Design and Implementation*. 2010, pp. 1–12.

[15] Ravi Chugh, Jeffrey A Meister, Ranjit Jhala, and Sorin Lerner. "Staged information flow for JavaScript." In: *Proceedings of the 30th ACM SIGPLAN conference on programming language design and implementation*. 2009, pp. 50–62.

[16] "About Node.js, and why you should add Node.js to your skill set? http : / / blog . training.com/2016/09/about-nodejs-and-why-you-should- add.html. 2016." In: (2016).

[17] Marijn Haverbeke. *Eloquent JavaScript: a modern introduction to programming*. No Starch Press, 2018.

[18] Preethi Kasireddy. "JavaScript Modules: A Beginners Guide." In: *Viitattu* 20.2016 (2016).

[19] Xin Liu, Meina Kan, Wanglong Wu, Shiguang Shan, and Xilin Chen. "VIPLFaceNet: an open source deep face recognition SDK." In: *Frontiers of Computer Science* 11.2 (2017), pp. 208–218.

[20] Dhanya Therese Jose, Antorweep Chakravorty, and Chunming Rong. "Distributed computational framework in TOTEM architecture enabled by blockchain." In: (2020), pp. 83–88. DOI: 10.1109/ICCSE49874.2020.9201683.

[21] "https://nodejs.org/en/." In: (2020).

[22] "http://browserify.org/." In: (2020).

Paper IV: Integrating Big Data and Blockchain to Manage Energy Smart Grid - TOTEM Framework

Integrating Big Data and Blockchain to Manage Energy Smart Grid - TOTEM Framework

Dhanya Therese Jose[1], **Jørgen Holme**[1], **Antorweep Chakravorty** [1], **Chunming Rong**[1]

[1] Department of Electrical Engineering and Computer Science, University of Stavanger

Abstract:
The demand for electricity is increasing exponentially day by day especially with the arrival of electric vehicles. In smart community neighborhood project, it demands electricity should be produced at the household or community level and sell or buy according to the demands. Since the actors can produce, sell and also buy according to the demands, thus the name prosumers. ICT solutions can contribute to this in several ways such as machine learning for analysing the household data for customer demand and peak hour for the usage of electricity, blockchain as a trustworthy platform for selling or buying, data hub, and ensure data security and privacy of prosumers. TOTEM: Token for controlled computation is a framework that allows the users to analyze the data without moving the data from the data owner´s environment. It also ensures the data security and privacy of the data. Here in this article, we will show the importance of TOTEM architecture in the EnergiX project and how the extended version of TOTEM can be efficiently merged with the demands of the current and similar projects.

1 Introduction

The electricity infrastructure nowadays is lacking mechanisms to handle mass and concentrated power consumption. It is especially with the arrival of electric vehicles (EV), the charging demands and the peak over consumptions. To handle such peak hour issues without changing the existing structure of the grid, integration of renewable microenergy sources at household or neighborhood levels can be implemented. Therefore it is quite necessary to control and make use of potential prosumers and develop tools to efficiently manage household or neighborhood energy production and consumption. Proper mechanisms are essential to collect the combined microenergy sources, integrate community level power storage units, predict production and consumption, and allow sharing/trading of energy between households, neighborhoods, and communities. In order to address this kind of challenges, ICT-based smart solutions can be used. A data-driven decentralized energy system can efficiently contribute to the supply-demand market relationship between prosumers (consumers and producers).

In the Smart Community Neighborhood project the conventional end prosumers will produce the energy locally using environment friendly energy production and storage technologies and allow them to meet their own requirements or can sell it back to the grid. Electric vehicles (EV) batteries can also play an important role in a way that it can be used as an additional resource to the local energy storage system. These resources are capable of participating in balancing consumption during peak hours or fluctuations and also solve congestion problems. Upstream suppliers such as Distribution system operators (DSO) and Transmission system operators (TSO) require data about the power consumption and demands to manage the power distribution, transmission, and integration of community level resources to guarantee a reliable operation. The resources and the data streams from homes will be exponentially large, therefore to maintain such large and diverse data there should be a high information intensive data hub. Blockchain based technologies can be used for recording energy generated, shared/traded and stored into the storage level by the prosumers. Each transaction in the blockchain will be trans-

parent and secure [1]. In order to protect privacy and user's assets, blockchain uses cryptography [2]. Due to the openness of blockchain, every node in the network is transparent to the complete ledger, which may cause privacy leakage. In addition, digital assets cannot prove their ownership as easily as physical currency, crypto-techniques are required to prove the ownership of digital assets. Asymmetric encryption and hash function are the two techniques mainly used in blockchain. Finally to predict the peak hour consumption and production of energy from each household, neighborhood and community, machine learning techniques are effective and essential. A self organising and self optimizing, secure and privacy preserving community energy management system by utilizing the information intensive data hub, blockchain and machine learning is the aim of the smart community neighborhood project.

By obtaining energy usage patterns such as peak hour consumptions, production from each households/neighborhood and energy demands, the prosumers will get a better understanding about the current trend. It shows the analysis of data will improve the efficiency of the energy infrastructure. The analysis of data through conventional methods demands the data need to be transferred across the network, which is highly network intensive and due to security breaches data owners may be cautions to provider their data. TOTEM [3] [4], which stands for Token for controlled computation is a patented framework (US Patent No.:US11121874B2) which provide a decentralized solution for these concerns about the data transfer and analyses. This framework combines both the blockchain technologies and the big data systems. Each transaction through blockchain is secured and tamper-proof but having limitations on handling large data and parallel computations. Big data technologies provide solution for computation of large data through parallel computing. These properties of both blockchain and big data technologies complement and open a new direction for computing large datasets without moving the data across the network. The TOTEM project ensures data security by allowing organizations to open their data centers and allow disruptive business models. This framework can be used in the smart community neighborhood (EnergiX project) as well for enabling data hubs from different communities to open up their database for allowing data

analysis from different communities or users without moving their data. In this article, a detailed explanation about how to utilize TOTEM for the EnergiX project and also shows the scope of an extended version of TOTEM which is dealing with the multi provider architecture.

The structure of this paper is arranged as follows, Section 2 contains the related or relevant works and Section 3 shows the background for the present paper and a detailed explanation about TOTEM architecture. Section 4 shows how the TOTEM can be incorporated with the infrastructure. A description of the extended TOTEM architecture and its implementation is presented in Section 5 and Section 6 respectively. The conclusion is given in Section 7. .

2 Related Works

The blockchain and big data technologies were combined for various purposes. Possibilities of using blockchain on big data systems such as decentralized management for private data, IoT communication, resolution of digital property, and public institutions were discussed in [5]. For reinforcing the security of big data platforms, a blockchain based access control framework is proposed in [6]. This framework achieved objectives such as user-driven, transparency, fine granularity, pseudonymity and unlinkability. However, while adopting blockchain technology to handle access control functions additional critical issues were emerged in the framework. A blockchain access control ecosystem which ensures a better way to manage access control of large datasets and how to avoid data breaches is proposed in [7]. The architecture built on a private and permissioned blockchain for a decentralized security system. Blockchain technology provides solution to the challenges associated with traditional and centralized access control because it ensures the data transparency, traceability, secure data sharing, auditability, and data self-sovereignty for the owner. A scalable blockchain-based big data storage for distributed computing is proposed in HBasechainDB [8]. HBasechainDB makes it easy for organizations which have Hadoop ecosystem-based business logic to accommodate blockchain. HBasechainDB which built on the

Hadoop ecosystem, also inherits efficient big data processing as well. In [9], proposed "Mystiko", a new blockchain storage that is built over the Apache Cassandra distributed database to incorporate big data. Mystiko ensures high scalability, transaction throughput and availability.

So far, we have discussed the available integration of blockchain and big data technologies. The TOTEM architecture is a patented novel architecture which combines both blockchain and big data and enables secure handling of the data without moving the data over the network. The data user uses TOTEM SDK to create a MapReduce code for computation, which will be executed within the data owner's environment. This computational system prevents the execution of any malicious functions in the user code, by putting constraints on computational operations. A pre-defined totem will be assigned to authorized users, based on their computational needs. A smart contract performs pre-checks on user submitted code and associated totem value, by using a totem estimator table to determine the required totem for executing the given code. A totem manager and an updater is introduced to coordinate computation of the user code until it exits gracefully, or the assigned totem gets exhausted. In paper [10], it deals with the proof of concept of the new components such as totem manger and updaters introduced in TOTEM framework.

Machine learning is a part of artificial intelligence that deals with the study of algorithms which allows the system to learn by itself and improve the results with experience [11]. Machine learning plays an important role in load balancing and the prediction of electricity usage. This will help in efficient and effective working of the infrastructure when it comes to the prosumers demand, production, supply, or storage management. Some of the examples that show the necessity are as follows; [12] deals with the application of feature selection methods for the purpose of predicting household energy consumption. In this paper they follow a two-stage framework for identifying candidate features based on literature studies and data characteristics of a load profile and then it selects a subset of relevant features using the feature selection methods. Another one [13] deals with Short Term Load Forecasting Using Smart Meter Data which is a generalization analysis. An application of machine learning for

the energy management of loads and sources in smart grid networks is employed in Anderson et al. [14]. Rudin et al. [15] proposed a framework where machine learning can be used for the prediction of failures of the system components. In some other works [16] and [17], with the advantage of machine learning techniques, malicious activity prediction and intrusion detection problems are analyzed at the network layer of the smart grid communication system. Also in [18] by using the distributed spare attacks model, which is proposed in [19] and machine learning algorithm, they worked on the detection of false data injection attacks. These papers show the relevance of data analysis in the infrastructure with the available dataset from smart meter data or household data which is not secure, and privacy preserved. The relevance of blockchain technology comes here because it guarantees secured transactions. A recent survey suggests the Decentralized applications (DApps) with blockchains promise no trust on authorities and overcomes the key challenges of security and privacy problems [20]. Thus blockchain in the EnergiX project act as a platform for trading the electricity according to the demand of prosumers, in a secured manner. Since the big data plays a vital role in decision making for prosumers and blockchain act as a trading platform, it is relevant to integrate both technologies for a better and efficient working of a secure environment. The properties of these two technologies can complement each other, which would enable a new way of computing upon large datasets in a secured manner.

3 Background

In this section we will discuss the background technologies used in the TOTEM architecture, to get a clear picture while explaining the concepts in the upcoming sections. Big data systems and blockchain technologies are the two main components in the TOTEM framework. Blockchain allows authorized users to perform their required analyses on the data owner's data, without moving the data across the network. An overview of the technologies used in the framework is given below.

3.1 Big Data Analytics

Big data is the term that represents the datasets that are too large to be efficiently interpreted, collected, managed, and processed, using traditional methods or mechanisms [21]. Some of the main features of big data such as volume, variety, velocity and veracity [22] are quite good enough to describe the properties. These formerly mentioned features will denote the size of generated data, the types and nature of the given data, how often the data is generated or speed of the data generation and processing, and the data quality or value, respectively [23]. The proper method or strategy of analyzing such large volumes of data is known as big data analytics. Apache Hadoop, Spark, MongoDB, Cassandra and Neo4j are some of the popular frameworks commonly used for big data analytics. Hadoop is one of the leading open source frameworks on the list which can run on premises or in the cloud. However, each framework has its own advantages and drawbacks.

Hadoop is a framework that is capable of distributed processing of large datasets with clusters of computers and it is an open source implementation based on a programming model called MapReduce [24]. The Hadoop framework mainly consists of two layers such as the Hadoop Distributed File System (HDFS) [25] and previously mentioned distributed processing mechanism, MapReduce. The HDFS consists of a master-slave architecture, where the master node or the NameNode maintains the file systems metadata. The files are divided into fixed-size blocks and are stored in the slave node or DataNodes. Mapping of blocks to a particular DataNodes is determined by the NameNode according to some of the features such as ease of access and free slot. DataNodes are responsible for read-write operations in the file system. DataNodes are also responsible for the creation, deletion and replication of blocks, but all these operations are based on the instructions given by the NameNode. It will also send heartbeats to the NameNode periodically to indicate that the corresponding DataNode is currently active. The secondary NameNode is the node that keeps checkpoints of the file system metadata on the NameNode in the HDFS. Functions such as map and reduce are the two tasks performed in the distributed processing framework. Receiving the

input data and converting it into granular structure done by map function, and the output of this function will be tuples with key-value pairs. These results from the map function will be taken as the input for reduce function. In reduce function, the inputs combine and groups those tuples according to a unique key value. The results from the functions are stored in the distributed file system. The MapReduce framework has a JobTracker [26], which is responsible for monitoring the availability of the resources and allocating resources. It is also responsible for scheduling tasks for TaskTrackers. TaskTrackers will compute the tasks and provide the status information back to the master or the JobTracker at regular intervals of time. If a TaskTracker goes down, the JobTracker will immediately reschedule the corresponding failed task to the next available TaskTracker and if a JobTracker goes down, the entire process will halt.

3.2 Blockchain Technology

The blockchain is an open distributed ledger that can record each transaction through it in a very efficient manner [27]. All the transactions recorded in blockchain are transparent to all users in the corresponding blockchain network and these transactions are tamper-proof, which means they cannot be modified. There is no central node to control the entire network as in a centralized system, but it has peer-to-peer communication between nodes and therefore it is called a decentralized system. Secure hash functions are used in the blockchain for storing the transactions. The nodes can individually verify the transactions. To timestamp digital documents secure hash functions were used in 1991 by Haber and Stornetta [28]. After that, in 1992 Merkle tree was used for time-stamping several documents into one block. Later in 2009, it got more attention, when Satoshi Nakamoto introduced Bitcoin [29]. Bitcoin can be described as a peer-to-peer permissionless blockchain, which means it is completely decentralized. An example for public blockchain is Ethereum. The public blockchain is also known as permissionless blockchains because the participants in the network are anonymous and there are no restrictions in joining the network for the verification process [30]. Another type of blockchain is private or permissioned blockchain

which requires permission to join the network. It is only restricted to users within a particular organization or group of organizations and only selected nodes by the blockchain consortium can participate in the verification process. Hyperledger Fabric and Ripple are examples of permissioned or private blockchain. Adding transactions to a block after verification and appending them to the existing chain can be done by any of the nodes in the network depending on the type of blockchain. To avoid conflict between nodes on adding the same transaction, an agreement should be made between the nodes. It can be done with the consensus algorithm which chooses the right node to append the new block. In general, these consensus algorithms can be proof-based or voting-based algorithms [31]. Blockchain can enable smart contacts which were proposed by Nick Szabo [32]. Blockchain-based smart contracts are computer programs that can executed in a decentralized manner. The transactions will occur only when the requirements or conditions in the smart contract are satisfied. The computational complexity of the program plays an important role in efficient execution, for instance, Ethereum introduced the 'gas' concept to tie up with execution complexity of smart contract to financial limitations [33].

3.2.1 Hyperledger Fabric

Hyperledger is an open source project created to enhance blockchain for enterprises. The project started in 2015, hosted by the Linux Foundation and is a collaborative effort between many different companies. It comprises over 230 organizations and several projects, including IBM's Hyperledger Fabric. *Hyperledger Fabric*[8] is an enterprise grade permissioned distributed ledger framework created by IBM. It focuses on a modular and configurable architecture, allowing it to meet the requirements of many different use cases such as banking, insurance, healthcare and energy trading. Fabric supports smart contracts written in general purpose programming languages including Java, Go and Node.js, which means users do not need to learn a domain specific language to write them.

[8] `https://hyperledger-fabric.readthedocs.io/en/release-1.4`

3.2.2 Channels

As previously mentioned, permissioned blockchains force users to be authorized before joining the network. Hyperledger Fabric allows for privacy and confidentiality through channels. A channel is formed by a consortium of organizations, which share a separate channel ledger and are free to transact as long as they conform to the policies defined on the channel. This allows for transparency among the members of the consortium, while still keeping their transactions private from outsiders. Note that the channel we describe here is known as an application channel. This differs from the system channel, which controls the configuration of the Fabric network. In this present work, we refer to application channels when mentioning channels. A channel ledger will comprise a world state and a transaction log. The world state represents the current state of the channel ledger, while the transaction log is the history of transactions that have to lead to the current world state, i.e., the transaction log is the blockchain. A channel will also logically host smart contracts, which in Fabric are written in chaincode, and may be invoked by applications that wish to interact with the ledger.

3.2.3 Peers and Orderers

The nodes that comprise a Hyperledger Fabric network are primarily peer nodes and orderer nodes, which cooperate to ensure that only proper transactions are committed to the ledger. Peers may take on different roles in the network, however, for now it is enough to know that some peers act as endorsing peers, which will endorse a transaction before sending it to the orderer. The following steps are taken to commit a transaction to the ledger.

(1) A transaction proposal is sent to each endorsing peer, which will run and subsequently endorse the transaction before sending it to an orderer.

(2) The orderer will ensure that the transaction is endorsed by the necessary peers. Then it will add the transaction to the next block and distribute it to all the peers in the channel.

(3) Each peer will then inspect the block to validate that every peer has received the same result. Upon successful validation, the peers will commit the block to the ledger.

Every peer will additionally host a ledger instance for each channel in which it is participating.

3.3 Private data collections

When a transaction is committed to the channel ledger, it is broadcast to all peers and orderers participating in the channel. As previously mentioned, every peer holds a copy of the channel ledger. This means that any organization may access all data that are transacted on the channel if they are a member.

Hyperledger Fabric utilizes channels to keep transactions transparent between a subset of organizations while keeping them private from the rest of the network. However, consider the case where another subset of organizations on a channel needs to keep their transactions private. One possible solution would be to create separate channels for each of these cases, although, this would surely clutter the network, increasing complexity and introducing unnecessary configurations. Hyperledger Fabric introduced *private data collections*[9] to aid such cases. When using private data collection, there are two types of data being transmitted: actual private data and a hash of the data. The actual data are only sent to peers from organizations that are authorized to see it, using a *gossip protocol*[10] . The private data is stored in a separate private database on the authorized peers and are accessible from chain code on these peers. The orderer is not involved in this process, keeping it private from orderer organizations as well. The hash of the data is treated as a normal transaction, meaning that it is endorsed by peers, sent to the orderer for validation and broadcast to every peer on the channel. Note that we are required to set up anchor peers for this communication to work. Anchor peers

[9] `https://hyperledger-fabric.readthedocs.io/en/release-1.4/private-data/private-data.html`

[10] `https://hyperledger-fabric.readthedocs.io/en/release-1.4/gossip.html`

are used for cross-organization communication. This is essential when using private data collections since the gossip protocol communicates private data peer-to-peer between authorized organizations.

4 TOTEM architecture with two scenarios

TOTEM, Token for controlled computation is an architecture that integrates blockchain technologies and big data systems to take advantage of both the technologies for secure and privacy preserved data analytics. It is a three layer architecture as shown in Figure 1, with a blockchain consortium, computation layer and storage layer.

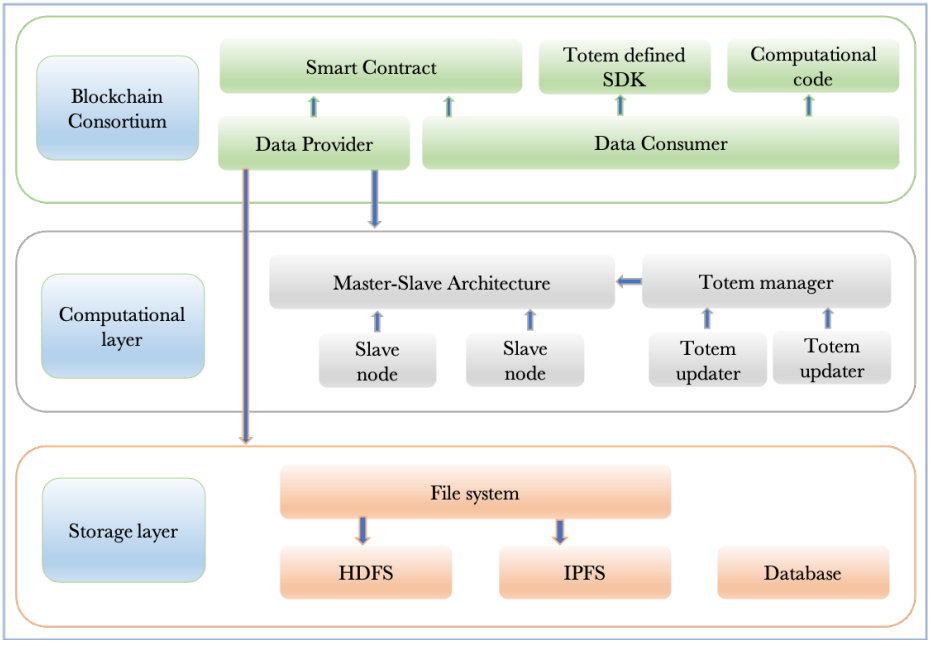


Figure 1: Three layers of TOTEM architecture

(1) The blockchain consortium deals with the blockchain network that connects the data user and the data provider. Through an SDK, the data user will submit the computational code for analysis and the smart contract will decide whether to continue or stop the execution depending on the totem value. The entity

that continuously monitors and controls the computational complexity of the code given to the system for data analysis is also known as totem. For each authorized actor in the system, a pre-defined totem value will be assigned according to the computational needs.

(2) The computational layer is where the actual computation takes place and is in the data owner's environment. At the time of code execution, depending upon the computational complexity, the totem value assigned will get reduced in each step of execution and will continue until the final step of the code or when the totem value exhausts. We have demonstrated a Hadoop master-slave in the figure and in order to monitor and update the live totem value, there is a totem manager at the master level and totem updaters in each slave nodes.

(3) Storage layer contains the actual data collected from the households, electric car consumptions, wind farms and other prosumer sources. It can be a file system such as HDFS and IPFS (Inter-Planetary File System)) or any database.

In the TOTEM architecture, the blockchain consortium has two main actors: a data provider, which can be an organization or group of organizations together and a data customer, which can be a single user or organizations. Authorized data customers can execute their own opcodes in the currently available datasets, without accessing the data but by sending the code across the network towards where the actual data resides. The data provider is the one who provides the metadata of their own data to the blockchain and it will provide the required resources for the computation of the opcode given by the authorized users for analyzing the dataset present in the data provider environment. Data providers are also responsible for deploying the necessary smart contracts. The smart contract will do a pre-check on the code before sending it into the data provider's environment for actual execution. It monitors the infinite loops for malicious functions in the code submitted by the data customer for execution. The monitoring pattern should be in a standardized format and as a blockchain consist of multiple data providers, the smart contract must

be a collective effort of all the data providers in the network. The computational layer and storage layer will be in the data provider's environment. The computational layer consists of a master-slaves architecture and totem value monitoring and updation according to the computation complexity is done with the components; totem manager and updaters. In the blockchain network, for a user to join, proper authorization is required. In Hyperledger Fabric we have a Membership Service provider for membership. As we mentioned before, data providers will publish the metadata of the dataset they own and data consumers can view that information and send requests to analyze that particular dataset with data consumers' particular code to the data provider. It is required to have enough totem value in the data consumer. The workflow of the TOTEM architecture which involves a single data provider and data consumer is explained in the following subsection 4.1 Scenario 1.

4.1 Scenario 1: With single data provider

Consider that we have a Community neighborhood-1 with a Data Hub(C1DH) and a researcher (R), who would like to analyze the data available in the Data Hub for a better understanding of the power generation and usage in this particular community. According to the TOTEM architecture, the two actors in the blockchain consortium are C1DH (data provider) and R (data consumer). Assume that both authorized users having registered and have proper certification from a certificate authority (CA) in the blockchain network. C1DH collects and stores all the information from the community smart home, electric vehicles and other resources that produce or consume electricity. It contains information related to the electricity generation, usage, and storage of excess electricity by each household or resource. It may contain sensitive data from household smart meter which need to be handled with privacy preserving mechanisms, such as filtering those data fields accordingly before proceeding for analysis. Data Hub contains the data related to peak hour consumption of each household or entire community, excess power that can be utilized later by selling to the required consumer, who all are the potential prosumers in the community, etc.

Assume that the researcher wants to know about the peak hour consumption of an entire community for a particular time period for the research purpose. The TOTEM architecture contains a TOTEM defined SDK, where the data consumer can write their particular code through the SDK. The TOTEM defined SDK contains a set of rules and formats to follow for writing the code. The researcher will write the query, S1 and submit it for analysis. The query, S1, will go through the smart contract where a preliminary check for the required totem value for analyzing the particular code for the particular dataset is enough or not. Once the preliminary check is done and has enough totem value to proceed, it will be given to the Community Data Hub, C1DH for the actual execution of the code on the required dataset. After each step of opcode execution, the totem value will be reduced according to the complexity of that particular opcode and the available balance will get updated. Also, after a set of opcodes, the available totem value will be again checked and confirmed with the smart contract and recorded as transactions. This execution will continue until the final result is produced or exists immediately if the totem value is exhausted in between the execution. Figure 2 given below shows the workflow of this scenario.

4.2 Scenario 2: With multiple data provider

The second scenario is to assume that the researcher, R wants to know about the combined peak hour consumption of both the community, C1 and C2. The information related to the first community such as household data, electricity prosumers data and electric car energy data is available in C1DH and the second community is available in Community neighborhood-2 with a Data Hub (C2DH). Since the data is not sent across any of the networks, and we aim for a secured and privacy preserved driven data analysis, we need to solve this situation without sharing the actual data from one community to the other. Resemble scenario 1 for scenario 2 also with the TOTEM defined SDK, the query S2 will be submitted by the researcher, R and it will go through a preliminary check for the totem value status. If the researcher is authorized to analyse the dataset available in both C1DH and C2DH then the S2 will be given to those Data Hubs. Both

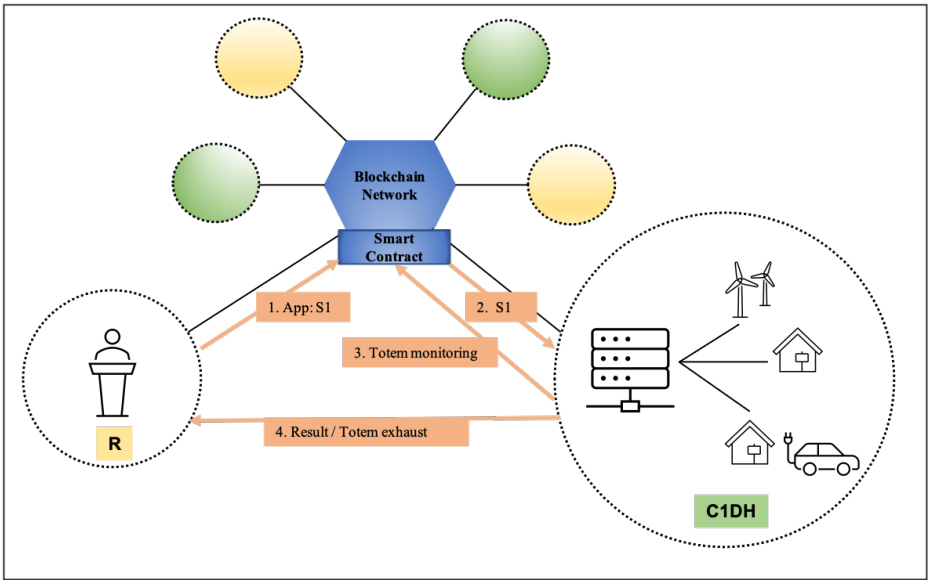


Figure 2: Scenario 1: With single data provider

the Data Hub will simultaneously monitor and update the totem value, and the value will be recorded as transactions in the blockchain network. Here the entire execution will stop immediately if the totem value gets exhausted in between the execution. Otherwise, the final result of both C1DH and C2DH will be combined with proper communication between the Community Data Hubs and the result will be published to the researcher. Figure 3 given below shows the workflow of the scenario with multiple data providers. Dealing with multiple providers is not discussed in the original TOTEM: Token for controlled computation framework. Thus as an extended version of the TOTEM architecture shows how to deal with multiple data providers in the TOTEM architecture in the following Section 5.

5 Extended architecture : with multiple data providers

As mentioned earlier, we need to assume that an opcode has already been submitted to both C1DH and C2DH through the TOTEM

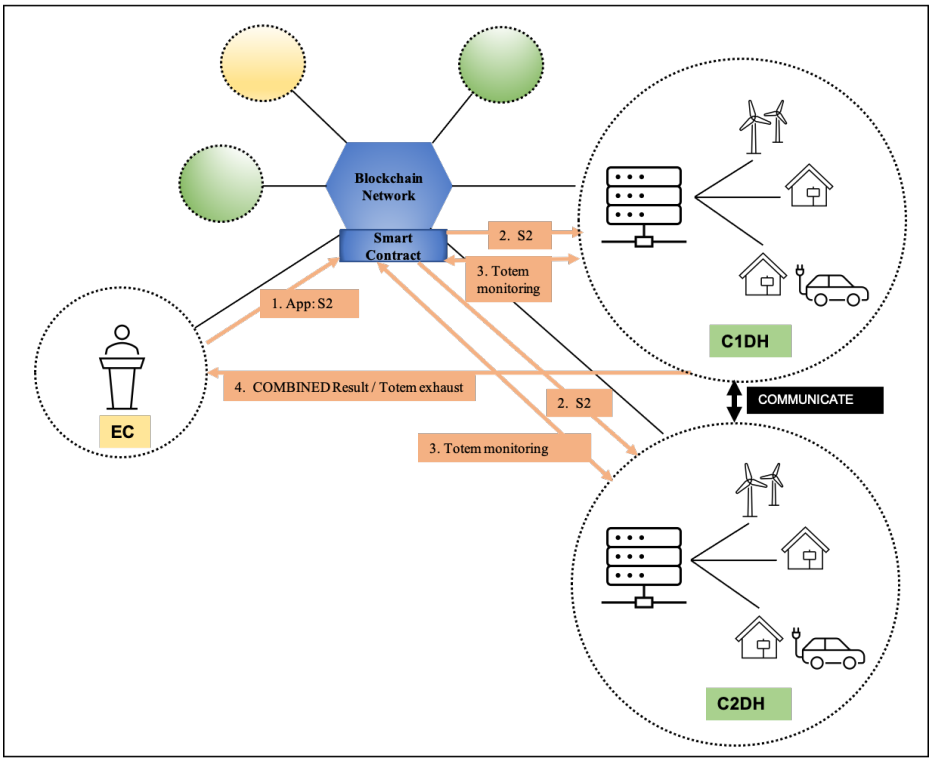


Figure 3: Scenario 2: With Multiple data provider

architecture network by the data consumer, R. Our goal is to retrieve the combined result of the submitted opcode without sharing the data occurring between the data providers. Our proposed solution will consist of allowing each individual data provider to execute its given code, and then store the respective result in Hyperledger Fabric. We achieve this by provisioning computational infrastructure in the form of Docker containers at the location of the data providers. These containers will form the Hadoop cluster on which we will run our computational code and will be provisioned with the help of Ansible.

Ansible [11] is open source software that automates the process of IT infrastructure and application deployment. An Ansible managed infrastructure will consist of one or several control nodes, which will have Ansible installed on them, and managed nodes that will

[11] `https://www.ansible.com/overview/it-automation`

receive instructions from the control nodes. The system allows us to construct playbooks, which essentially are recipes for tasks that need to be performed at a remote location. By using a push configuration, Ansible does not require any client side installation, meaning, the data providers do not require any additional software to perform the given tasks. Rather, Ansible uses SSH with public key authentication and requires the data provider to grant access to the data consumer before any commands can be pushed. Our system utilizes Hyperledger Fabric's permissioned blockchain to govern and grant access to data providers' resources.

An illustration of our system for governing access between a data consumer and a data provider is shown in Figure 4. Authorized data consumers will have to obtain a one time code (OTC) from the blockchain and send this along with their public key to the data providers' resources to authenticate themselves and gain access.

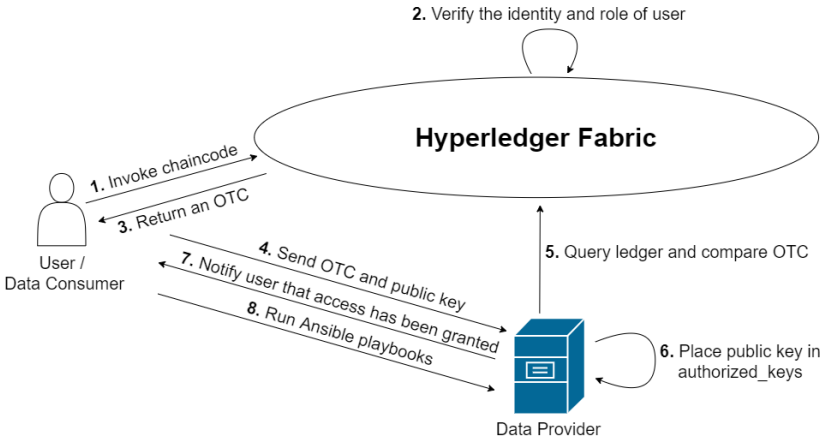


Figure 4: Architecture for governing remote resource access

The data provider will query the blockchain to make sure that the data consumer's one time code is legitimate, and subsequently add their public key to their list of authorized keys. Once this is done, the data consumer will be allowed to push Ansible commands to deploy the necessary infrastructure needed for running the opcode. Our playbooks will provision the infrastructure, run the remote computational code, and store each respective result in Hyperledger

Fabric. However, simply putting the result in a state on the ledger will expose the data to all participating peers and orderers, which is undesirable in scenarios where the data providers require total privacy of their data. Therefore, we propose the usage of private data collections (PDCs) for storing the results. An illustration of our proposed architecture for using private data collections is shown in Figure 5.

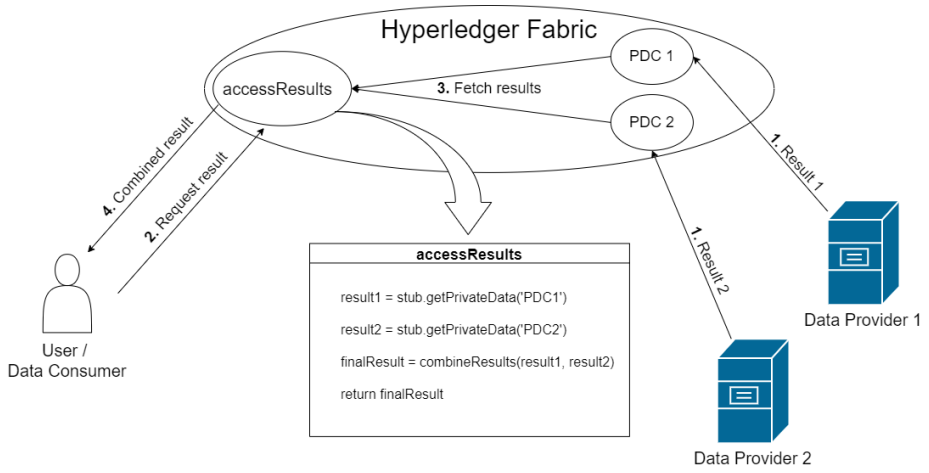


Figure 5: Scenario 2: Access result with Multiple data providers

In the Figure 5, we observe 1) two data providers who put the results in their respective private data collection, once they complete the required computation. Hence the results residing in their own collections, the data consumer can 2) invoke the accessResults chaincode, which is shown as pseudocode in the figure. In short, the chaincode will 3) fetch results from the private data collections and then perform the necessary operations to combine the results. For example in our scenario, we have results from a wordcount job, meaning the function will transform the results to JSON objects, sum values with matching keys, and add key-value pairs which are unique to an object. When the results are combined, the chaincode will 4) return the final JSON object as a binary data stream. We will demonstrate an implementation of this system by deploying Hyperledger Fabric

on Microsoft Azure cloud using both a single Kubernetes cluster and a distributed multicluster environment.

6 Implementation and Results

To illustrate a scenario in which we can demonstrate the aforementioned system in a truly distributed way, consider the following: Data consumer R residing in Stavanger wants to obtain a combined statistical result from two community data hubs, C1DH and C2DH, residing in Spain and the Netherlands respectively. In this case, the system will have to work despite a massive regional difference, while permitting each data provider absolute privacy and control of their own data. The difference in rules and regulations pertaining to data management for these different regions may be vast, however, this system will allow any participant to comply with their respective region's rules and provide authorized data consumers an opportunity to compute their own code. First, we will set up the system on a single cluster residing in one region, followed by a demonstration of a distributed multi cluster environment spanning two different regions. *Azure*[12] is a cloud service created by Microsoft. It offers a plethora of services, including the *Azure Kubernetes Service* (AKS). AKS offers a fully managed Kubernetes service and allows users to easily scale their infrastructure when needed. In this section, we will utilize AKS to provide a multi-node Kubernetes cluster.

6.1 Method 1: Deploying Hyperledger Fabric on a single Kubernetes cluster

Provisioning a Kubernetes cluster in Azure using the *Azure portal* is a simple eventuality. In the portal, first *Create a resource* and choose *Kubernetes Service*. Here we specify some basic settings such as which subscription to use, a resource group, a cluster name and a region. Later we experiment with regions to create a multi-cluster distributed Hyperledger Fabric network.

[12] `https://azure.microsoft.com/en-us/`

After basic configurations, we specify the size of our *node pool*. The node pool will contain the nodes which will host the Hyperledger Fabric infrastructure. Here, it is essential that we first consider the number of nodes it takes to run our Hyperledger Fabric network. Our network comprises three peer organizations with one peer each and one orderer organization. This results in three peers, three certificate organizations and one orderer; each requiring one node, i.e., we required seven nodes in our node pool to launch the network.

Furthermore, AKS allows us to choose the size of each node. There are different specifications for each choice, allowing the user to consider their needs for the number of CPUs, size of RAM, number of disks, etc. We consider that the peer node must contain at least two disks since we need one disk for the peer and one disk for holding copies of ledgers. It is sufficient for us to use the smallest Virtual Machine(VM) size, which contains four disks.

In the *authentication* tab, we turn off *role-based access control* for a smoother and easier way for us to connect to the cluster. For *networking*, *integrations* and *tags*, we use the default values. We are now ready to review and create our cluster. During setup, we observe that Azure restricts the number of CPUs one can have in a single region. In order to have a sufficient number of CPUs, we choose the pay-as-you-go plan, which allows us to have up to ten CPUs running in each region.

Once the cluster has been successfully reviewed, we can create it and subsequently connect to it from the *Azure Cloud Shell*. This shell has the `kubectl` client pre-installed, which is the client used for interacting with a Kubernetes cluster. However, we must first configure `kubectl` to connect to our cluster. We do this with the `az aks get-credentials` command and specify the name and resource group pertaining to our cluster.

To deploy and operate the Hyperledger Fabric network, we use a tool called *PIVT*[13]. PIVT provides *Helm charts*, to facilitate launching a Fabric network, as well as interacting with it. Helm[14] is a package manager for Kubernetes, which manages charts. The charts provided

[13] `https://github.com/hyfen-nl/PIVT`

[14] `https://helm.sh/`

by PIVT can be used for

- Configuring and launching a Hyperledger Fabric network.
- Populating the network declaratively with channels, peers and chaincode.
- Adding new peers to run networks and updating channel configurations declaratively.
- Backing up and restoring the state of the network.

Before we are able to use any of PIVT's functionalities, we must install the prerequisites. We first use the `wget` command to download all the binaries and subsequently add them to our path. When all the prerequisites are added, the next step is to launch the network from the Azure Cloud Shell. Then create the channel and install chaincode using PIVT's helm charts. However, when deploying our network in the cloud it is essential that we use a proper load balancer to activate external IP addresses for our services, i.e., we need external IP addresses for our peers, certificate authorities and orderers such that users may interact with them. Using PIVT, we may activate this behavior by passing the `peer.externalService.enabled` and `orderer.externalService.enabled` flags, and setting them to `true`. This tells PIVT to include the definitions of *externalServices*.

In Figure 6, observe the *External IP* column. Here, we see our external services obtaining IP addresses for external access. In the figure, we also observe that the status of the external orderer's IP is `pending`. This is because Azure is working to assign a proper IP address, which may take a few minutes. Once all external services have received an IP address, we can access the network by updating our connection profile with the external IP addresses. For example, we can access the Stavanger peer using `51.104.146.139:7051`.

After completing these steps, we deployed a functional Hyperledger Fabric network across several nodes in Azure using AKS. However, we are only utilizing one cluster. In a real-world scenario, organizations might need to host their infrastructure (peers, certificate authorities, etc.) in the cloud provider of their choice or on their own premises.

This would require multiple clusters, possibly hosted in different parts of the world, communicating with each other to form a single Hyperledger Fabric network. To demonstrate this, we are using PIVT to deploy our Hyperledger Fabric network on two AKS clusters, residing in different regions.

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)
hlf-ca--netherlands	NodePort	10.0.146.235	<none>	7054:31701/TCP
hlf-ca--spain	NodePort	10.0.139.33	<none>	7054:31702/TCP
hlf-ca--stavanger	NodePort	10.0.11.54	<none>	7054:31700/TCP
hlf-couchdb--netherlands--peer0	ClusterIP	10.0.56.248	<none>	5984/TCP
hlf-couchdb--spain--peer0	ClusterIP	10.0.59.101	<none>	5984/TCP
hlf-couchdb--stavanger--peer0	ClusterIP	10.0.218.208	<none>	5984/TCP
hlf-orderer--ordererorg	ClusterIP	10.0.237.155	<none>	7050/TCP
hlf-orderer--ordererorg--orderer0	NodePort	10.0.174.59	<none>	7050:32700/TCP
hlf-orderer-external--ordererorg--orderer0	LoadBalancer	10.0.58.13	<pending>	7050:31417/TCP
hlf-orderer-lb	ClusterIP	10.0.41.193	<none>	7050/TCP
hlf-org-peer--netherlands	ClusterIP	10.0.160.198	<none>	7051/TCP,7052/TCP
hlf-org-peer--spain	ClusterIP	10.0.62.27	<none>	7051/TCP,7052/TCP
hlf-org-peer--stavanger	ClusterIP	10.0.210.152	<none>	7051/TCP,7052/TCP
hlf-peer--netherlands--peer0	NodePort	10.0.9.218	<none>	7051:30001/TCP,7052:30179/TCP
hlf-peer--spain--peer0	NodePort	10.0.210.56	<none>	7051:30002/TCP,7052:31846/TCP
hlf-peer--stavanger--peer0	NodePort	10.0.55.52	<none>	7051:30000/TCP,7052:31229/TCP
hlf-peer-external--netherlands--peer0	LoadBalancer	10.0.24.220	20.191.49.205	7051:30486/TCP
hlf-peer-external--spain--peer0	LoadBalancer	10.0.247.116	51.104.146.119	7051:32465/TCP
hlf-peer-external--stavanger--peer0	LoadBalancer	10.0.4.183	51.104.146.139	7051:32353/TCP
kubernetes	ClusterIP	10.0.0.1	<none>	443/TCP

Figure 6: External IP addresses from our services

6.2 Method 2: Deploying a Hyperledger Fabric network in a distributed cross-cluster environment

In order for us to separate our Hyperledger Fabric infrastructure, we first need to create another cluster in AKS. We use the same configurations described in Section 6.1, except we now have to consider a different number of nodes. We divide our network as shown in Table 1.

From the table, we observe that cluster 1 requires four nodes, while cluster 2 requires three. This is because a peer organization requires one node per peer, and one node per certificate authority, while an orderer organization only requires one node per orderer. Thus, we need four nodes for the two peer organizations in cluster 1, and three nodes for the peer and orderer organizations in cluster 2.

After we have created our clusters, we take inspiration from PIVT's *"Cross-cluster Raft network"* example[15]. Following this example, we

[15] `https://github.com/APGGroeiFabriek/PIVT#cross-cluster-raft-network`

Table 1: Overview of Clusters in Cross-Cluster Environment

Cluster	**Region**	**Members**	**Number of nodes**
1	North Europe	Spain (C1DH) and Netherlands (C2DH)	4
2	South-East Asia	Stavanger (R) and Orderer	3

first create two separate PIVT projects for each cluster by simply copying the files. Next, we alter the `network.yaml` and `crypto-config.yaml` files. In `crypto-config.yaml`, we have to specify *external peer organizations* for cluster 2, as well as an *external orderer organization* for cluster 1. Note that as opposed to PIVT's example, we do not enable TLS for our example. This is to simplify network communication for our proof-of-concept, however, it should be enabled in a production environment.

Another important difference when launching our cross-cluster network is the use of *host aliases* and *external host aliases*. Host aliases are simply domain names along with their respective Cluster IP, while external host aliases are domain names along with their external IP. These are needed in order for the two clusters to be aware of each other's external services. To collect these host aliases, we first launch the network in a *broken state*, which means to launch the network without starting the peer and orderer pods. Before these are started, we will collect host aliases and external host aliases using shell scripts provided by PIVT. For each cluster, it needs to be handled separately. Furthermore, we need to copy the external host aliases of cluster 2 into the host aliases of cluster 1, and vice versa. Now, each cluster has the proper addresses for communicating with its external resources. Note that we are again using `LoadBalancer` for granting external IP addresses to our services. Therefore, it is important that we wait until all services have obtained an external IP before collecting external host aliases.

Afterward, upgrade the network with the host aliases using a helm chart provided by PIVT. The setup of this cross-cluster example requires a number of operations to be performed on each cluster separately, and in the right order. To facilitate the process, and make it less error-prone, we wrote two shell scripts to automatically launch the network. When each component is running on both clusters, we can create channels and install chaincode using the same helm charts as before, and subsequently instantiate the chaincode using our Node.js script. Once these operations are done, we have a fully functioning multi-cluster Hyperledger Fabric network, with infrastructure residing in different parts of the world.

Note that this is a proof-of-concept implementation. For simplicity's sake, we only deploy two clusters in two different regions on Azure AKS. However, even though both data providers are residing in the same region, it would be fully possible for them to reside separately anywhere in the world.

7 Conculsion

In this paper, we have shown how the TOTEM architecture environment can be adapted into a smart community neighborhood project (EnergiX project) for data analysis in a secured manner. An extended version of the TOTEM architecture is also proposed as a solution if the data consumer demands a combined result from data providers as a part of data analysis. We have implemented the architecture as a part of the proof of concept. In the implementation, chaincode in the Hyperledger Fabric is used to manage the access to a remote resource and to the provisional computational resources as Docker containers that form the Hadoop cluster is done by using Ansible. The Hadoop cluster will perform the required computation in an isolated environment with remote resources. Enrolled users in the network obtain the OTC for authentication by invoking the chaincode. Private data collections in Hyperledger Fabric are used to ensure data privacy in a multi- provider scenario. Eventually, we demonstrated the system by deploying it using PIVT and Kubernetes in Azure using AKS on a single cluster and also across two clusters residing in

different regions of the world. This system also allows organizations with common interests to collaborate without the need for complete trust. All activity is kept private from the rest of the network, while all data is kept private between the data providers on the channel.

Some of the improvements that can be made in the current implementation are regarding the proper mechanism for securely transporting the OTC/Public Key and also extending the computational possibilities of the system as we use only a dummy computation in this present work. In future work, we consider this into account and will execute the necessary steps. For example, a relevant use of our system would help in training machine learning models across several remote datasets. This would require a different approach for combining results or private datasets, possibly, a multi-party computation (MPC) protocol could be used to realize this functionality. An obvious future direction would be to integrate this solution with the rest of TOTEM's proposed architecture, its performance analysis and platform efficiency. Our system tackles the issues in TOTEM regarding data and resource governance, running computations at remote locations, as well as safely returning combined results in a multi-provider scenario. Furthermore, the computational code would have to be transacted on the blockchain. One way to solve this would be to install and instantiate some chaincode that would evaluate the submitted computational code. If the code is deemed non-malicious, the chaincode will estimate a totem value, produce an OTC, and return them both to the user.

ACKNOWLEGEMENT
This research was funded by the Project number 267967: Energix of NFR (Norwegian Research Council) and Grant number 825134: ARTICONF of European Union's Horizon 2020 program.

References

[1] Harry Halpin and Marta Piekarska. "Introduction to Security and Privacy on the Blockchain." In: *2017 IEEE European*

Symposium on Security and Privacy Workshops (EuroS&PW). IEEE. 2017, pp. 1–3.

[2] Mingli Wu, Kun Wang, Xiaoqin Cai, Song Guo, Minyi Guo, and Chunming Rong. "A Comprehensive Survey of Blockchain: From Theory to IoT Applications and Beyond." In: *IEEE Internet of Things Journal* 6.5 (2019), pp. 8114–8154. DOI: 10.1109/JIOT.2019.2922538.

[3] Dhanya Therese Jose, Antorweep Chakravorty, and Chunming Rong. "TOTEM : Token for controlled computation: Integrating Blockchain with Big Data." In: (2019), pp. 1–7. DOI: 10.1109/ICCCNT45670.2019.8944855.

[4] Dhanya Therese Jose, Antorweep Chakravorty, and Chunming Rong. *Method for analyzing data using a blockchain, a data provider and a data customer therefor*. US Patent 11,121,874. Sept. 2021.

[5] Elena Karafiloski and Anastas Mishev. "Blockchain solutions for big data challenges: A literature review." In: (2017), pp. 763–768. DOI: 10.1109/EUROCON.2017.8011213.

[6] A Outchakoucht Jp Leroy H Es-Samaali, Nn Van, and R Nakagawa T Tanouchi S Kodama. "A Blockchain-based Access Control for Big Data." In: *Journal of Computer Networks and Communications* 5 (2017), pp. 137–147.

[7] Uchi Ugobame Uchibeke, Kevin A Schneider, Sara Hosseinzadeh Kassani, and Ralph Deters. "Blockchain access control ecosystem for big data security." In: (2018), pp. 1373–1378.

[8] Manuj Subhankar Sahoo and Pallav Kumar Baruah. "HBasechain DB-a scalable blockchain framework on hadoop ecosystem." In: (2018), pp. 18–29.

[9] Eranga Bandara, Wee Keong Ng, Kasun De Zoysa, Newton Fernando, Supun Tharaka, P Maurakirinathan, and Namal Jayasuriya. "Mystiko—blockchain meets big data." In: *2018 IEEE international conference on big data (big data)*. IEEE. 2018, pp. 3024–3032.

[10] Dhanya Therese Jose, Antorweep Chakravorty, and Chunming Rong. "Distributed computational framework in TOTEM architecture enabled by blockchain." In: (2020), pp. 83–88. DOI: 10.1109/ICCSE49874.2020.9201683.

[11] Tom M Mitchell and Tom M Mitchell. *Machine learning*. Vol. 1. 9. McGraw-hill New York, 1997.

[12] Aida Mehdipour Pirbazari, Antorweep Chakravorty, and Chunming Rong. "Evaluating feature selection methods for short-term load forecasting." In: *2019 IEEE International Conference on Big Data and Smart Computing (BigComp)*. IEEE. 2019, pp. 1–8.

[13] Aida Mehdipour Pirbazari, Mina Farmanbar, Antorweep Chakravorty, and Chunming Rong. "Short-term load forecasting using smart meter data: A generalization analysis." In: *Processes* 8.4 (2020), p. 484.

[14] Roger N Anderson, Albert Boulanger, Warren B Powell, and Warren Scott. "Adaptive stochastic control for the smart grid." In: *Proceedings of the IEEE* 99.6 (2011), pp. 1098–1115.

[15] Cynthia Rudin, David Waltz, Roger N Anderson, Albert Boulanger, Ansaf Salleb-Aouissi, Maggie Chow, Haimonti Dutta, Philip N Gross, Bert Huang, Steve Ierome, et al. "Machine learning for the New York City power grid." In: *IEEE transactions on pattern analysis and machine intelligence* 34.2 (2011), pp. 328–345.

[16] Zubair Md Fadlullah, Mostafa M Fouda, Nei Kato, Xuemin Shen, and Yousuke Nozaki. "An early warning system against malicious activities for smart grid communications." In: *IEEE Network* 25.5 (2011), pp. 50–55.

[17] Yichi Zhang, Lingfeng Wang, Weiqing Sun, Robert C Green II, and Mansoor Alam. "Distributed intrusion detection system in a multi-layer network architecture of smart grids." In: *IEEE Transactions on Smart Grid* 2.4 (2011), pp. 796–808.

[18] Mete Ozay, Inaki Esnaola, Fatos Tunay Yarman Vural, Sanjeev R Kulkarni, and H Vincent Poor. "Machine learning methods for attack detection in the smart grid." In: *IEEE transactions on neural networks and learning systems* 27.8 (2015), pp. 1773–1786.

[19] Mete Ozay, Inaki Esnaola, Fatos T Yarman Vural, Sanjeev R Kulkarni, and H Vincent Poor. "Sparse attack construction and state estimation in the smart grid: Centralized and distributed models." In: *IEEE Journal on Selected Areas in Communications* 31.7 (2013), pp. 1306–1318.

[20] Kaifeng Yue, Yuanyuan Zhang, Yanru Chen, Yang Li, Lian Zhao, Chunming Rong, and Liangyin Chen. "A survey of decentralizing applications via blockchain: The 5g and beyond perspective." In: *IEEE Communications Surveys & Tutorials* 23.4 (2021), pp. 2191–2217.

[21] Min Chen, Shiwen Mao, and Yunhao Liu. "Big data: A survey." In: *Mobile networks and applications* 19.2 (2014), pp. 171–209.

[22] Judith Hurwitz, Alan Nugent, Fern Halper, Marcia Kaufman, et al. "Big data for dummies." In: 336 (2013).

[23] Parth Chandarana and M Vijayalakshmi. "Big data analytics frameworks." In: (2014), pp. 430–434.

[24] Jeffrey Dean and Sanjay Ghemawat. "MapReduce: a flexible data processing tool." In: *Communications of the ACM* 53.1 (2010), pp. 72–77.

[25] Dhruba Borthakur. "The hadoop distributed file system: Architecture and design." In: *Hadoop Project Website* 11.2007 (2007), p. 21.

[26] Vinod Kumar Vavilapalli, Arun C Murthy, Chris Douglas, Sharad Agarwal, Mahadev Konar, Robert Evans, Thomas Graves, Jason Lowe, Hitesh Shah, Siddharth Seth, et al. "Apache hadoop yarn: Yet another resource negotiator." In: (2013), pp. 1–16.

[27] Karim R Lakhani and M Iansiti. "The truth about blockchain." In: *Harvard Business Review* 95.1 (2017), pp. 119–127.

[28] S Haber and WS Stornetta. “How to Time-Stamp a Digital Document, Menezes AJ, Vanstone SA (eds) Advances in Cryptology-CRYPTO’90. CRYPTO 1990.” In: *Lecture Notes in Computer Science* 537 (1991).

[29] Satoshi Nakamoto. “Bitcoin: A peer-to-peer electronic cash system Bitcoin: A Peer-to-Peer Electronic Cash System.” In: *Bitcoin. org. Disponible en https://bitcoin. org/en/bitcoin-paper* (2009).

[30] Gareth W Peters and Efstathios Panayi. “Understanding modern banking ledgers through blockchain technologies: Future of transaction processing and smart contracts on the internet of money.” In: (2016), pp. 239–278.

[31] Giang-Truong Nguyen and Kyungbaek Kim. “A survey about consensus algorithms used in blockchain.” In: *Journal of Information processing systems* 14.1 (2018), pp. 101–128.

[32] Nick Szabo et al. “Smart contracts.” In: (1994).

[33] Vitalik Buterin et al. “A next-generation smart contract and decentralized application platform.” In: *white paper* 3.37 (2014), pp. 2–1.

Paper V: Application of Artificial Intelligence in secure decentralized computation enabled by TOTEM

Application of Artificial Intelligence in secure decentralized computation enabled by TOTEM

Dhanya Therese Jose[1], **Antorweep Chakravorty** [1], **Chunming Rong**[1]

[1] Department of Electrical Engineering and Computer Science, University of Stavanger

Abstract:

TOTEM is a framework that allows users to execute their own code on access restricted datasets with controlled computation. It provides data security by restricting the exchange of data across the network, instead, the data will be executed within the data owner's environment itself. This ensures the data owners have full control over their data. The framework has been patented under US Patent No: US11,121,874 B2 in 2021. In this study, we are focusing on two areas, (i) how to improve the core features of the TOTEM architecture to an advanced level by introducing AI and (ii) the application of the TOTEM architecture in various sectors. Blockchain-enabled federated learning is getting more attention these days and its unique advantages while used within the TOTEM architecture are also discussed in this article.

1 Introduction

In this modern era, data could act as a key solution for increased dilemmas around the world if it is utilised properly. Data collected from various sectors and business sources have different stories to tell and patterns to represent. Data analysis helps to understand the data and figure out the pattern present in the dataset. Effectively summarizing this information gained from data will have the potential to solve the problems and shows a better future for the corresponding sectors or businesses. Data owners are reluctant to provide their data due to data security and privacy concerns. Ensuring data availability for analysis while providing privacy and security for the data is the basic concept behind the TOTEM architecture [1].

TOTEM allows authorized data users to analyze the data with their own code, without moving the data across the network, instead, the data analysis will be taken place in the data owners' environment. In order to avoid malicious functions which may appear in the data users' code, by putting constraints, a controlled computation will be performed in the TOTEM architecture. The data user will get an opportunity to write their own code through the TOTEM-defined SDK [2]. The user-written code will be preliminarily checked and once the code format is satisfied it estimates the totem value required to perform a particular code on the specified dataset available in the data owner's environment. The totem value is estimated according to the complexity of the user's submitted code. If the data user has enough totem value in their account to proceed with the actual execution, the code will be transferred to the data owner's environment and execution is performed. During the actual execution, the live monitoring of the totem value will also be carried out periodically with the help of TOTEM managers and updaters. It will continue until the execution is completed or when the totem value is exhausted.

TOTEM-defined SDK will check the format, and according to the keywords and data types in the code, it will estimate the totem value. It is demonstrated with simple computations such as basic loops and arithmetic statements to be carried out and estimate the value according to the operands and opcode complexity. However, if we need to analyze a dataset with the provided user code, the run time

code complexity depends on the operands as well as the size and type of the dataset. In this case for a more accurate totem value estimation, we need a better solution. Can machine learning be applied to the submitted code? What are the possibilities or changes additionally required for the user-submitted code? The present paper will discuss answers to the above questions. Apart from machine learning, the paper also discusses the possibilities of federated learning along with the multi-provider scenario in the TOTEM architecture provided in [3] and how it can contribute to the polished functioning of the system.

The rest of the paper is structured as follows. In Section 2 the background work is presented, and relevant works related to the present study are discussed in Section 3. For an accurate estimation of runtime complexity, how the machine learning is used in the TOTEM defined SDK and the possibilities of federated learning in the TOTEM architecture are described in Section 4. Finally, in Section 5 we have concluded this present study.

2 Background

2.1 Machine learning

Artificial Intelligence (AI) is the process of artificially making machines as intelligent as humans by learning and developing problem-solving skills through algorithms. Machine learning is a branch of artificial intelligence that deals with computational algorithms that are intended to replicate human intelligence by learning from available data [4]. Machine learning works on the principles adapted from computer science, probability, statistics, and artificial intelligence. Machine learning application is popular,and it is acquired in most of the industry sectors and is also applicable in day-to-day life. The application of machine learning algorithms heaps in various fields such as pattern recognition [5], entertainment, finance, and medical applications.

2.1.1 Machine learning Algorithms

Machine learning algorithms can be categorized into four: supervised learning, semi-supervised learning, unsupervised learning, and reinforcement learning.

Supervised learning is based on supervision [6] and train the machines using the labelled dataset and the machine predicts the output for the coming input data whereas in unsupervised learning there is no need for supervision and the machine is trained using the unlabeled dataset and the input sample alone is given for the learning process. Semi-supervised learning lies between Supervised and Unsupervised machine learning that uses large number of unlabeled data and less number of labeled data to train a model. Reinforcement learning works on a feedback-based process, in which an AI agent automatically explore its surrounding by hitting and trail, taking action, learning from experiences, and improving its performance. In reinforcement learning, there is no labelled data like supervised learning, and agents learn from their experiences only.

Classification and regression are two types of supervised learning problems. Clustering technique comes under the category of unsupervised learning. Text document classifier will come under the category semi-supervised learning. Positive reinforcement learning and negative reinforcement learning are also two types of reinforcement learning methods. In supervised learning regression algorithm is recommended for solving regression problems in which there is a linear relationship between input and output variables. Linear regression and Logistic regression are two such algorithms. Classification algorithm is recommended for solving the classification problems in which the output variable is categorical, and it is applied to test data to figure out which particular category it belongs to is called classification. Decision tree, Random forest, Support Vector Machines (SVM) and K-nearest neighbour are the algorithms for classification problems.

2.1.2 Deep learning

Conventional machine learning has limitations when it needs to handle the raw data obtained [7] as input. It demands domain expertise

and proper engineering work for feature extraction and transforming the raw data into feature vectors to input into the system to obtain or detect the classification category accordingly. Instead, the method that is used to feed the raw data as input and it automatically figure out the corresponding representation required for classification is known as representation learning. Deep learning is a representation learning method and it contains multiple levels for representation. It is designed with simplified non-linear functions to transform from one level to the next higher level. With the combination of such transformations, even complex functions can also be learned. Deep learning adopts an artificial neural network to execute complex computations on a large volume of data. Convolutional Neural Networks (CNNs), Long Short Term Memory Networks (LSTMs), and Recurrent Neural Networks (RNNs) are some of the popular deep learning algorithms.

2.1.3 Applications of Machine learning and Deep learning

Plenty of applications are there around for machine learning and deep learning in this modern era. It helps in many ways in almost all sectors of life. The applications are vast, and it starts from our basic needs such as food, transportation and shelter to high-level business strategies. In [8] a detailed review of machine learning and deep learning applications is presented. Prediction, semantic analysis, natural language processing and computer vision are the major domains that apply machine learning. Deep learning can be applied in various domains such as information retrieval, computer vision prediction, semantic analysis, natural language processing, and customer relationship management. Object recognition, object detection, and processing are subdomains of computer vision that also apply deep learning algorithms.

2.2 Federated leaning

Federated learning is a concept proposed by Google [9], in 2016. Federated learning is a machine learning technique that separately trains the learning algorithm on local datasets available in multiple edge devices or servers and the model obtained from each device

will be collected and aggregated to get the global model. The local data present in each device does not need to exchange to obtain a global model, therefore the entire process is without centralizing the data, which ensures data privacy. Differential privacy [10] is a generally used method to appraise the privacy leakage of sensitive data during the learning process. The concept of federated computation will describe the secure aggregation process of results obtained from different devices.

2.2.1 Federated computation

Federated computation is a map-reduce function for decentralized data with a privacy-preserving aggregation as a built-in function. In Federated computation privacy technologies used are on-device data, federated and secure aggregation, federated model averaging, and deferentially private model averaging. It follows privacy principles such as only aggregate data, ephemeral reports, focus collection, and not memorizing individual data.

2.2.2 Federated computation categories

Federated learning can be categorized into three; vertical and horizontal federated learning and federated transfer learning [11]. If the features of two datasets are the same and overlap, then horizontal federated learning can handle it better. Horizontal federated learning split the datasets horizontally and selects the portion of data with the same features but different users for training. In horizontal federated learning if the number of users increases the sample size will increases. Whereas vertical federated learning is used when different datasets with different features need be to trained and join the model to obtain a global model. Federated transfer learning is vertical federated learning enforced with an already trained model on a quite similar dataset, utilised for different problems. Federated averaging, FedAvg is the first Federated learning algorithm developed by Google. Other variants of FedAvg are FedProx, FedMa and FedOpt. Various applications of federated learning in different fields are mentioned in [12]. Multiparty database querying without exposing the data in

main domains such as finance and healthcare are some examples.

2.3 Blockchain

Blockchain is a shared, open distributed ledger with all transactions through it recorded and those records are immutable [13]. Each transaction through the blockchain is transparent to all users in that particular blockchain and these transactions cannot be modified. i.e, transactions are tamper-proof. Blockchain is a decentralized system so there is no central server to manage or control other nodes participating in the network instead it allows peer-to-peer communication in the network. Secure hash functions were used in 1991 by Haber and Stornetta for timestamping digital documents [14]. Blockchain technology uses secure hash functions for storing the transactions and each node can verify these transactions individually.

Hyperledger Fabric [15] is a permissioned distributed ledger framework created by IBM. It started as an open-source project to enhance the blockchain for enterprises. Some fundamental elements of Hyperledger are channels for nodes to communicate with each other, peers to manage ledgers and smart contacts, and ordering service nodes to receive transactions from other nodes. Membership service provider MSP is another component that deals with the membership operations of each member in the network. Private data collection [15] is also introduced in Hyperledger Fabric.

3 Related works

Machine learning and federated learning are two essential fields that have been undergoing wide research and coming up with new developments with better results. The categorization and applications of these learning algorithms are briefly mentioned in the background section. Utilising these machine learning algorithms, the estimated totem value mentioned in the TOTEM architecture [1] [16] can be improved, and also figure out the category of federated learning that need to be applied for the multi-provider scenario in the TOTEM are the two objectives of this article. For the former objective, one of the

relevant studies on the defined direction is mentioned in the article [17]. The dataset for the experiment CoRCoD: Code Runtime Complexity Dataset is extracted from available online coding platforms. Feature engineering and code embedding are the basic operations to obtain the outcome and thus can compare the performances. For static code analysis in the TOTEM for the user-submitted code, it is highly relevant. To obtain the representation of the source program by effectively retrieving the syntactic and semantic features of the corresponding code is performed in this work. For defect prediction with abstract syntax trees of code to produce the feature generation is presented in [18]. Supervised learning methods for runtime prediction of algorithms are proposed in [19], but it also agrees that the execution time is not a standard measure for analysing the efficiency of any algorithms. Using extracted hand-engineered features from control flow and data dependency graphs of codes for automatic grading of programs is described in [20]. These above mentioned works and similar studies show how feature extraction can be performed and predict the complexity of the codes for different purposes. In this work, we will introduce an extension part for the TOTEM architecture to predict the complexity of the submitted code by the user and thus calculate the estimated totem value required for a particular code to execute.

Federated learning and its applications are mentioned in the previous section. Federated learning requires a central aggregation system to combine and produce a global model, also need adversary control on the participants included in the learning process. Another issue encountered is with the model security. Federated learning ensures data privacy but not security on the model generated. Blockchain-enabled Federated learning can fix these problems since it does not require a third party for central aggregation and both data privacy and security of the model are ensured. Smart contracts and built-in blockchain security mechanisms resolve or avoid conflict between the participants. In a recent survey on the application and implementation of blockchain-enabled federated learning frameworks [21] it is apparently explained. Apart from this, the paper discussed the challenges and future directions for blockchain-enabled federated learning, especially for the Internet of Vehicles. Another article [22], which can be catego-

rized as a systematic literature review of blockchain-based federated learning. It examines the existing federated learning issues, and briefly discusses the essence of existing blockchain-based federated learning. It surveys architectures that are proposed to show the relevance of using blockchain-enabled federated learning to ensure model data security. An architecture that is designed for secure data sharing through blockchain is defined and further development for the data exchange scenario to a machine learning problem with integration of federated learning is defined in [23]. It also used federated learning in the consensus process of blockchain, so that the computation work can be utilised for federated learning purposes. These articles prove that the proposed data-sharing system ensures security, at the same time high efficiency and good accuracy. "FLchain" [24] is an architecture, that uses a blockchain network for enhancing the security of federated learning. It studies the application of federated learning on various kinds of data and shows the results obtained from the experiment showing the influence of this integration in various fields. One of several examples is [25], a privacy-preserved federated learning approach is used to predict traffic flow. Similarly, the relevance of federated learning in the TOTEM architecture is one of the main objectives of our present study. In the multi-provider scenario of the TOTEM architecture, the user requests to analyse two datasets from two different data owners. Utilising the advantage of using blockchain alone with federated learning can solve the problem in this scenario. The method to resolve it with an example will be carried out in the coming sections.

4 Role of Artificial Intelligence in TOTEM Architecture

TOTEM architecture can utilise machine learning technology for improving the pre-calculation of totem value required for a particular code to execute. The possibilities and requirements to enable this facility is explained in this section.

4.1 Prior work

The TOTEM is the architecture proposed in [1], it enables the data users to submit their own code to analyse the data in data owner's environment, without sharing the actual data. The TOTEM is a three-layered architecture; Blockchain consortium, Storage layer and Computational layer. Here both the computational layer and storage layer are part of the data owners' environment.

The data needed for execution will be present in the storage layer and when these data need to be executed as per the data user request, the execution will happen in the computational layer. The result of the execution will be passed to the data user. The request from the data user, the code for execution and the results all needed to be transferred in a secure and transparent manner. Moreover, the agreement between the users and data owners, authorization of users, request rejection and acceptance need to be controlled and managed efficiently and should be secured. This is the purpose of the blockchain consortium in the proposed architecture. Thus, all the transactions will be secure and tamper-proof. The user code submitted needs to be controlled in order to avoid any malicious functions in the user code such as infinite loops or malicious functions, a controlled computation is required thus an entity called totem, is introduced in the architecture. Additional to this, articles [16], [2] and [3] gives a detailed explanation of newly introduced components such as totem manager and updaters in the computational layer, the architecture of the TOTEM defined SDK, and extended multi- provider TOTEM architecture correspondingly. Figure 1 below shows the three-layered architecture of the TOTEM. Two actors in the TOTEM are data consumers and data owners. The Data consumer (data user) has access only to the blockchain consortium layer whereas the data provider (data owner) has access to the blockchain layer and owns the computational layer and storage layer. The machine learning part is introduced in the architecture to process the code written in TOTEM defined SDK and predicts the complexity of that particular code.

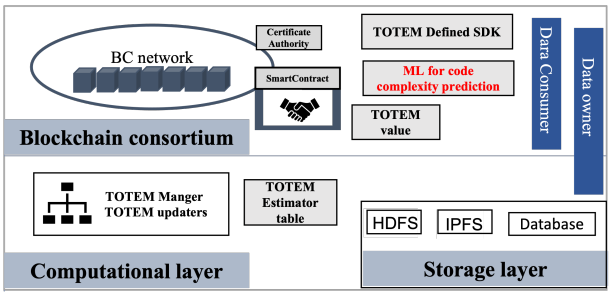


Figure 1: Layers of TOTEM with Actors

4.2 Machine learning enabled TOTEM

One of the initial and crucial parts of the TOTEM architecture is that the estimation of the totem value required to execute a particular code submitted by the data user. After the estimated value calculation, a preliminary check will be carried out. Each data user will have an account with totem values, it can be purchased as their requirement and demands. The purpose of a preliminary check is to make sure that the data user has enough value for the execution requested by them.

The estimated totem value was calculated by checking the operands and opcodes present in the code and a fixed value according to the totem estimator table, which is maintained by the data owners, will be substituted, and finally sum up the value. It was the initial method, explained for the TOTEM architecture. But this will suit only for simplified codes since we need to go through each line by line and if complexity increases the value may not be accurate as well. In this case, the estimated value can be too different from the actual needed value. Thus, there are chances to reject the request for execution if the calculated totem value is larger than the actual value and if there is not enough totem value in the data user account. Another chance is to estimate a lower value than the actual and once the actual execution happens and when it realizes that the totem value with data user exhausted, the whole execution will be stopped immediately and the final result will not be obtained. Hence it makes it clear that an accurate value will avoid these issues and more trust in the required totem value. The workflow of machine learning enabled

totem value calculation is given in Figure 2. Once the user code is written in a specific format the relevant features of the code need to be extracted in the specified format to apply the prediction algorithm model. This model will be updated after each new code is evaluated and obtain the complexity. The value of the totem can be calculated with the complexity obtained from the prediction.

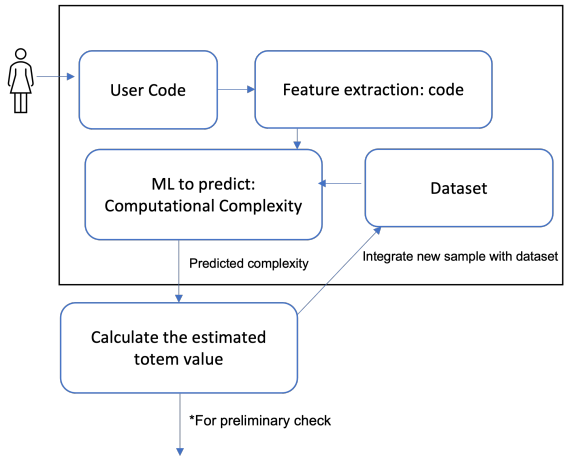


Figure 2: Workflow of Machine learning enabled TOTEM

With this new proposal for the calculation of totem value, another aspect is that in the original architecture each opcode and operand according to the type, complexity should be calculated with the value corresponding to each in the estimator table. The obtained values from that will be added together to obtain the final value. When the code level increases according to the requirement of the data user, it will again make things complicated. Hence the more convenient and better way is to estimate totem value by considering the previous sample codes and their complexity as well. Thus, one should maintain the dataset with specified fields obtained from the codes and label the complexity of the codes. With the sample dataset which may work initially with expected errors and after obtaining good enough data, it will predict more accurate results. Hence the model will get more and more accurate in course of time.

Dataset updation will be another vital part of the process. After each prediction, the corresponding fields from the code predicted value

and totem value calculated should be updated. Another essential action is that, if the totem value after the actual execution is different from the predicted value, we must update the existing one with the real value of the totem used.

4.2.1 Feature extraction

The process of identifying and transforming raw data into features that can be utilised for machine learning purposes is called feature engineering. Feature creation, transformation, feature extraction and feature selection are the main steps involved in feature engineering. Feature extraction is the conversion of the raw data into numerical features, in a way that it ensures the features of the original data will be preserved. In our present study we need to convert our code written in TOTEM defined SDK format to a dataset that needs to be used as an input for the prediction algorithm. A sample TOTEM defined SDK output and conversion to data required for prediction is shown in Figure 3.

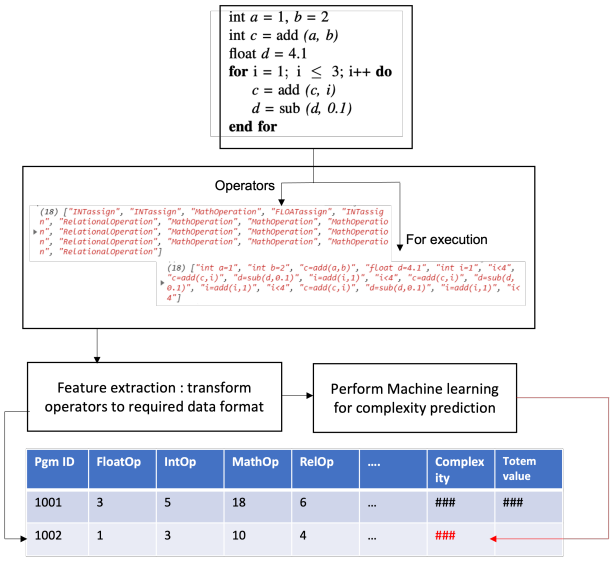


Figure 3: Sample format and dataset updation

The fields given are in sample format. Here we have given the number of float, int type declarations, mathematical operations,

relational operations, etc. The code given consists of the initialisation of variables and a simple For loop. But when it comes to the code for analysis of certain datasets, the size of data needed to be executed is also a major factor, and hence we have to include the dataset size.

4.2.2 Dataset

Dataset for training the model is not available for the present study. Hence, we use the TOTEM defined SDK format for coding. In the beginning, the calculation of the totem estimate will be carried out with the help of an estimator table as explained in the TOTEM architecture; where the allowed keywords and corresponding value of totem required for each keyword to be executed is given. The data collection will start from the code submission stage and the fields will be updated through feature extraction and then the totem value obtained from the calculation will be added to another field totem. It will be updated later after the final execution happens if that is different from the calculated totem estimate before. If a halt of execution due to lack of totem value in real execution occurred, again the value should be changed to the unknown for further studies and need to be updated.

4.2.3 ML Algorithm for totem value prediction

Classification of the computational complexity of the program is the requirement. Supervised classification algorithms with high accuracy will be used as the model for prediction. Totem value can also be predicted in this other than complexity thus supervised regression algorithms are preferred.

4.3 Federated learning with TOTEM

Federated learning can fix some of the challenges that we face in the multi-provider TOTEM architecture. For example, dataset need to be trained to obtain a model. If the dataset is from different resources or in the TOTEM scenario from different data providers, where data sharing is not possible due to data privacy and security issues, the global combined model by training each dataset can be

obtained using the federated learning technique. Federated learning can be applied to the TOTEM architecture in two ways. 1. Cross-silo federated learning where multiple data providers are involved in the process. 2. Cross-device federated learning where data users and data provider(s) are involved.

4.3.1 Cross-Silo Federated learning.

In cross-silo federated learning, the participants are basically organisations or companies. It is normal that each candidate in cross-silo federated learning is responsible to participate in the entire learning process. The number of participants can vary from two to many, but will be less compared to participants in cross-device. In the TOTEM architecture, we have data providers, from the same or different fields. When the providers come with a common interest in having a global model with training the datasets from different organizations (data providers), with a common connection in their dataset they own, cross-silo federated learning can be applied. Instead of a central server in federated learning for aggregating the model updates, we can utilise blockchain technology for a decentralized aggregation process as well. One of the applications is to continuously monitor and train the data collected from wind farms located in different locations. It is sensible to have a model which is continuously updated with the data collected, for a better power production system. For that, it is important to observe, learn and understand the behavior of some of the vital parameters such as wind speed, wind direction, weather, etc. to change and adjust the direction, pitching and yaw of the wind turbines. According to the TOTEM architecture, the wind farm data collected is only available in different data provider's environments. These data need to be trained and models need to be updated continuously to aggregate the models obtained from all the providers. Here the beneficiaries from the global model are all the providers. But the aggregation of models needs to be carried out through a collective decision from all data providers participating in the learning process. The totem value corresponding to computational cost demands for the aggregation process computation will be deducted commonly from the participants in the cross-silo federated learning. Figure 4

shows the representation of the sample system mentioned above.

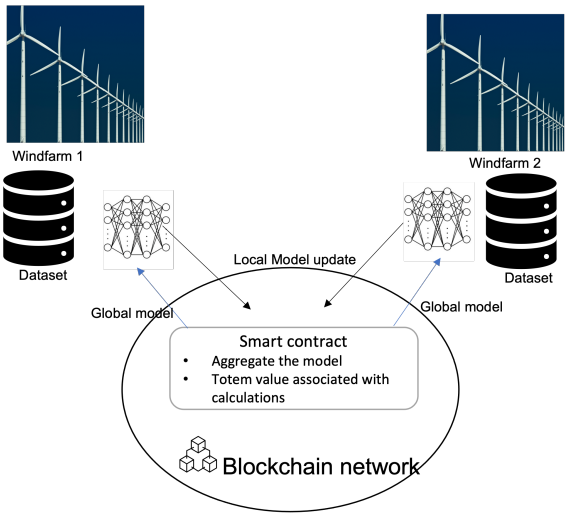


Figure 4: Example for cross-silo federated learning

- The participants, both the wind farm data providers in Figure 4 need to initiate the learning process through an agreement.
- From each data provider the local training will be carried out and upload the model obtained.
- The models obtained from data providers will be aggregated.
- The new global model obtained will be then given to the data providers.

Here we discussed about how different data providers can aggregate their trained model using federated learning. Another option for the data users in the TOTEM architecture is that data users can submit the initial model for the training process, and data providers can train their own data in the model and aggregate the model for obtaining the global model. Here the data users have to spend the cost for the training and aggregation of the model as it is the requirement for the data user. The aggregation can be done by one of the data providers depending on the agreement that they have.

4.3.2 Cross-Device Federated learning

Cross-device federated learning applies to devices with a small amount of data for training. It also demands a large number of participants for a successful training process. In the TOTEM perspective, the data users can participate in training and model updation for the data provider to gain rewards as totem value. This gives opportunities to data users to collect the totem value for further use later. However, for this detailed study is required and critically examine the challenges in terms of the TOTEM perspective.

5 Conclusion

TOTEM is a framework that integrates blockchain and big data systems to ensure secure and privacy driven data analytics by utilising the complementary properties of both technologies. It allows data users to analysis their own code on data provider's data, without moving the data across the network. Additional entities and components for monitoring malicious functions are also integrated into the system. The framework is patented under US Patent No.: US11,121,874 B2 [26]. The two main directions where artificial intelligence can be applied in the TOTEM architecture to improve the features are discussed in this article. One direction is to implement a machine learning algorithm, to train the data collected from previous estimations to estimate the totem values required for particular code execution. The accurate estimation of totem value will help the users to execute their codes flawlessly and increase trust in the architecture. However, there are limitations in terms of available data for the initial prediction of the totem value. The article proposes the conventional estimation of the totem value for initial times and machine learning can be implemented as the system matures and have enough data to predict the totem values. Another direction is to apply federated learning for a global model, obtained by training dataset from different data providers. However further detailed examination and implementation challenges need to be sorted out in the future work.

References

[1] Dhanya Therese Jose, Antorweep Chakravorty, and Chunming Rong. "TOTEM : Token for controlled computation: Integrating Blockchain with Big Data." In: (2019), pp. 1–7. DOI: 10.1109/ICCCNT45670.2019.8944855.

[2] Behfar Behzad, Dhanya Therese Jose, Antorweep Chakravorty, and Chunming Rong. "TOTEM SDK: an open toolset for token controlled computation managed by blockchain." In: (2021), pp. 1–8. DOI: 10.1109/CSDE53843.2021.9718489.

[3] Dhanya Therese Jose, Jørgen Holme, Antorweep Chakravorty, and Chunming Rong. "Integrating big data and blockchain to manage energy smart grids—TOTEM framework." In: *Blockchain: Research and Applications* 3.3 (2022), p. 100081. ISSN: 2096-7209. DOI: https://doi.org/10.1016/j.bcra.2022.100081. URL: https://www.sciencedirect.com/science/article/pii/S2096720922000227.

[4] Issam El Naqa and Martin J Murphy. "What is machine learning?" In: (2015), pp. 3–11.

[5] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*. Vol. 4. 4. Springer, 2006.

[6] Rich Caruana and Alexandru Niculescu-Mizil. "An empirical comparison of supervised learning algorithms." In: (2006), pp. 161–168.

[7] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." In: *nature* 521.7553 (2015), pp. 436–444.

[8] IBM Cloud Education. "Machine learning." In: *URL: https://www.ibm. com/cloud/learn/machine-learning* (2020).

[9] Jakub Konečn, H Brendan McMahan, Daniel Ramage, and Peter Richtárik. "Federated optimization: Distributed machine learning for on-device intelligence." In: *arXiv:1610.02527* (2016).

[10] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H Yang, Farhad Farokhi, Shi Jin, Tony QS Quek, and H Vincent Poor. "Federated learning with differential privacy: Algorithms and performance analysis." In: *IEEE Transactions on Information Forensics and Security* 15 (2020), pp. 3454–3469.

[11] Qiang Yang, Yang Liu, Yong Cheng, Yan Kang, Tianjian Chen, and Han Yu. "Federated learning." In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 13.3 (2019), pp. 1–207.

[12] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. "Federated machine learning: Concept and applications." In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 10.2 (2019), pp. 1–19.

[13] Karim R Lakhani and M Iansiti. "The truth about blockchain." In: *Harvard Business Review* 95.1 (2017), pp. 119–127.

[14] S Haber and WS Stornetta. "How to Time-Stamp a Digital Document, Menezes AJ, Vanstone SA (eds) Advances in Cryptology-CRYPTO'90. CRYPTO 1990." In: *Lecture Notes in Computer Science* 537 (1991).

[15] "Hyperledger Fabric Documentation Release 1.4." In: *Hyperledger* (2019).

[16] Dhanya Therese Jose, Antorweep Chakravorty, and Chunming Rong. "Distributed computational framework in TOTEM architecture enabled by blockchain." In: (2020), pp. 83–88. DOI: 10.1109/ICCSE49874.2020.9201683.

[17] Jagriti Sikka, Kushal Satya, Yaman Kumar, Shagun Uppal, Rajiv Ratn Shah, and Roger Zimmermann. "Learning Based Methods for Code Runtime Complexity Prediction." In: *European Conference on Information Retrieval*. Springer. 2020, pp. 313–325.

[18] Jian Li, Pinjia He, Jieming Zhu, and Michael R Lyu. "Software defect prediction via convolutional neural network." In: *2017 IEEE international conference on software quality, reliability and security (QRS)*. IEEE. 2017, pp. 318–328.

[19] Frank Hutter, Lin Xu, Holger H Hoos, and Kevin Leyton-Brown. "Algorithm runtime prediction: Methods & evaluation." In: *Artificial Intelligence* 206 (2014), pp. 79–111.

[20] Shashank Srikant and Varun Aggarwal. "A system to grade computer programming skills using machine learning." In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2014, pp. 1887–1896.

[21] Mustain Billah, Sk Mehedi, Adnan Anwar, Ziaur Rahman, Rafiqul Islam, et al. "A Systematic Literature Review on Blockchain Enabled Federated Learning Framework for Internet of Vehicles." In: *arXiv preprint arXiv:2203.05192* (2022).

[22] Dongkun Hou, Jie Zhang, Ka Lok Man, Jieming Ma, and Zitian Peng. "A systematic literature review of blockchain-based federated learning: Architectures, applications and issues." In: *2021 2nd Information Communication Technologies Conference (ICTC)*. IEEE. 2021, pp. 302–307.

[23] Yunlong Lu, Xiaohong Huang, Yueyue Dai, Sabita Maharjan, and Yan Zhang. "Blockchain and federated learning for privacy-preserved data sharing in industrial IoT." In: *IEEE Transactions on Industrial Informatics* 16.6 (2019), pp. 4177–4186.

[24] Umer Majeed and Choong Seon Hong. "FLchain: Federated learning via MEC-enabled blockchain network." In: *2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS)*. IEEE. 2019, pp. 1–4.

[25] Yi Liu, JQ James, Jiawen Kang, Dusit Niyato, and Shuyu Zhang. "Privacy-preserving traffic flow prediction: A federated learning approach." In: *IEEE Internet of Things Journal* 7.8 (2020), pp. 7751–7763.

[26] Dhanya Therese Jose, Antorweep Chakravorty, and Chunming Rong. *Method for analyzing data using a blockchain, a data provider and a data customer therefor*. US Patent 11,121,874. Sept. 2021.

Appendix

US011121874B2

(12) **United States Patent**
Jose et al.

(10) **Patent No.: US 11,121,874 B2**
(45) **Date of Patent: Sep. 14, 2021**

(54) **METHOD FOR ANALYZING DATA USING A BLOCKCHAIN, A DATA PROVIDER AND A DATA CUSTOMER THEREFOR**

(71) Applicant: **The University of Stavanger**, Stavanger (NO)

(72) Inventors: **Dhanya Therese Jose**, Stavanger (NO); **Antorweep Chakravorty**, Stavanger (NO); **Chunming Rong**, Stavanger (NO)

(73) Assignee: **The University of Stavanger**, Stavanger (NO)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 159 days.

(21) Appl. No.: **16/660,500**

(22) Filed: **Oct. 22, 2019**

(65) **Prior Publication Data**

US 2021/0119796 A1 Apr. 22, 2021

(51) **Int. Cl.**
G06F 7/04 (2006.01)
H04L 9/32 (2006.01)
(Continued)

(52) **U.S. Cl.**
CPC ***H04L 9/3213*** (2013.01); ***G06F 9/3891*** (2013.01); ***G06F 16/907*** (2019.01); ***H04L 9/0643*** (2013.01); *H04L 2209/38* (2013.01)

(58) **Field of Classification Search**
CPC . H04L 63/061; H04L 9/3213; H04L 12/1453; H04L 12/1417; H04L 9/32313;
(Continued)

(56) **References Cited**

U.S. PATENT DOCUMENTS

2018/0097779 A1* 4/2018 Karame G06Q 20/065
2019/0179939 A1* 6/2019 Govindarajan G06F 16/2445
(Continued)

OTHER PUBLICATIONS

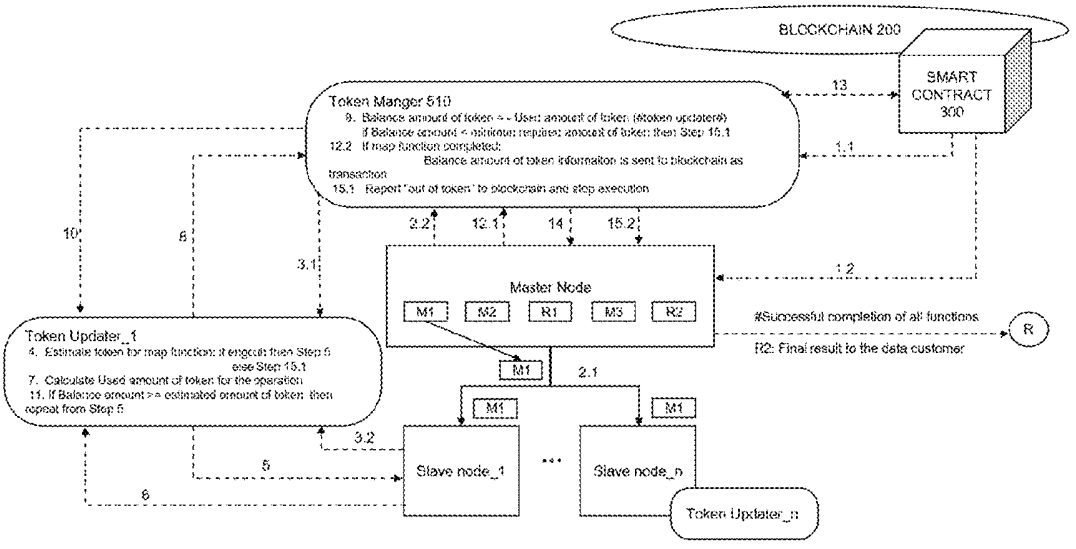
Bruno Skvorc, May 24, 2018, Ethereum: How Transaction Costs are Calculated, pp. 1-7. (Year: 2018).*
(Continued)

Primary Examiner — Luu T Pham
Assistant Examiner — Jenise E Jackson
(74) *Attorney, Agent, or Firm* — Adsero IP

(57) **ABSTRACT**

A method for analyzing data using a blockchain, a data provider and a data customer therefor are disclosed. The method analyzing data using a blockchain is provided wherein a plurality of data sets is stored and processed in a data storage in a distributed manner using a cluster of nodes. The method comprises steps of deploying a smart contract to the blockchain according to a request from a data customer, receiving a request for executing code for data sets selected by a data customer, estimating an amount of token required for executing the code for the selected data sets in the data storage, and controlling, in said distributed manner using the cluster of nodes, execution of the code for the selected data sets based on the balance amount of token while the balance amount of token is greater than the estimated amount of token. The request for executing code includes code to be executed and a balance amount of token which the data customer currently has. The code to be executed includes a set of computational operations. The balance amount of token is updated after execution of each computational operation in said distributed manner. The amount of token represents number of units for an entity which controls computational complexity of the code requested by the data customer.

14 Claims, 8 Drawing Sheets



(51) **Int. Cl.**
G06F 16/907 (2019.01)
G06F 9/38 (2018.01)
H04L 9/06 (2006.01)

(58) **Field of Classification Search**
CPC . H04L 9/0643; H04L 2209/38; G06F 9/3891; G06F 21/62; G06F 21/6245; H04W 12/06
USPC 726/9
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

2019/0319808	A1*	10/2019	Fallah	H04L 9/30
2019/0340607	A1*	11/2019	Lynn	G06Q 20/389
2020/0027085	A1*	1/2020	Lee	G06Q 20/401
2020/0184448	A1*	6/2020	Jain	G06Q 20/40

OTHER PUBLICATIONS

Zuchowski, Nov. 14, 2017, Ethereum: Everything you want to know about Gas, pp. 1-8. (Year: 2017).*
Kasireddy, Sep. 13, 2017, How does Ethereum Work, Anyway?, pp. 1-27. (Year: 2017).*
Christidis et al, Blockchains and Smart Contracts for Internet of Things, IEEE, May 10, 2016, pp. 2292-2303. (Year: 2016).*
Pan et al, EdgeChain: An Edge-IoT Framework and Prototype Based on Blockchain and Smart Contracts, IEEE, Oct. 26, 2018, pp. 4719-4732. (Year: 2018).*

* cited by examiner

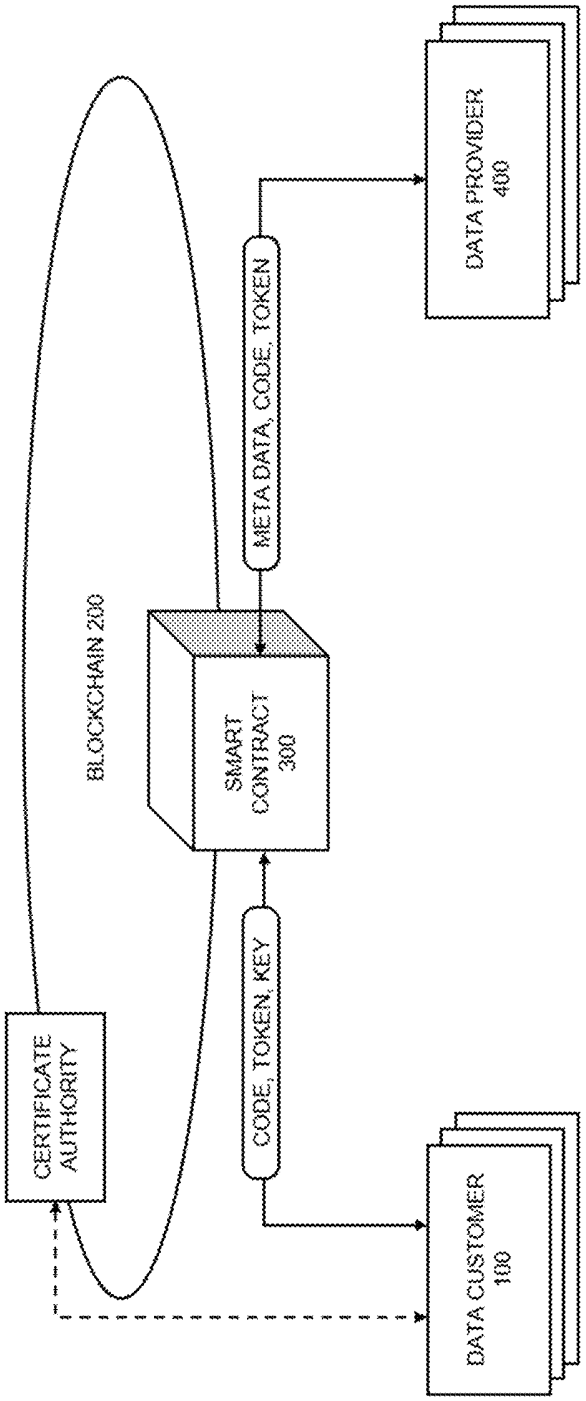

DATA PROVIDER
400
META DATA, CODE, TOKEN
BLOCKCHAIN 200
SMART
CONTRACT
300
CERTIFICATE
AUTHORITY
CODE, TOKEN, KEY
DATA CUSTOMER
100

Fig. 1

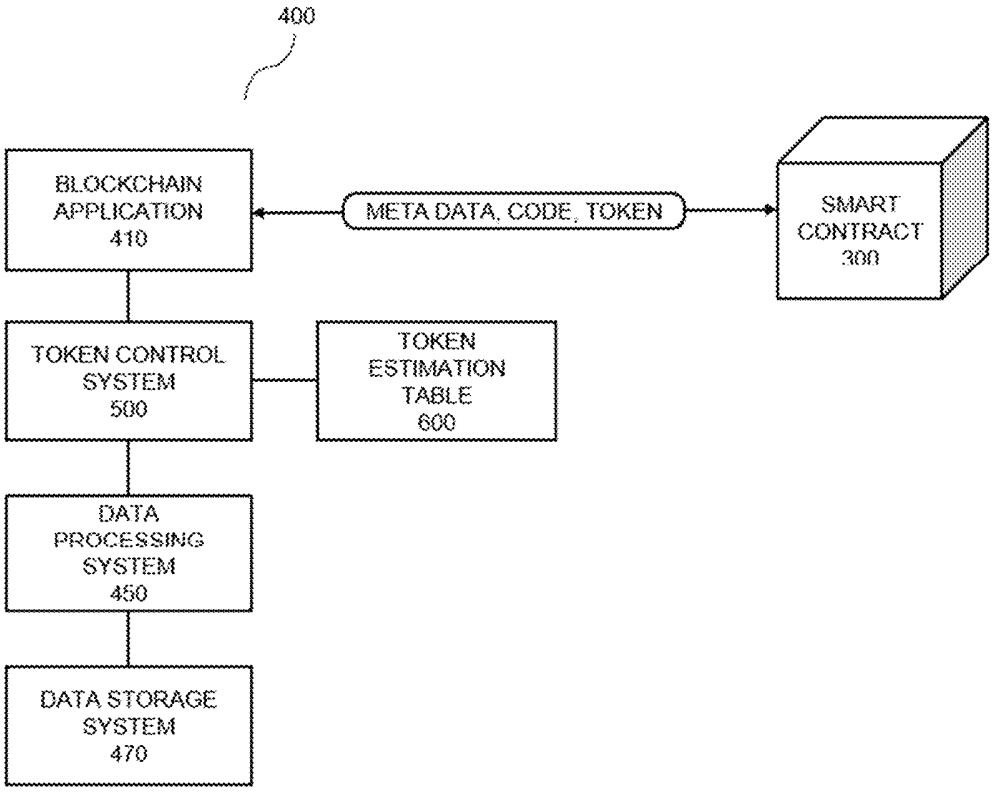

400
BLOCKCHAIN APPLICATION 410
META DATA, CODE, TOKEN
SMART CONTRACT 300
TOKEN CONTROL SYSTEM 500
TOKEN ESTIMATION TABLE 600
DATA PROCESSING SYSTEM 450
DATA STORAGE SYSTEM 470

Fig. 2

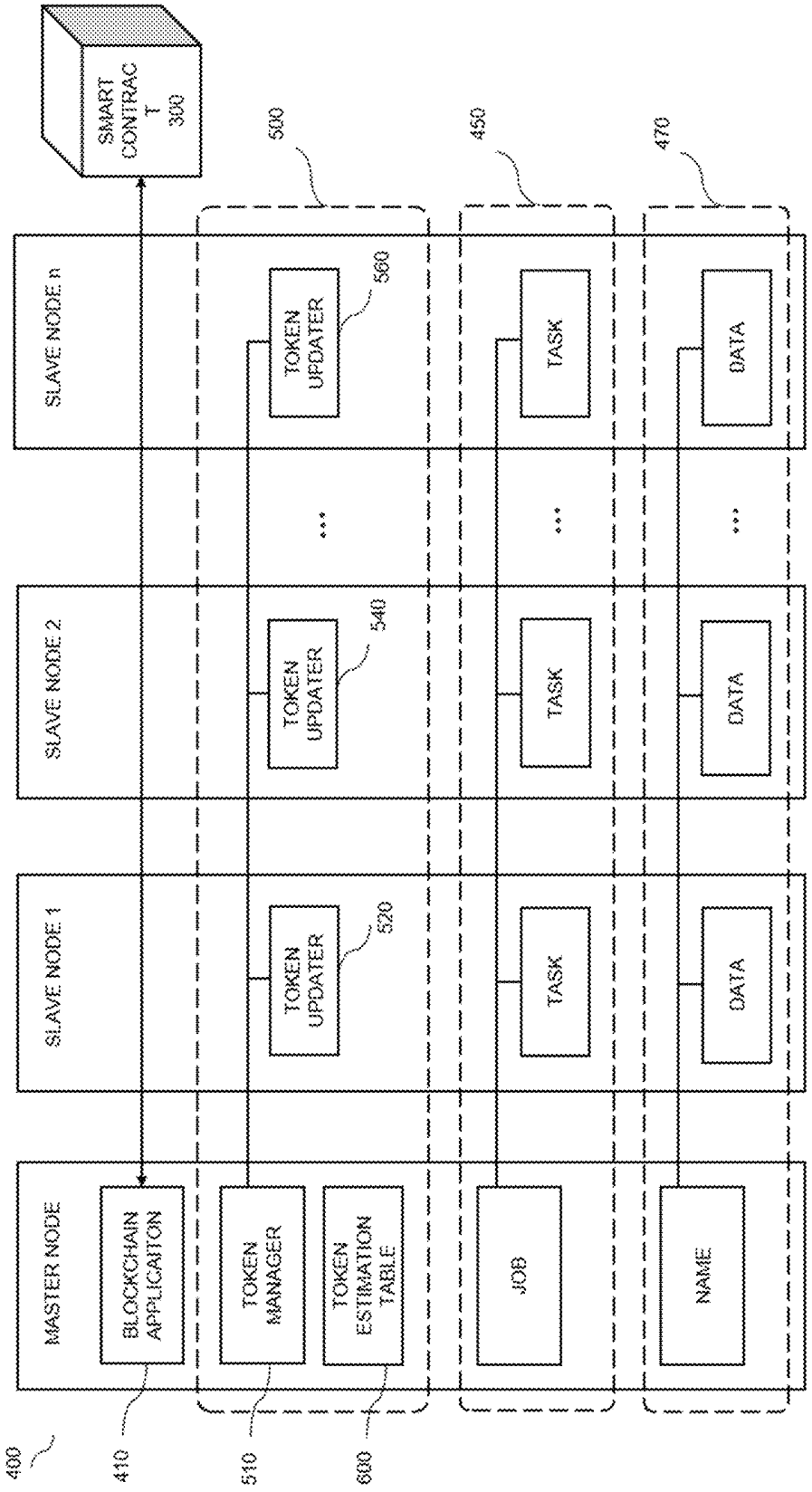

SMART CONTRACT 300
500
450
470
SLAVE NODE n
SLAVE NODE 2
SLAVE NODE 1
MASTER NODE
TOKEN UPDATER
560
TOKEN UPDATER
540
TOKEN UPDATER
520
TASK
TASK
TASK
DATA
DATA
DATA
BLOCKCHAIN APPLICATION
TOKEN MANAGER
TOKEN ESTIMATION TABLE
JOB
NAME
400
410
510
600

Fig. 3

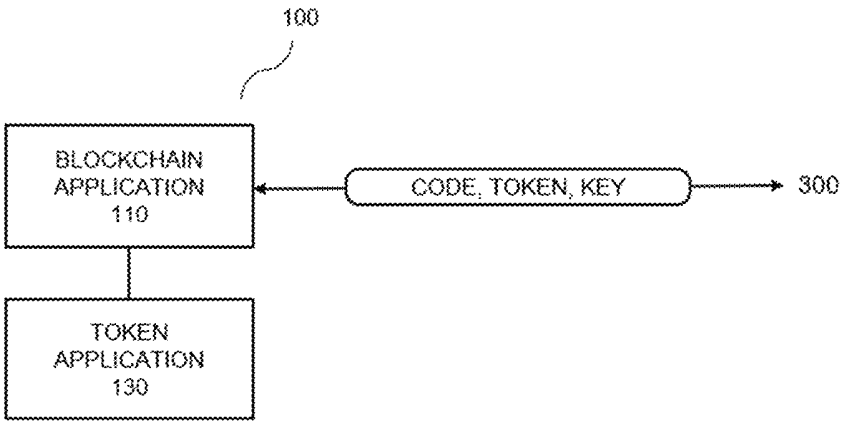

100
BLOCKCHAIN APPLICATION 110
CODE, TOKEN, KEY
300
TOKEN APPLICATION 130

Fig. 4

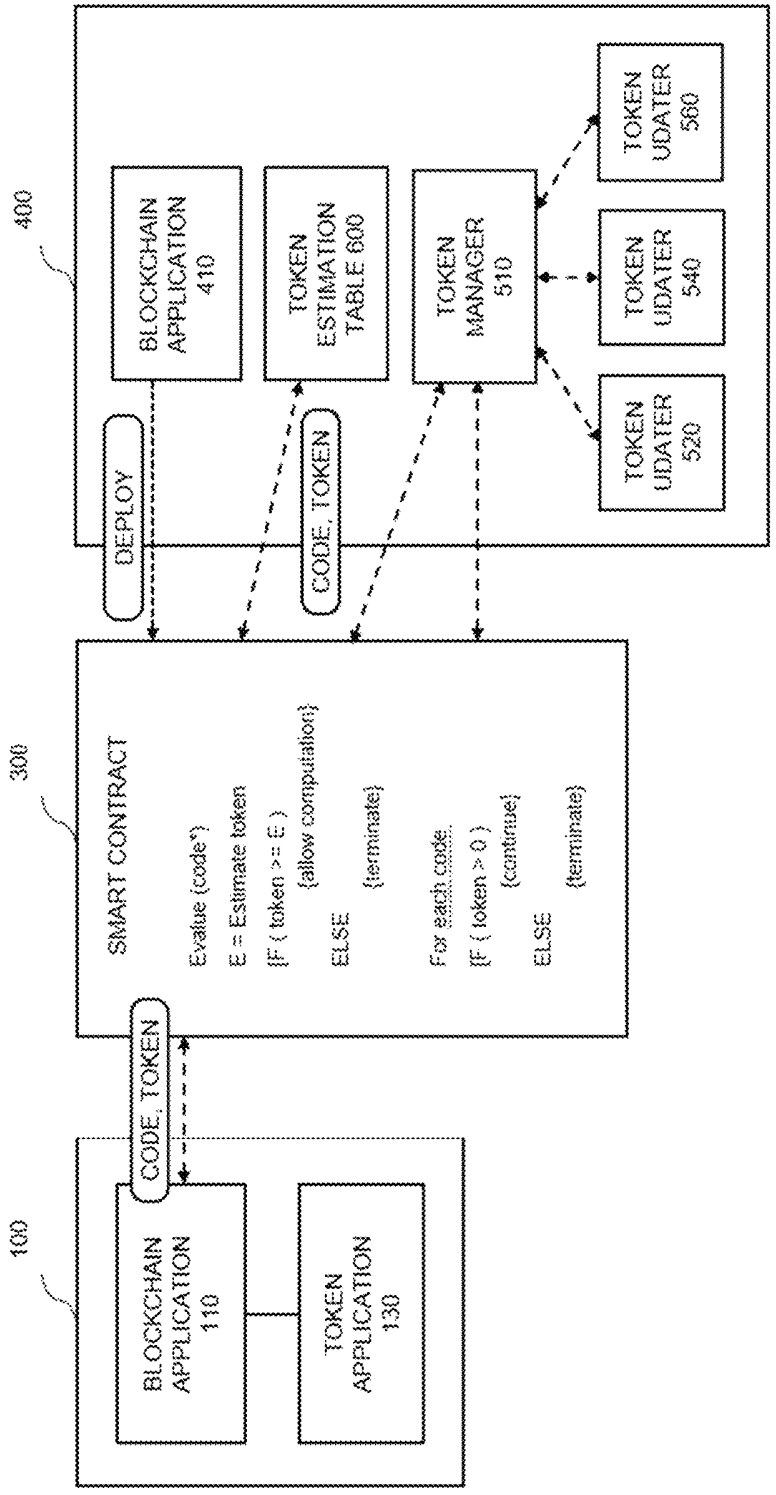

100
BLOCKCHAIN APPLICATION 110
TOKEN APPLICATION 130
CODE, TOKEN
300
SMART CONTRACT
Evalue (code*)
E = Estimate token
IF (token >= E)
{allow computation}
ELSE
{terminate}
For each code
IF (token > 0)
{continue}
ELSE
{terminate}
DEPLOY
CODE, TOKEN
400
BLOCKCHAIN APPLICATION 410
TOKEN ESTIMATION TABLE 600
TOKEN MANAGER 510
TOKEN UDATER 520
TOKEN UDATER 540
TOKEN UDATER 560

Fig. 5

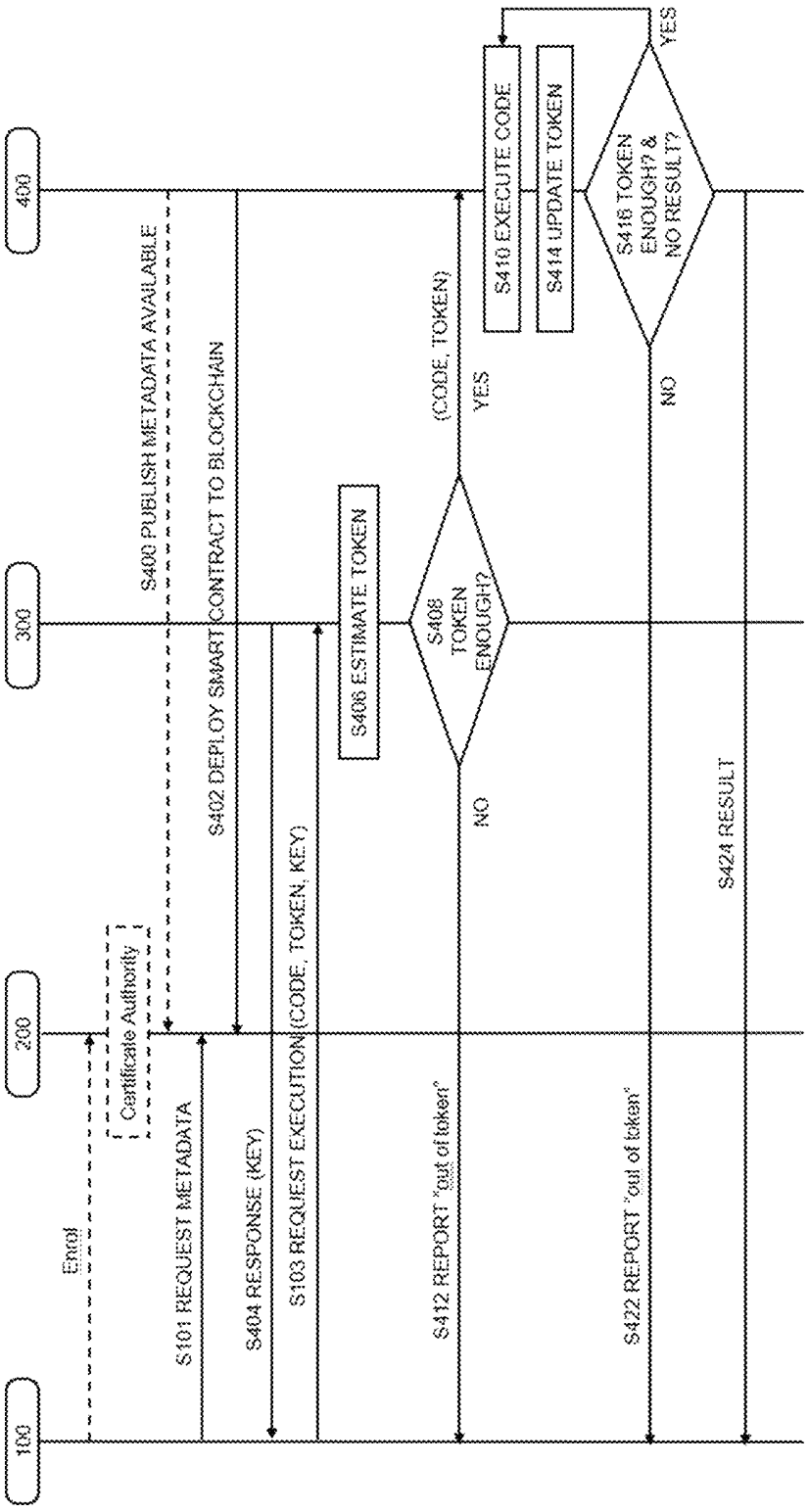

100
200
300
400
Email
Certificate Authority
S101 REQUEST METADATA
S400 PUBLISH METADATA AVAILABLE
S402 DEPLOY SMART CONTRACT TO BLOCKCHAIN
S404 RESPONSE (KEY)
S103 REQUEST EXECUTION (CODE, TOKEN, KEY)
S406 ESTIMATE TOKEN
S408 TOKEN ENOUGH?
S412 REPORT "out of token"
NO
YES
(CODE, TOKEN)
S410 EXECUTE CODE
S414 UPDATE TOKEN
S416 TOKEN ENOUGH? & NO RESULT?
YES
NO
S422 REPORT "out of token"
S424 RESULT

Fig. 6

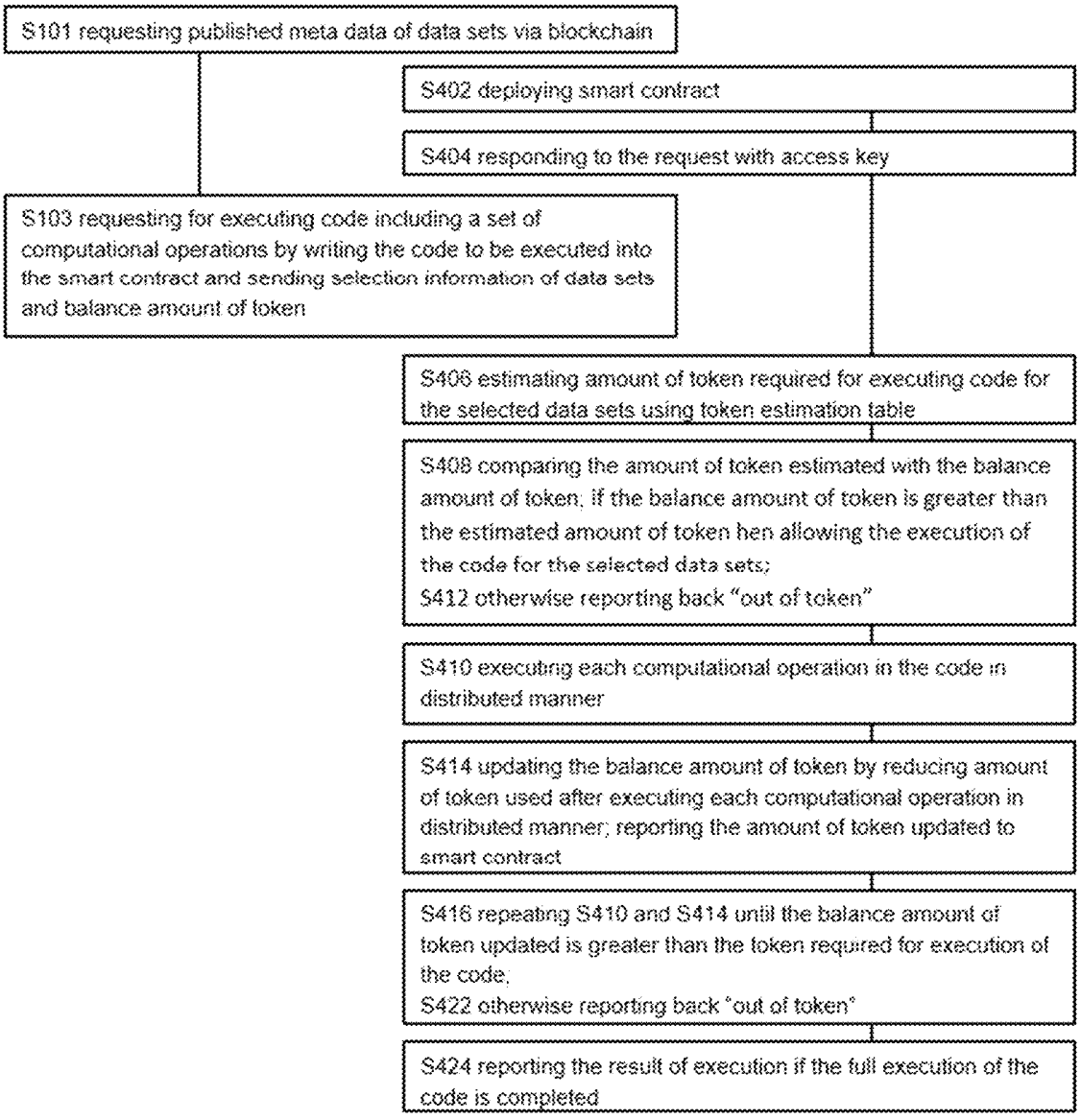

S101 requesting published meta data of data sets via blockchain
S402 deploying smart contract
S404 responding to the request with access key
S103 requesting for executing code including a set of computational operations by writing the code to be executed into the smart contract and sending selection information of data sets and balance amount of token
S406 estimating amount of token required for executing code for the selected data sets using token estimation table
S408 comparing the amount of token estimated with the balance amount of token; if the balance amount of token is greater than the estimated amount of token hen allowing the execution of the code for the selected data sets;
S412 otherwise reporting back "out of token"
S410 executing each computational operation in the code in distributed manner
S414 updating the balance amount of token by reducing amount of token used after executing each computational operation in distributed manner; reporting the amount of token updated to smart contract
S416 repeating S410 and S414 until the balance amount of token updated is greater than the token required for execution of the code;
S422 otherwise reporting back "out of token"
S424 reporting the result of execution if the full execution of the code is completed

Fig. 7

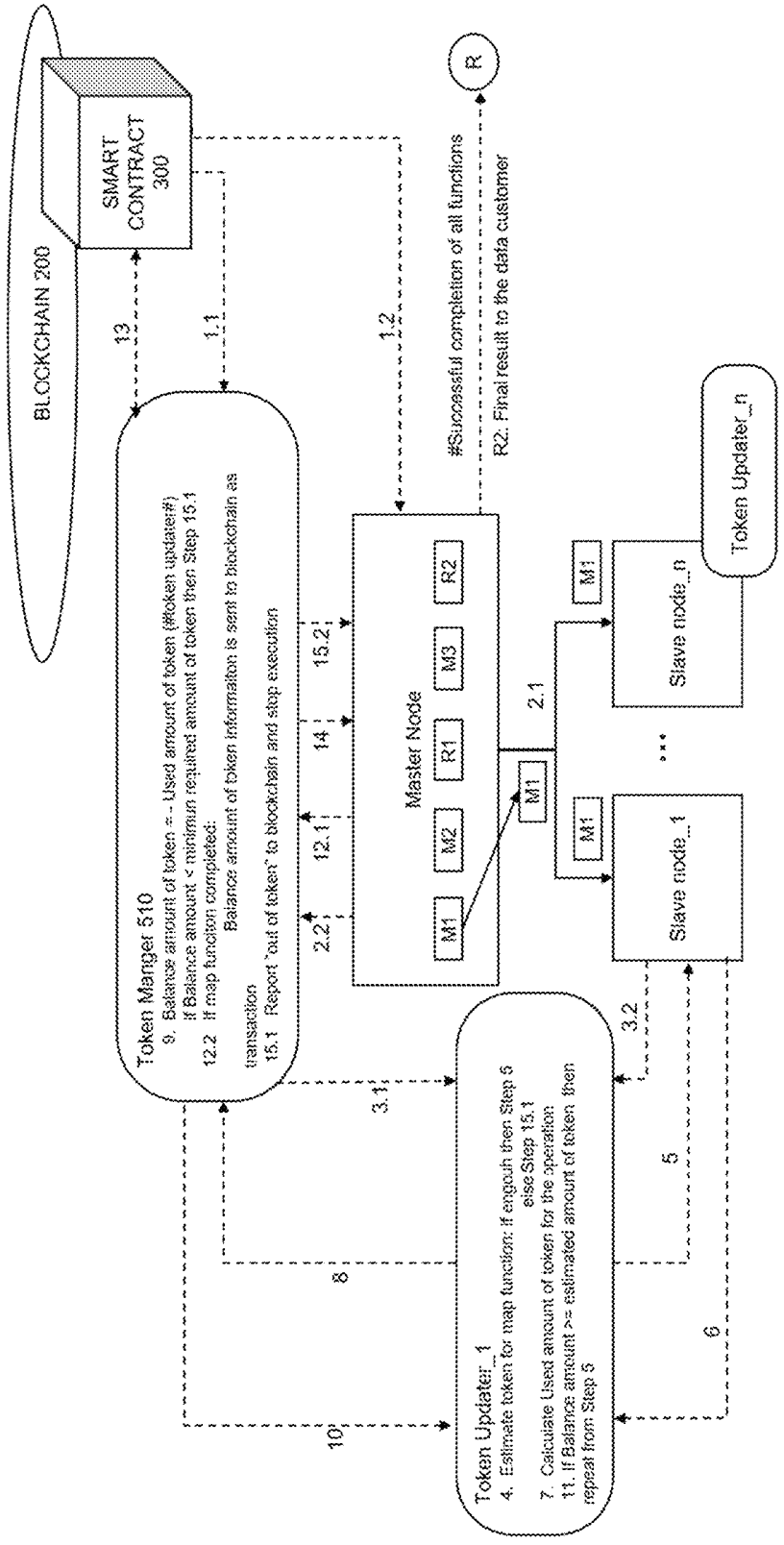

BLOCKCHAIN 200
SMART CONTRACT 300
13
1.1
1.2
R
#Successful completion of all functions
R2: Final result to the data customer
Token Manger 510
9. Balance amount of token =- Used amount of token (#token updater#)
If Balance amount < minimum required amount of token then Step 15.1
12.2 If map function completed:
Balance amount of token information is sent to blockchain as transaction
15.1 Report "out of token" to blockchain and stop execution
2.2
12.1
14
15.2
Master Node
M1
M2
R1
M3
R2
2.1
M1
M1
M1
Slave node_1
Slave node_n
Token Updater_n
3.1
3.2
5
6
8
10
Token Updater_1
4. Estimate token for map function: If enough then Step 6
else Step 15.1
7. Calculate Used amount of token for the operation
11. If Balance amount >= estimated amount of token, then repeat from Step 5

Fig. 8

METHOD FOR ANALYZING DATA USING A BLOCKCHAIN, A DATA PROVIDER AND A DATA CUSTOMER THEREFOR

TECHNICAL FIELD

The present invention relates generally to the field of data analyzing in a data storage. The present invention relates specially to the field of a method for analyzing data using a blockchain, a data provider and a data customer therefor according to the preamble of the independent claim.

BACKGROUND

The world increasingly becomes driven by data. Needs to deal with a large scale of data and diverse data streams keep increasing exponentially. Accordingly, it gets a main issue how a large scale of data and diverse data streams can be analyzed in a secure manner and how the privacy for the data can be protected.

Conventional methods for analyzing data demand that the data to be analyzed should be moved across networks for analysis thereof. However, transferring such a large scale of data is highly demanding on networks. Moreover, many of data providers are reluctant to furnish their own data via networks for the reason of privacy and security issues. Therefore, there is a need for a new method for analyzing a large scale of data without transferring the data to be analyzed to an external network in order to guarantee the security and the privacy of the data.

Various blockchain technologies are introduced in order to guarantee the security of data. Blockchains can be defined as a chronological database of transactions grouped in a block and validated by a network of nodes, with multiple blocks added one after another in a chain. Blockchains can be also interpreted as an open distributed ledger, which records transactions efficiently. The transactions recorded in blockchain are tamperproof and transparent to all users in the network. It is a peer-to-peer communication between nodes, therefore no central node controls entire network. Each node individually verifies all transactions directly. Cryptographically secured hash functions are implemented to store transactions in the blockchain.

Blockchain can be classified into two types, namely permissioned and permissionless. The main differentiator is based on requirements for authorizing nodes in a network. Permissionless blockchains are public and allow anonymous users to participate and contribute their computational power. There are no restrictions in joining the network for the verification process. Bitcoin and Ethereum are examples of the public or permissionless blockchain. In permissioned or private blockchain, permission to join the network is restricted to users within an organization or group of organizations. Only the selected nodes by the blockchain consortium can participate in the verification process. Hyperledger Fabric and Ripple are examples for permissioned or private blockchain.

As blockchains are decentralized, when a transaction is proposed, its validity can be verified by any node in the network. These nodes add the transactions to a block and append it to the existing chain. However, there is a chance that more than one node can come up with a new block to append to the blockchain at the same time. In order to prevent this situation, an agreement should be made between the nodes about the node that is chosen to append a new block. This agreement is called the consensus algorithm which can be either proof based or voting based.

One of the biggest advantages of blockchain is smart contract. Blockchain based smart contracts can be any kind of program that will be executed in a distributed manner without any centralized third-party node. Each transaction in a blockchain will occur only if the conditions in the smart contracts are met. In public blockchain, as it is anonymous, there is a chance that anyone can deploy smart contracts which requires high computation. Particularly in Proof of Work consensus algorithm, as all users in the network participate in validation, if the smart contract requires high execution time or it contains an infinite loop, large delays to the network is inevitable. This kind of denial-of service attack, which is so called DOS attack, could incur a catastrophic situation to the whole network. Therefore, it is important to limit the complexity of computations in smart contracts.

Meanwhile, big data represents data sets that are very large to be efficiently interpreted, collected, managed and processed using traditional mechanisms. The growth of big data can be described by the primary characteristics of volume, velocity, variety and veracity. In term of volume the size of data has been grown exponentially by units of MB, GB, TB and PB. In term of velocity, the speed of data processing has been evolved from batch processing to periodic processing, and further to real time processing. In term of variety of data, the type of data has been developed variously from table, data base, photo, web, audio, social, video, mobile, to even unstructured data. Big data veracity refers to the quality of data such as biases, noises and abnormality in data. Big data analytics stands for the method of strategy of analyzing large volume of data. Some of the popular frameworks for big data analytics are Hadoop, Spark, MongoDB, Strom, Cassandra, Neo4j, etc. Each framework has its own advantages and disadvantages.

Thus, there is a need for a solution allowing an safer way for security and privacy issues and an more efficient way for handling large data and parallel computations.

As a prior art, US 2019/0050854 A1 discloses an example blockchain-based digital data exchanges including data publisher endpoint devices and data subscriber endpoint systems. In response to a request from a data subscriber endpoint system, initiate a transaction to provide a data subscriber endpoint system with access to the data by the blockchain network via the data mart publisher client. The prior art achieves some improvement of data protection using a blockchain, however, it still has some problem that the data itself moves from the storage to the data subscriber system via external network.

SUMMARY

It is an object of the proposed technology to meet the above described needs. It is also an object to provide an improved method for analyzing data using a blockchain, a data provider and a data customer therefor with improved protection of the security and the privacy of data according to the preamble of the independent claim.

In a first aspect of the proposed technology, the objects are achieved by a method for analyzing data using a blockchain wherein a plurality of data sets is stored and processed in a data storage in a distributed manner using a cluster of nodes. The method comprises steps of i) data provider deploying a smart contract to the blockchain, ii) receiving a request for executing code for data sets selected by a data customer, iii) estimating an amount of token required for executing the code for the selected data sets in the data storage, and iv) controlling, in said distributed manner using the cluster of

nodes, execution of the code for the selected data sets based on the balance amount of token while the balance amount of token is greater than the estimated amount of token. The request for executing code includes code to be executed and a balance amount of token which the data customer currently has. The code to be executed includes a set of computational operations. The balance amount of token is updated after execution of each computational operation in said distributed manner. The amount of token represents number of units for an entity which controls computational complexity of the code requested by the data customer.

The step of controlling further comprises steps of executing each computational operation in the code for the selected data sets in said distributed manner, updating the balance amount of token by reducing an amount of token used after executing each computational operation in said distributed manner, and repeating the steps of executing and updating as long as the balance amount of token is enough for further execution of each computational operation based on the estimated amount of token and until getting a result of full execution of the code requested by the data customer. The cluster of nodes includes a master node and one or more slave nodes. The steps of execution and updating are performed in the one or more slave nodes in said distributed manner and the step of repeating is controlled in the master node in said distributed manner.

Before the step of deploying, the method further comprises publishing metadata of available data sets among the plurality of data sets stored in the data storage to the blockchain wherein the metadata includes information of identifying each data set in the data storage system without moving any data set from the data storage system to the data customer.

The step of estimating estimates an amount of token required for execution based on data type of each data set and code type of each computational operation to be executed. The step of estimating (S**406**) estimates the amount of token required for execution by looking up a token estimation table. The step of estimating includes comparing balance amount of token which the data customer currently has with the amount of token estimated for execution and controlling to allow or terminate execution of the code for the selected data sets based on result of comparison.

The step of controlling includes monitoring whether the balance amount of token which the data customer currently has is enough for executing each computational operation and controlling whether to continue execution of each computational operation or not. The step of controlling further comprises managing a total balance amount of token and reporting the total balance amount to the deployed smart contract after execution of each computational operation by a token manger in a master node and after execution of each computational operation, reducing each balance amount of token by each amount of token used in parallel by one or more token updaters in each slave node and reporting the each balance amount reduced to the token manager. The step of controlling further comprises reporting lack of amount of token to the data customer in case that the balance amount of token which the data customer currently has is not enough for executing each computational operation. The step of controlling further comprises reporting a result of execution to the data customer in case full execution of the code requested is completed.

In a second aspect of the proposed technology, the objects are achieved by a data provider using a blockchain which comprises a data storage system configured to store a plurality of data sets in a distributed manner using a cluster of nodes, a data processing system configured to process the plurality of data sets in a distributed manner using the cluster of nodes, a token control system configured to control processing of the data processing system based on a token in a distributed manner using the cluster of nodes and a blockchain application configured to deploy a smart contract to the blockchain. In a case that the deployed smart contract receives a request for executing code for certain data sets from a data customer, the deployed smart contract estimates amount of token required for executing the code for the certain data sets where the code to be executed includes a set of computational operations. The token control system controls, in a distributed manner using the cluster of nodes, execution of the code for the certain data set based on balance amount of token while the balance amount of token is greater than the estimated amount of token. The balance amount of token is updated after execution of each computational operation in a distributed manner.

The token control system controls the data processing system to execute each computational operation for the selected data sets in a distributed manner, and updates the balance amount of token by reducing amount of token used after execution of each computational operation in a distributed manner, and repeats the executing and the updating as long as the balance amount of token is enough for further execution of each computational operation based on the estimated amount of token and until getting a result of full execution of the code requested by the data customer.

The cluster of nodes includes a master node and one or more slave nodes. The executing and the updating are performed in the one or more slave nodes in a distributed manner and the repeating is controlled in the master node in a distributed manner.

The blockchain application publishes metadata of available data sets among the plurality of data sets stored in the data storage system to the blockchain. The metadata includes information of identifying each data set in the data storage without moving actual data set from the data storage system to the data customer.

The deployed smart contract responds to the request for execution from the data customer with access key thereto. The deployed smart contract also estimates amount of token required for execution based on data type of each data set and code type of each computational operation to be executed. The deployed smart contract estimates amount of token required for execution by looking up a token estimation table. The deployed smart contract compares balance amount of token which the data customer currently has with the amount of token estimated for execution and controls the data processing system to allow or terminate execution of the code for the selected data sets based on result of comparison.

The token control system monitors whether the balance amount of token which the data customer currently has is enough for executing each computational operation and controls whether to continue execution of each computational operation or not. The token control system comprises a token manager configured to manage total balance amount of token and report the total balance amount to the deployed smart contract after execution of each computational operation in a master node, and one or more token updaters configured to reduce each balance amount of token by each amount of token used in parallel in each slave node after execution of each computational operation; and reporting the each balance amount reduced to the token manager. The token manager reports lack of amount of token to the data customer in case that the balance amount of token which the

data customer currently has is not enough for executing each computational operation. The token manager also reports a result of execution to the data customer in case full execution of the code requested is completed.

In a third aspect of the proposed technology, the objects are achieved by a data customer using a blockchain comprising a blockchain application configured to request for published metadata of available data sets among a plurality of data sets in a data storage system to the blockchain, and a token application configured to select certain data sets using the metadata of available data sets and request for executing code for the selected data sets with amount of token which the data customer currently has to a smart contract. The token application receives an access key from the smart contract and writes the code for the selected data sets. The code includes a set of computational operations. The metadata includes information of identifying each data set in the data storage system.

The data customer views only metadata of available data set, the available data sets which are stored in a data provider and sends the code to be executed to the data provider which has available data sets.

Technical Advantage

By the proposed technology, an effective framework which integrates the advantages of blockchain and big data, is provided for analyzing data in a large scale of data storage. A more efficient way for handling large data with parallel computing is provided. The security and the privacy of data is improved for the data provider which owns the data without sending the data itself to an external network rather receiving codes to be executed from the data customer. A controlled computation based on the computational complexity is provided for the data provider to prevent infinite loop, large delays to the network, or mal functions such DOS attacks.

BRIEF DESCRIPTION OF THE DRAWINGS

A more complete understanding of the above mentioned and other features and advantages of the present invention will be apparent from the following detailed description of the drawings, wherein:

FIG. **1** schematically illustrates an example of data analyzing system using blockchain.

FIG. **2** illustrates a diagram of a data provider using blockchain.

FIG. **3** shows an example embodiment of a data provider which is implemented in a distributed manner.

FIG. **4** illustrates a diagram of a data customer using blockchain.

FIG. **5** shows an example embodiment of interaction between smart contract, data customer and data provider.

FIG. **6** shows an example embodiment of workflow of data provider, data customer, smart contract and blockchain.

FIG. **7** illustrates the workflow for analyzing data using a blockchain.

FIG. **8** shows an example embodiments of workflow for a token control system of the data provider in a distributed manner.

ITEM LIST

100 data customer
110 blockchain application of data customer
200 blockchain
300 smart contract
400 data provider
410 blockchain application of data provider
450 data processing system
470 data storage system
500 token control system
510 token manager
520, **540**, **560** token updaters
600 token estimation table

DETAILED DESCRIPTION

A generalized embodiment is described below. It provides a solution for an improved method for analyzing data using a blockchain, a data provider and a data customer therefore with improved protection of the security and the privacy of data.

An aspect of our proposed approach is to move computational code to the data sets in a large scale of data storage, without transferring the data sets themselves to an external network. Conventional methods for analyzing data demands the data should be moved across networks for analysis. Along with it being highly network intensive, data owners for the data provider are also reluctant to provide their data due to security and privacy breaches.

Another aspect of our proposed approach is to utilize one of frameworks for big data which allows computation on large data sets through parallel computing in a distributed manner using a cluster of nodes.

The third aspect of our proposed approach is to adapt the blockchain to guarantees each transaction secured and tamperproof. Since conventional blockchain is limited to handling large data and parallel computation. Our proposed approach integrates blockchain into big data system in distributed way and complements them each other. Therefore, it provides new way of computing upon large data sets without the moving the data sets themselves across the network. By the proposed approach provides a solution to allow a safer way for improved protection of the security and the privacy of data and a more efficient way for handling large data and parallel computations.

The fourth aspect of our proposed approach is to provide a concept of a token for controlled computation, which prevents the execution of malicious functions in codes given by the data customer, by putting constraints of computational operation. Pre-defined token is assigned to the authorized data customer based on its needs for computation. Smart contract performs pre-checks on the code submitted by the data customer and an amount of token which the data customer currently has. The smart contract also estimates the amount of token required for executing the given code by the data customer. A token manager and token updaters are introduced to coordinate token value usage during computation of user code until it exits gracefully with normal result of full execution of the user code, or the assigned amount of token gets exhausted.

FIG. **1** schematically illustrates an example of data analyzing system using blockchain. The data analyzing system using block chain includes a data customer (**100**) and a data provider (**400**) in blockchain (**200**). A smart contract (**300**) is deployed by the data provider to the blockchain (**200**).

This embodiment has two core components. The first component is a data customer (**100**) which can view a metadata of available data sets in the data storage which is provided form the data provider through the smart contract and request permission for execution code for a set of computational operations on a selected data sets in the data storage. A metadata of available data sets includes information of identifying each data set in the data storage. Actual

data set is not transferred to the data customer (**100**) but only the metadata of the actual data set is provided to the data customer (**100**) for selecting a set of data sets from the available data sets. Instead of transferring actual data set, the code to be executed can be transferred to the data provider (**400**).

The token represents an entity which can control computational complexity of the code requested by the data customer. The token can also control malicious functions of the code given by the data customer (**100**). Within a certain amount of token the code requested by the data customer can be executed in the data provider. Thereby prevents delays on the network. An amount of token required for executing code can be assigned to the data customer separately, which is out of scope of the present invention and not explained here in detail. During execution time, depending on the computational complexity of the code given by the data customer, a certain amount of token is reduced after each execution of the code. The execution of code will continue until it exits gracefully with a result of full execution or when the amount of token exhausts. The usage of token is monitored continuously by the data provider (**400**).

The request for executing code can includes code to be executed for the selected data set and the amount of token which the data customer (**100**) currently has. An access key may be provided by a data provider (**400**) through a smart contract (**300**) according to the request from the data customer (**100**).

The second component is a data provider (**400**) which can deploy a smart contract (**300**). The data provider provides access key for the deployed smart contract (**300**) to the data customer (**100**) if the data customer is authorized to a blockchain (**200**). The data customer (**100**) will submit the computational code that need to be executed on the selected dataset. If the data customer has enough amount of token in their account, the data provider can control the execution of the code on the actual data set which is maintained by the data provider (**400**).

The smart contract (**200**) which was deployed by the data provider (**400**) will perform a preliminary check of the code to be given from the data customer (**100**). The smart contract also can evaluate whether the amount of token provided by the data customer (**100**) satisfies the computational needs of the code provide by the data customer. The smart contract (**300**) control for the data provider (**400**) to execute the code within the amount of token the data customer currently has.

FIG. **2** illustrates a diagram of a data provider using blockchain.

Referring to FIG. **2**, the data provider (**400**) includes a data storage system (**470**), a data processing system (**450**), a token control system (**500**) and a blockchain application (**410**). The token estimation table (**600**) can be included in the data provider (**400**).

The data storage system (**470**) can store a plurality of data sets in a distributed manner using a cluster of nodes. The data processing system (**450**) can process the plurality of data sets in a distributed manner using the cluster of nodes. The token control system (**500**) can control processing of the data processing system based on an amount of token in a distributed manner using the cluster of nodes. The blockchain application (**410**) can deploy a smart contract (**300**) to the blockchain (**200**). In a case that the deployed smart contract (**300**) receives a request for executing code for certain data sets from a data customer (**100**), the deployed smart contract of the data provider (**400**) estimates amount of token required for executing the code for the certain data sets. The code to be executed includes a set of computational operations such as arithmetic, assignment, relational, or read operations.

The blockchain application (**410**) can publish a metadata of available data sets among the plurality of data sets stored in the data storage system (**470**) to the blockchain (**200**). The metadata includes information of identifying each data set in the data storage without moving actual data set from the data storage system to the data customer (**100**).

The token control system (**500**) can control, in a distributed manner using the cluster of nodes, execution of the code for the certain data set based on balance amount of token while the balance amount of token is greater than the estimated amount of token. The balance amount of token is updated after execution of each computational operation in a distributed manner.

The token control system (**500**) can control the data processing system (**450**) to execute each computational operation for the selected data sets in a distributed manner, and update the balance amount of token by reducing amount of token used depending on the computational complexity of the given code after execution of each computational operation in a distributed manner. The process of the executing and the updating of the data processing system (**450**) can be repeated as long as the balance amount of token is enough for further execution of each computational operation based on the estimated amount of token and until getting a result of full execution of the code requested by the data customer.

FIG. **3** shows an example embodiment of a data provider which is implemented in a distributed manner.

Referring to FIG. **3**, the data provider (**400**) is comprised by a cluster of nodes which includes a master node (MASTER NODE) and one or more slave nodes (SLAVE NODE 1, 2, . . . , n). The executing and the updating are performed in the one or more slave nodes in a distributed manner and the repeating is controlled in the master node in a distributed manner. The data storage system (**470**), the data processing system (**450**) and the token control system (**500**) can have a master-slave structure which allows the data and the token be stored, processed and controlled in a distributed manner. Thereby a large scale of data can be handled efficiently.

The token control system (**500**) can monitor whether the balance amount of token which the data customer currently has is enough for executing each computational operation and control whether to continue execution of each computational operation or not.

The token control system (**500**) comprises a token manager (**510**) and one or more token updaters (**520**, **540**, **560**). The token manger (**510**) can manage total balance amount of token and report the total balance amount to the deployed smart contract (**300**) after execution of each computational operation in a master node. One or more updaters (**520**, **540**, **560**) can reduce each balance amount of token by each amount of token used for computational operation depending on the computational complexity in parallel in each slave node after execution of each computational operation and report the each balance amount reduced to the token manager (**510**). The token manager (**510**) can report lack of amount of token to the data customer (**100**) in case that the balance amount of token which the data customer currently has is not enough for executing each computational operation. The token manager (**510**) can also report a result of execution to the data customer (**100**) in case full execution of the code requested is completed.

FIG. **4** illustrates a diagram of a data customer using blockchain.

Referring FIG. **4**, a data customer (**100**) includes a blockchain application (**110**) and a token application (**130**). The blockchain application (**110**) can request for a published metadata of available data sets among a plurality of data sets in a data storage system to the blockchain (**200**). The token application (**130**) can select certain data sets using the metadata of available data sets and request for executing code for the selected data sets with amount of token which the data customer (**100**) currently has to a smart contract (**300**). The token application (**130**) can receive an access key from the smart contract (**300**) and provide the code to be executed for the selected data sets into the smart contract (**300**). The code includes a set of computational operations. The metadata includes information of identifying each data set in the data storage system.

The data customer (**100**) can view only metadata of available data sets which are stored in a data provider (**400**) instead of receiving actual data set from the data provider (**400**). The data customer (**100**) can send rather the code to be executed to the data provider (**400**) which has available data sets through the smart contract (**300**). The data customer can also use a software development kit for the token application to create code for computation for example a MapReduce code and then that code will be evaluated in the smart contract for preliminary check of required token. Thereby the security and the privacy of data can be improved.

FIG. **5** shows an example embodiment of interaction between smart contract, data customer and data provider.

Referring FIG. **5**, the deployed smart contract (**300**) can respond to the request for execution from the data customer (**100**) with access key thereto. The deployed smart contract (**300**) also estimates amount of token required for execution based on data type of each data set such as integer, double, date, etc. and code type of each computational operation to be executed such as arithmetic, assignment, relational, or read operations. The deployed smart contract (**300**) estimates amount of token required for execution by looking up a token estimation table (**600**).

The deployed smart contract (**300**) compares balance amount of token which the data customer (**100**) currently has with the amount of token estimated for execution and controls the data processing system (**450**) to allow or terminate execution of the code for the selected data sets based on result of comparison.

Furthermore, the deployed smart contract (**300**) can receive an updated balance amount of token after execution of each computational operation from the token manager (**510**) and controls the data processing system (**450**) to allow or terminate execution of the code.

The deployed smart contract (**300**) which is shown in FIG. **5** is merely an example of an embodiment and can be adapted or modified variously.

FIG. **6** shows an example embodiment of workflow of data provider, data customer, smart contract and blockchain.

Referring FIG. **6**, an authorized data customer (**100**) which can be an entity, or an organization can perform computation on an available data set using blockchain consortium. The data customer is authorized by registering or enrolling to the blockchain consortium. When the data customer (**100**) is authenticated, the data customer (**100**) can view and request (S**101**) the corresponding metadata of data sets in a data storage and computational operation facility provided by the data providers (**400**). Based on the meta data and the computational requirements of the data customer, the amount of token needs can be estimated later by the data provider (**400**). Depending on the network, the data customer (**100**) can acquire certain amount of token from the blockchain consortium (**200**). The data provider (**400**) can deploy a smart contract (**300**) to blockchain. If the request is valid, the data customer (**100**) will get a response (S**404**) which contains access key to the smart contract (**300**). The data customer can request for execution of code by providing the code to be executed on the selected data set along with the access key and the amount of token currently has. The smart contract which was deployed by the data provider can perform a preliminary check of the code given by the data customer. The smart contract can estimate (S**406**) the amount of token required for execution of the given code for the selected data set and evaluate (S**408**) whether the provided token satisfies the computational needs of the provided code. The smart contract can use a token estimation table to find the amount of token estimated. A detailed description of the token estimation table (**600**) is given in FIG. **8**. If the amount of token available is greater than the amount of token estimated, the data provider (**400**) will allow the execution of the provided code on the provided data set. If the preliminary check passes, the actual computation will take place in a customized computational framework which consists of a token control system (**500**), a data storage system (**470**) and a data processing system (**450**) in a mater node and one or more slave nodes. The customized computational framework of token control system includes a token manager (**510**) and one or more token updaters (**520**, **540**, **560**). A detailed description of the customized computational framework of token control system is given in FIG. **8**.

For each execution of a set of computational operations, the slave node will report the executed computational operation (S**410**) to the token updaters (**520**, **540**, **560**) in the slave node and in turn the token updaters send a response to the token manager in the master node. The token manager (**510**) will calculate (S**414**) the available balance amount of token based on the reports from all the updaters and send the updated balance amount of token back to all the updaters. When each token updater gets the response from the token manager and then sends a signal to each slave node for executing next set of computational operations. This process will be repeated until the execution of all the given code for computational operations is completed. Once a set of computational operations is executed in each slave node, the token updaters (**520**, **540**, **560**) in each slave nodes will update the balance amount of token to the token manager (**510**). Whenever the balance amount of token is not enough or less than the amount of token estimated for certain set of computational operations, the token manager (**510**) can immediately report (S**412**) an "out of token" status to the data customer through the smart contract and send a signal to the master node to immediately stop the whole execution. After the execution of each computational operations, the token manager (**510**) can send the amount of token used for that set of computational operations to the smart contract (**300**) in a blockchain network as a transaction. Then, the smart contract (**300**) which is already deployed in blockchain consortium will check whether the data customer (**100**) still has enough token for further execution of computational operations. If the data customer (**100**) has enough balance amount of token available, then control goes to the master node (**510**) to execute further computational operations. If the code required for executed on the selected data set finishes normally or the balance amount of token of the data customer is exhausted (S**416**), the corresponding result will be reported (S**422** or S**424**) to the data customer (**100**) via blockchain.

FIG. **7** illustrates the workflow for analyzing data using a blockchain.

Referring FIGS. **6** and **7**, the data customer (**100**) can enroll to the blockchain (**200**) and be authorized. The data provider (**400**) can publish metadata of available data set in the data storage to the blockchain.

The data customer (**100**) can request (S**101**) the published metadata of data sets to the blockchain. The metadata includes information of identifying each data set in the data storage system (**470**) without moving any actual data set from the data storage system (**470**) to the data customer (**100**).

The data provider (**400**) can deploy (S**402**) a smart contract (**300**) to the blockchain. According to the request for metadata from a data customer (**100**) it respond (S**404**) to the request with access key.

The data customer (**100**) can request (S**103**) for executing code including a set of computational operations by providing the code to be executed into the smart contract (**300**) and a balance amount of token which the data customer currently has. The data customer can further send selection information of data sets.

The data provider (**400**) can estimate (S**406**) an amount of token required for executing the code for the selected data sets in the data storage. The amount of token can be estimated and based on data type of each data set and code type of each computational operation to be executed. The token estimation table (**600**) can be looked up by the data provider for estimation. The detailed process of estimation will be described later with FIG. **8**.

The data provider (**400**) can compare (S**408**) the estimated amount of token with the balance amount of token to control the execution of the code for the selected data sets based on the balance amount of token while the balance amount of token is greater than the estimated amount of token.

The data provider (**400**) can allow the execution of the code for the selected data sets if the balance amount of token is greater than the estimated amount of token. Otherwise the data provider (**400**) reports (S**412**) the status "out of token" to the data customer (**100**).

If the data provider allows the execution of the code, the data provider executes (S**414**) each computational operation included in the code in a distributed manner using a master node and one or more slave nodes.

Then the data provider can update (**416**) the balance amount of token by reducing amount of token used after executing each computational operation in a distributed manner. The data provider can also report the amount of token updated to smart contract (**300**).

The data provider can control (S**416**) to repeat the steps of executing (S**410**) and updating (S**414**) until the balance amount of token updated is greater than the token required for execution of the code in a distributed manner using a cluster of nodes. The cluster of nodes includes a master node and one or more slave nodes. The steps of execution (S**410**) and updating (S**414**) can be performed in the one or more slave nodes in said distributed manner and the step of repeating is controlled in the master node in said distributed manner as shown in FIGS. **3** and **4**.

The step of controlling can further comprises managing a total balance amount of token and reporting the total balance amount to the deployed smart contract (**300**) after execution of each computational operation by a token manger (**510**) in a master node, and after execution of each computational operation, reducing each balance amount of token by each amount of token used in parallel by one or more token updaters (**520**, **540**, **560**) in each slave node and reporting the each balance amount reduced to the token manager (**510**). The step of controlling can further comprise reporting (S**418**) lack of amount of token to the data customer (**100**) in case that the balance amount of token which the data customer (**100**) currently has is not enough for executing each computational operation. The step of controlling further comprises reporting (S**424**) a result of execution to the data customer (**100**) in case full execution of the code requested is completed.

FIG. **8** shows an example embodiment of workflow for a token control system (**500**) of the data provider in a distributed manner.

Referring FIG. **8**, the data provider (**400**) provides both a large scale of data sets by the data storage system (**470**) and a platform for computation of the code by the data processing system (**450**) and the token control system (**500**). The platform can be called as "Customized Computational Framework (CCF)". As shown in FIG. **3**, in addition to the data storage system (**470**) and the data processing system (**450**) implemented in a distribute manner using a master node and a plurality of slave nodes, the token control system (**500**) including a token manager (**510**) and a plurality of token updaters (**520**, **540**, **560**) can be also implemented in a distributed manner using a master node and a plurality of slave nodes. Here the workflow of the token control system (**500**) which includes a token manager (**510**) and a plurality of token updaters (**520**, **540**, **560**) is described.

After the preliminary check for the estimation of token required for the execution of the code, the control goes to the CCF master node. The master node which is responsible to assign tasks for the plurality of slave nodes, start to send the required computational operations and data. As only an example of computational operations, map/reduce function will be used here for explanation, however, various type of computation operations can be used in a similar way.

Step 1.1

From blockchain an balance amount of token available and the estimated token required for executing the entire task will be given to the token manager (**510**). The token manger (**510**) is responsible to calculate the total used amount of the token after execution of each computational operations and give instruction to the master node whether to continue the execution further or not.

Step 1.2

Simultaneously the actual computation operation to be executed on the requested data set will be sent to the master node. In the workflow, the code contains three map functions M**1**, M**2** and M**3** and two reduce functions R**1** and R**2**. These functions must be in a customized map-reduce format according to the present invention.

Step 2.1

The master node will give instruction to all the slave nodes to execute the first map function M**1**.

Step 2.2

Simultaneously the master node will inform the token manger (**510**) that the instruction to execute the first function is given to all slave nodes.

Step 3.1

The token manager (**510**) sends the available token information to all the token updaters (**520**, **540**, **560**).

Step 3.2

Each token updater (**520**, **540**, **560**) will have the information regarding the function to be executed, M**1** and the data type of the requested data set from the respective slave node where each token updater is associated with.

Step 4

Each token updater (**520**, **540**, **560**) will estimate an amount of token required to execute the function M**1** on the requested data set. If the balance amount of token available is enough for the estimated amount of token for M**1**, then go to step 5. Otherwise go to step 15.

Step 5

An instruction is sent to the corresponding slave node to start the execution of the function.

Step 6

A function contains a set of computational operations which need to be executed. the computational operators can include any type of operations such as arithmetic, logical, relational, assignment, bitwise, etc. After executing each computational operation, the slave node will inform the corresponding token updater.

Step 7

The updater will calculate the amount of token used for execution of the computational operation on the given data type.

Step 8

The amount of token used for current computational operation will be updated to the token manager (**510**).

Step 9

In the token manger, the token updaters update the amount of token used and calculate the balance amount of token currently available. If the balance amount of token currently available is not enough for further operation then the control goes to step 15, which is to stop the execution immediately.

Step 10

After the calculating the balance amount of token currently available, it will be reported to the token updaters.

Step 11

If the token updater receives the signal from the token manager, which indicates that there is enough amount of token for further execution, these updaters will again send a signal to the corresponding slave nodes to continue the execution. It then repeats step 5 to step 11 until the completion of that function or termination due to lack of balance amount of token.

Step 12.1

The master node signals the token manager that the execution of that particular function completed successfully.

Step 12.2

Simultaneously the token manager updates the blockchain with the balance amount of token currently available.

Step 13

While sending the available token as a transaction to the blockchain, the smart contract will double check if the token balance is empty or not. If the token balance is not empty, that is enough, the control goes back to token manager and continues from step 14. Otherwise data customer will get the information "Out of Token" and hence the connection is closed.

Step 14

Otherwise, it will signal the master node to release the next function in the code. It can be a map or a reduce function, for example M**2** map function. Thus, the entire process will repeat from step 2.1 to step 13, the same as for the M**1** function.

Step 15.1

When the available token is empty, the following two actions will be taken place simultaneously. The first is to send an update to blockchain with the information of "Out of Token".

Step 15.1

The second is to send a signal to master node to stop the entire execution immediately. If a final result from last reduce function, R**2**, is available, the result will be published to the data customer through the blockchain and the connection will be closed.

Token Estimation Table

Hereinafter, an example of token estimation table is disclosed and it is explained how to evaluate the estimated token required for executing the code.

Token is an entity which controls the computational complexity of the computational operation given by the data customer. There can be some standard token value for each type of computational operations to be performed. These values can be the same for all the data providers present in the blockchain consortium. The table which describes the token requirement for performing each computational operation is known as the Token Estimation Table. The smart contract (**300**), which is executed before the required computation, will evaluate an estimated amount of token according to the requested code by the data customer. For this estimation, the smart contract can use the information in the token estimation table. The amount of token required can be estimated based on the code type of various computational operations in the code and data types of the selected data sets. The estimation table can include all the supported computational operations on the data. For example, the computational operation of adding two values with "double integer" data type would require more amount of token compared to that of "integer" data type. The data type eventually shows the bytes required to store the data.

Hence, it can be represented as a function of operator and the number of bytes required. The set of computational operations is given as O: {o1, o2, o3, . . . oo}, which includes the operation of arithmetic, logical, relational, assignment, bitwise, etc. Each operator in O has a weightage, W: {w1, w2, w3, . . . wo} corresponding to the computational complexity, which is further used for token estimation. The set of supported data types is given as D: {d1, d2, d3, dd} and the corresponding bytes required for each data type as B: {b1, b2, b3, . . . bd}.

A general formula for estimating token required can be written as:

$$\text{Estimated amount of token} = \Sigma_{i=1 \text{ to } n} w(oi) \times b(di) \qquad \text{[Equation 1]}$$

where, n represents the number of computational operations in the code, oi ∈ O and di ∈ D.

Smart Contract: Eval(Code*)

Eval(code*) on the smart contract performs a preliminary check to find the estimated amount of token required for the given code by the data customer. A sample calculation is shown below to demonstrate the actual execution of Eval (code*) according to the equation defined for estimated token. The data set in [Table 1] has three columns which contain the date, the customer id (C_id) with int data type and the bill amount (Bill) with the double data type.

TABLE 1

Sample data set Data set		
Date	C_id (int)	Bill (double)
25 Feb. 2019	01	100
25 Feb. 2019	02	200
25 Feb. 2019	03	150
26 Feb. 2019	02	100

TABLE 1-continued

Sample data set Data set		
Date	C_id (int)	Bill (double)
26 Feb. 2019	03	50
27 Feb. 2019	02	100

The code in [Table 2] contains a map and reduce function. The reduce function should have the input as a (key, [value]) pair, key represents each unique C_id and value represents an array of all Bills corresponding to each C_id. Reduce function here is, to sum up, all elements in each array.

TABLE 2

Sample pseudo code	
Function	Pseudo code
Map	f( ) {find (this.C_id, this.Bill)}

TABLE 2-continued

Sample pseudo code	
Function	Pseudo code
Reduce	f(key,values) {array.sum(values)}

The token estimation table is shown in [Table 3], which gives information regarding the weightage of each operator. It includes all the valid operators that can be used during execution. It also contains the number of bytes required for each data type.

TABLE 3

Sample Token estimation table Totem Estimator Table			
Operator		Datatype	
Operator	Weightage	Datatype	Bytes
Arithmetic: +	2	int	4
Assignment: #	1	double	8
Relational: #	1	date	3
Read	1	#	#
#	#	#	#

The estimation determines the required token which is demonstrated in the Eval(Code*) table in [Table 4]. According to the example, map function will read each row in the data set, i.e., a total of 6 rows. The weightage of the read operator and the data types involved in a row determines the required token for that statement to execute. Then it will have each tuple as (C_id, Bill), that requires only read operations per row.

TABLE 4

Sample estimation Eval(Code*)						
	n	Operator	$w(o_i)$	Data type (d_i)	$b(d_i)$	$w(o_i) * b(d_i)$
Map function						
6 rows: Repeat (n = 1, 2, 3,) 6 times	1	Read a row	1	int, double, date	3 + 4 + 8	15
n = 1 to 6			6*15			90
Reduce function						
	7	Read first row	1	int, double	4 + 8	12
	8	Read second row	1	int, double	4 + 3 * 8	28
	9, 10	Arithmetic: '+' (three elements)	2, 2	double	8, 8	32
	11	Read third row	1	int, double	4 + 2 * 8	20
	12	Arithmetic: '+' (two elements)	2	double	8	16
n = 7 to12						108

$\Sigma_{i=1 \text{ to } n} w(o_i) \times b(d_i) = 90 + 108 = 198$

The output of the map function will be sorted and shuffled in the data storage system (**470**). The input of reducer function will be a tuple (key, [value]) pair. Reading each input requires same effort as mentioned in the map function. After each read, the sum of Bills related to each C_id will be executed. C_id: 1 has only one Bill and thus no '+' operator required. C_id: 2 has three Bills and thus two '+' operator are required. Therefore, the required token will be twice the weightage of addition operation times bytes required to store the data type of Bill (double). Finally, C_id: 3 has only two bills, hence one '+' operator. The sum of required token for each operator mentioned here will give the total token required to execute the entire code* by the [Equation 1].

In summary, a set of embodiments might provide methods for analyzing data using a blockchain (**200**), wherein a plurality of data sets is stored and processed in a data storage in a distributed manner using a cluster of nodes. In an aspect, the method might comprise steps of: deploying (S**402**) a smart contract (**300**) to the blockchain by the data provider (**400**); receiving (S**404**, S**103**) a request for executing code for data sets selected by a data customer (**100**), wherein the

request for executing code includes code to be executed and a balance amount of token which the data customer currently has, where the code to be executed includes a set of computational operations; estimating (S406) an amount of token required for executing the code for the selected data sets in the data storage; and controlling (S408), in said distributed manner using the cluster of nodes, execution of the code for the selected data sets based on the balance amount of token while the balance amount of token is greater than the estimated amount of token, wherein the balance amount of token is updated after execution of each computational operation in said distributed manner, wherein the amount of token represents number of units for an entity which controls computational complexity of the code requested by the data customer.

In some methods, the step of controlling (S408) further comprises steps of executing (S410) each computational operation in the code for the selected data sets in said distributed manner; updating (S414) the balance amount of token by reducing an amount of token used after executing each computational operation in said distributed manner; and repeating (S416) the steps of executing (S410) and updating (S414) as long as the balance amount of token is enough for further execution of each computational operation based on the estimated amount of token and until getting a result of full execution of the code requested by the data customer. Additionally and/or alternatively, the cluster of nodes includes a master node and one or more slave nodes and the steps of execution (S410) and updating (S414) are performed in the one or more slave nodes in said distributed manner and the step of repeating is controlled in the master node in said distributed manner.

In some methods, the method might further comprise, before the step of deploying (S402), publishing (S400) metadata of available data sets among the plurality of data sets stored in the data storage to the blockchain, wherein the metadata includes information of identifying each data set in the data storage system (470) without moving any data set from the data storage system (470) to the data customer (100).

In any of the methods described above, the step of estimating (S406) estimates an amount of token required for execution might be based on data type of each data set and code type of each computational operation to be executed. Alternatively and/or additionally, in any of these methods, the step of estimating (S406) estimates the amount of token required for execution by looking up a token estimation table, and/or the step of estimating (S406) includes comparing balance amount of token which the data customer (100) currently has with the amount of token estimated for execution; and controlling to allow or terminate execution of the code for the selected data sets based on result of comparison.

In any of the methods described above, the step of controlling can include monitoring whether the balance amount of token which the data customer (100) currently has is enough for executing each computational operation; and controlling whether to continue execution of each computational operation or not. In some methods, the step of controlling further comprises: managing a total balance amount of token and reporting the total balance amount to the deployed smart contract (300) after execution of each computational operation by a token manger (510) in a master node; and after execution of each computational operation, reducing each balance amount of token by each amount of token used in parallel by one or more token updaters (520, 540, 560) in each slave node and reporting each balance amount reduced to the token manager (510). In some methods, the step of controlling further comprises: reporting (S418) lack of amount of token to the data customer (100) in case that the balance amount of token which the data customer (100) currently has is not enough for executing each computational operation; and reporting (S424) a result of execution to the data customer (100) in case full execution of the code requested is completed.

Another set of embodiments provides data providers (400) using a blockchain (200) A data provider (400) might comprise a data storage system (470) configured to store a plurality of data sets in a distributed manner using a cluster of nodes; a data processing system (450) configured to process the plurality of data sets in a distributed manner using the cluster of nodes; a token control system (500) configured to control processing of the data processing system (450) based on a token in a distributed manner using the cluster of nodes; and a blockchain application (410) configure to deploy a smart contract (300) to the blockchain (200).

In a case that the deployed smart contract (300) receives a request for executing code for certain data sets from a data customer (100), the deployed smart contract (300) estimates amount of token required for executing the code for the certain data sets where the code to be executed includes a set of computational operations, wherein the token control system (500) controls, in a distributed manner using the cluster of nodes, execution of the code for the certain data set based on balance amount of token while the balance amount of token is greater than the estimated amount of token wherein the balance amount of token is updated after execution of each computational operation in a distributed manner.

In some data providers (400) the token control system (500) controls the data processing system (450) to execute each computational operation for the selected data sets in a distributed manner, and updates the balance amount of token by reducing amount of token used after execution of each computational operation in a distributed manner, and repeats the executing and the updating as long as the balance amount of token is enough for further execution of each computational operation based on the estimated amount of token and until getting a result of full execution of the code requested by the data customer. In some data providers, the token control system (500) might comprise a token manager (510) configured to manage total balance amount of token and report the total balance amount to the deployed smart contract (300) after execution of each computational operation in a master node; and

one or more token updaters (520, 540, 560) configured to reduce each balance amount of token by each amount of token used in parallel in each slave node after execution of each computational operation; and reporting the each balance amount reduced to the token manager (510).

Another set of embodiments provides data customers (100). On such data customer (100) using a blockchain (200) might comprise a blockchain application (110) configured to request for published metadata of available data sets among a plurality of data sets in a data storage system (470) to the blockchain; and a token application (130) configured to select certain data sets using the metadata of available data sets and request for executing code for the selected data sets with amount of token which the data customer (100) currently has to a smart contract (300), wherein in case that the token application (130) receives an access key from the smart contract (300), the token application (120) writes the code for the selected data sets into the smart contract (300), and wherein the code includes a set of computational

operations and the metadata includes information of identifying each data set in the data storage system (**470**). Some data customers (**100**) might view only metadata of available data set, the available data sets which are stored in a data provider (**400**), and sends the code to be executed to the data provider (**400**) which has available data sets.

The present invention is not limited to the above-described preferred embodiments. Various alternatives, modifications and equivalents may be used. Therefore, the above embodiments should not be taken as limiting the scope of the invention, which is defined by the appending claims.

The invention claimed is:

1. A method for analyzing data using a blockchain,

wherein a plurality of data sets is stored and processed in a data storage in a distributed manner using a cluster of nodes, the method comprising:

deploying a smart contract to the blockchain by a data provider, the smart contract associated with an access key configured to authorize access to the smart contract;

receiving a request for executing code from a data customer for data sets of the data provider selected by the data customer, wherein the request for executing code includes the code from the data customer to be executed by the cluster of nodes, the access key, and a balance amount of token which the data customer currently has, where the code to be executed includes a set of computational operations;

estimating an amount of token required for executing the code for selected data sets in the data storage; and

controlling, in said distributed manner using the cluster of nodes, execution of the code for the selected data sets based on the balance amount of token while the balance amount of token is greater than an estimated amount of token, wherein controlling execution of the code for the selected data sets further comprises monitoring whether the balance amount of token which the data customer currently has is enough for executing each computational operation and, based on the monitored balance amount of token, controlling whether to continue execution of each computational operation or not; and

managing the balance amount of token, wherein the balance amount of token is updated in said distributed manner after execution of each computational operation at each respective node of the cluster of nodes, wherein the balance amount of token is reduced by a respective amount of token used in parallel by each respective node of the cluster of nodes;

wherein the amount of token represents number of units for an entity which controls computational complexity of the code requested by the data customer.

2. The method of claim **1**, wherein controlling further comprises:

executing each computational operation in the code for the selected data sets in said distributed manner;

updating the balance amount of token by reducing an amount of token used after executing each computational operation in said distributed manner; and

repeating executing and updating as long as the balance amount of token is enough for further execution of each computational operation based on the estimated amount of token and until getting a result of full execution of the code requested by the data customer.

3. The method of claim **2**, wherein the cluster of nodes includes a master node and one or more slave nodes and, executing and updating are performed in the one or more slave nodes in said distributed manner and repeating is controlled in the master node in said distributed manner.

4. The method of claim **1**, further comprising, before deploying:

publishing metadata of available data sets among the plurality of data sets stored in the data storage to the blockchain,

wherein the metadata includes information of identifying each data set in the data storage system without moving any data set from the data storage system to the data customer.

5. The method of any one of claim **1**, wherein estimating estimates an amount of token required for execution based on data type of each data set and code type of each computational operation to be executed.

6. The method of claim **1**, wherein estimating estimates the amount of token required for execution by looking up a token estimation table.

7. The method of claim **1**, wherein estimating includes comparing balance amount of token which the data customer currently has with the amount of token estimated for execution; and controlling to allow or terminate execution of the code for the selected data sets based on result of comparison.

8. The method of claim **1**, wherein controlling further comprises:

managing a total balance amount of token and reporting the total balance amount to the deployed smart contract after execution of each computational operation by a token manger in a master node; and

after execution of each computational operation, reducing each balance amount of token by each amount of token used in parallel by one or more token updaters in each slave node and reporting each balance amount reduced to the token manager.

9. The method of claim **1**, wherein controlling further comprises:

reporting lack of amount of token to the data customer in case that the balance amount of token which the data customer currently has is not enough for executing each computational operation; and

reporting a result of execution to the data customer in case full execution of the code requested is completed.

10. A data provider system using a blockchain comprising:

a data storage system configured to store a plurality of data sets in a distributed manner using a cluster of nodes;

a data processing system configured to process the plurality of data sets in a distributed manner using the cluster of nodes;

a token control system configured to control processing of the data processing system based on a token in a distributed manner using the cluster of nodes; and

a blockchain application configure to deploy a smart contract to the blockchain, the smart contract associated with an access key configured to authorize access to the smart contract,

wherein, the data provider system is configured to receive a request for executing code from a data customer for data sets of the data provider selected by the data customer, wherein the request for executing code includes the code from the data customer to be executed by the cluster of nodes, the access key, and a balance amount of token which the data customer currently has, wherein the code to be executed includes a set of computational operations;

wherein, when the deployed smart contract is configured to receive the request for executing code from the data customer for certain data sets of a data provider, the deployed smart contract estimates amount of token required for executing the code for the certain data sets,

wherein the token control system is configured to control, in a distributed manner using the cluster of nodes, execution of the code for the certain data set based on balance amount of token while the balance amount of token is greater than the estimated amount of token, wherein controlling execution of the code for selected data sets further comprises monitoring whether the balance amount of token which the data customer currently has is enough for executing each computational operation and, based on a monitored balance amount of token, controlling whether to continue execution of each computational operation or not, and wherein the token control system is configured to manage the balance amount of token, wherein the balance amount of token is updated in a distributed manner after execution of each computational operation at each respective node of the cluster of nodes, wherein the balance of amount of token is reduced by a respective amount of token used in parallel by each respective node of the cluster of nodes.

11. The data provider of claim **10**, wherein the token control system is configured to control the data processing system to execute each computational operation for the selected data sets in a distributed manner, is configured to update the balance amount of token by reducing amount of token used after execution of each computational operation in a distributed manner, and is configured to repeat the executing and the updating as long as the balance amount of token is enough for further execution of each computational operation based on the estimated amount of token and until getting a result of full execution of the code requested by the data customer.

12. The data provider of claim **10**, wherein the token control system comprises:

a token manager logic executable by the token control system configured to manage total balance amount of token and report the total balance amount to the deployed smart contract after execution of each computational operation in a master node; and

one or more token updater logic executable by the token control system to reduce each balance amount of token by each amount of token used in parallel in each slave node after execution of each computational operation; and reporting each balance amount reduced to the token manager.

13. A system comprising:

a data customer system using a blockchain comprising:

a blockchain application configured to request for published metadata of available data sets among a plurality of data sets of a data provider in a data storage system to the blockchain; and

a token application configured to select certain data sets using the metadata of available data sets and request for executing code for the selected data sets, the request including a balance amount of token which a data customer currently has, and an access key, to a smart contract; and

a data provider system using the blockchain comprising:

the data storage system configured to store the plurality of data sets in a distributed manner using a cluster of nodes;

a data processing system configured to process the plurality of data sets in a distributed manner using the cluster of nodes;

a token control system configured to control processing of a data processing system based on token in a distributed manner using a cluster of nodes;

the blockchain application configured to deploy the smart contract to the blockchain;

wherein the smart contract is configured to associate with the access key, and the access key is configured to authorize access to the smart contract,

wherein in a case that the token application is configured to receive the access key from the smart contract, the token application writes the code for the selected data sets into the smart contract,

wherein the code includes a set of computational operations and the metadata includes information of identifying each data set in the data storage system,

wherein, the data provider system is configured to receive a request for executing code from the data customer for data sets of a data provider selected by the data customer, wherein the request for executing code includes the code from the data customer to be executed by the cluster of nodes, the access key, and the balance amount of token,

wherein, when the deployed smart contract is configured to receive the request for executing code from the data customer for certain data sets of the data provider, the deployed smart contract estimates an amount of token required for executing the code for the certain data sets,

wherein the token control system is configured to control, in a distributed manner using the cluster of nodes, execution of the code for the certain data set based on the balance amount of token while the balance amount of token is greater than the estimated amount of token, wherein controlling execution of the code for the selected data sets further comprises monitoring whether the balance amount of token which the data customer currently has is enough for executing each computational operation and, based on a monitored balance amount of token, controlling whether to continue execution of each computational operation or not, and

wherein the token control system is configured to manage the balance amount of token, wherein the balance amount of token is updated in a distributed manner after execution of each computational operation at each respective node of the cluster of nodes, wherein the balance of amount of token is reduced by a respective amount of token used in parallel by each respective node of the cluster of nodes.

14. The data customer system of claim **13**,

wherein the data customer system is configured to view only metadata of available data set, the available data sets which are stored in a data provider, and sends the code to be executed to the data provider which has available data sets.

* * * * *