Blockchain: Research and Applications 3 (2022) 100081

Contents lists available at ScienceDirect

Blockchain: Research and Applications

journal homepage: www.journals.elsevier.com/blockchain-research-and-applications

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

Dhanya Therese Jose *, Jørgen Holme, Antorweep Chakravorty, Chunming Rong **

Department of Computer Science Electrical Engineering, University of Stavanger, 4036, Stavanger, Norway

ARTICLE INFO

Keywords:
Blockchain
Big data analytic
Hyperledger fabric
Hadoop
MapReduce
Docker
PIVT

ABSTRACT

The demand for electricity is increasing exponentially day by day, especially with the arrival of electric vehicles. In the smart community neighborhood project, electricity should be produced at the household or community level and sold or bought according to the demands. Since the actors can produce, sell, and buy according to the demands, thus the name prosumers. ICT solutions can contribute to this in several ways, such as machine learning for analyzing the household data for customer demand and peak hours for the usage of electricity, blockchain as a trustworthy platform for selling or buying, data hub, and ensuring data security and privacy of prosumers. TOTEM: Token for controlled computation is a framework that allows 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 the 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 lacks mechanisms to handle mass and concentrated power consumption. It is especially with the arrival of electric vehicles (EVs), the charging demands, and the peak overconsumption. To handle such peak hour issues without changing the existing structure of the grid, the 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 the sharing/trading of energy between households, neighborhoods, and communities. In order to address these kinds 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 environmentally friendly energy production and storage technologies and allow them to meet their own requirements or can sell it back to the grid. EV batteries can also play an important role in a way that they 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 solving 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 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 highly information-intensive data hub. Blockchain-based technologies can be used for recording energy generated, shared/traded, and stored in the storage level by prosumers. Each transaction in the blockchain will be transparent and secure [1]. In order to protect privacy and user 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, and crypto-techniques are required to prove the ownership of digital assets. Asymmetric encryption

* Corresponding author.
** Corresponding author.
E-mail addresses: dhanya.t.jose@uis.no (D.T. Jose), chunming.rong@uis.no (C. Rong).

https://doi.org/10.1016/j.bcra.2022.100081
Received 4 November 2021; Received in revised form 24 February 2022; Accepted 22 March 2022

and hash functions 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-organizing 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 consumption, production from each household/neighborhood, and energy demands, the prosumers will get a better understanding of the current trend. It shows that the analysis of data will improve the efficiency of the energy infrastructure. The analysis of data through conventional methods demands that the data be transferred across the network, which is highly network intensive, and due to security breaches, data owners may be cautious about providing their data. TOTEM [3,27], which stands for Token for controlled computation, is a patented framework (US11121874B2) that provides a decentralized solution for these concerns about the data transfer and analyses. This framework combines both blockchain technologies and big data systems. Each transaction through blockchain is secured and tamperproof but has limitations in handling large data and parallel computations. Big data technologies provide solutions for the 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 data sets 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 databases to allow data analysis from different communities or users without moving their data. In this article, a detailed explanation of how to utilize TOTEM for the EnergiX project is given and also shows the scope of an extended version of TOTEM which deals 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 of 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 Sections 5 and 6, respectively. The conclusion is given in Section 7.

2. Related works

Blockchain and big data technologies have been combined for various purposes. The 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 Ref. [28]. For reinforcing the security of big data platforms, a blockchain-based access control framework is proposed in Ref. [29]. This framework achieved objectives such as user-driven, transparency, fine granularity, pseudonymity, and unlink ability. However, while adopting blockchain technology to handle access control functions, additional critical issues emerged in the framework. A blockchain access control ecosystem that ensures a better way to manage access control of large data sets and how to avoid data breaches is proposed in Ref. [30]. The architecture built on a private and permissioned blockchain for a decentralized security system. Blockchain technology provides a solution to the challenges associated with traditional and centralized access control because it ensures 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 [31]. HBasechainDB makes it easy for organizations that have Hadoop ecosystem-based business logic to accommodate blockchain. HBasechainDB, which is built on the Hadoop ecosystem, also inherits efficient big data processing. In Ref. [32] Bandara et al. 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 that 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 prechecks 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 an updater are introduced to coordinate the computation of the user code until it exits gracefully, or the assigned TOTEM gets exhausted. In Ref. [4], it deals with the proof-of-concept of the new components, such as TOTEM managers and updaters introduced in the TOTEM framework.

Machine learning is a part of artificial intelligence that deals with the study of algorithms that allow the system to learn by itself and improve the results with experience [20]. Machine learning plays an important role in load balancing and the prediction of electricity usage. This will help in the 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 [5]; deals with the application of feature selection methods for the purpose of predicting household energy consumption. In this paper, they followed 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 [6] 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 Ref. [21]. Rudin et al. [22] proposed a framework where machine learning can be used for the prediction of failures of the system components. In some other works [23,24], 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 Ref. [25], by using the distributed spare attacks model, which is proposed in Ref. [26], and the 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 data set from smart meter data or household data, which is not secure and is not privacy preserved. The relevance of blockchain technology comes here because it guarantees secured transactions. A recent survey suggests that Decentralized applications (DApps) with blockchains promise no trust in authorities and overcome the key challenges of security and privacy problems [33]. Thus, blockchain in the EnergiX project acts as a platform for trading electricity according to the demand of prosumers, in a secured manner. Since big data plays a vital role in decision making for prosumers and blockchain acts as a trading platform, it is relevant to integrate both technologies for better and more efficient working in a secure environment. The properties of these two technologies can complement each other, which would enable a new way of computing upon large data sets in a secure 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 of 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 data sets that are too large to be efficiently interpreted, collected, managed, and processed using traditional methods or mechanisms [7]. Some of the main features of big data, such as volume, variety, velocity, and veracity [8], 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 are generated or the speed of the data generation and processing, and the data quality or value, respectively [9]. 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 data sets with clusters of computers and it is an open source implementation based on a programming model called MapReduce [10]. The Hadoop framework mainly consists of two layers, the Hadoop Distributed File System (HDFS) [11] and the previously mentioned distributed processing mechanism, MapReduce. The HDFS consists of a master-slave architecture, where the master node or the Name Node maintains the file systems metadata. The files are divided into fixed-size blocks and are stored in the slave node or Data Nodes. The mapping of blocks to a particular Data Node is determined by the Name Node according to some of the features, such as ease of access and free slot. Data Nodes are responsible for read-write operations in the file system. Data Nodes are also responsible for the creation, deletion, and replication of blocks, but all these operations are based on the instructions given by the Name Node. It will also send heartbeats to the Name Node periodically to indicate that the corresponding Data Node is currently active. The secondary Name Node is the node that keeps checkpoints of the file system metadata on the Name Node 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 is done by the 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 the reduce function. In the reduce function, the inputs combine and group 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 Job Tracker [12], which is responsible for monitoring the availability of the resources and allocating resources. It is also responsible for scheduling tasks for Task Trackers. Task trackers will compute the tasks and provide the status information back to the master or the job tracker at regular intervals of time. If a task tracker goes down, the job tracker will immediately reschedule the corresponding failed task to the next available task tracker, and if a job tracker 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 [13]. All the transactions recorded in the blockchain are transparent to all users in the corresponding blockchain network, and these transactions are tamperproof, 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; 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 [14]. After that, in 1992, Merkle tree was used for time-stamping several documents into one block. Later in 2009, it got more attention when Nakamoto introduced Bitcoin [15]. Bitcoin can be described as a peer-to-peer permissionless blockchain, which means it is completely decentralized. An example of a public blockchain is Ethereum. The public blockchain is also known as a permissionless blockchain because the participants in the network are anonymous and there are no restrictions on joining the network for the verification process [16]. 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 [17]. Blockchain can enable smart contacts, which were proposed by Szabo [18]. Blockchain-based smart contracts are computer programs that can be 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 the execution complexity of smart contracts to financial limitations [19].

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[1] 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.

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 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.

[1] https://www.hyperledger.org/use/fabric.

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.

i. A transaction proposal is sent to each endorsing peer, which will run and subsequently endorse the transaction before sending it to an orderer.
ii. 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.
iii. 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[2] 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[3]. The private data are stored in a separate private database on the authorized peers and are accessible from chain codes 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 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 Fig. 1, with a blockchain consortium, computation layer, and storage layer.

a. 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.
b. 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 node.
c. The storage layer contains the actual data collected from households, electric car consumption, wind farms, and other prosumer sources. It can be a file system such as HDFS and IPFS (InterPlanetary 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 data sets without accessing the data but by sending the code across the network toward 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 data set present in the data provider environment. Data providers are also responsible for deploying the necessary smart contracts. The smart contract will do a precheck 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 consists 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 updating 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. All the users can join. In Hyperledger Fabric, we have a membership service provider for membership. As we mentioned before, data providers will publish the metadata of the data set they own, and data consumers can view that information and send requests to analyze that particular data set 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: Scenario 1.

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 have 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, EVs, 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 meters, which need to be handled with privacy-preserving

[2] https://hyperledger-fabric.read the docs.io/en/release-1.4/private-data/private-data.html.
[3] https://hyperledger-fabric.readthedocs.io/en/release-1.4/gossip.html.

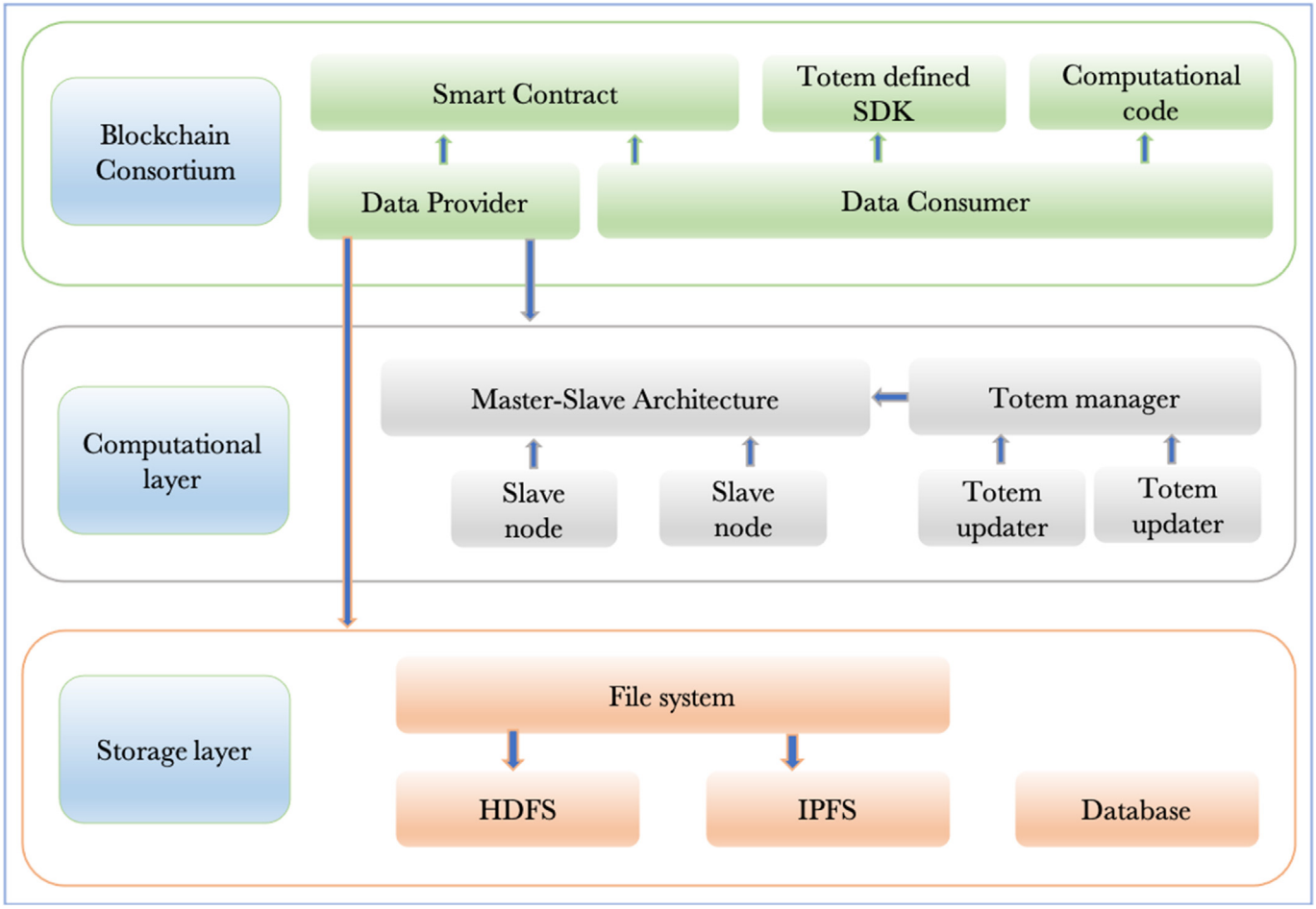

Fig. 1. Three layers of the TOTEM architecture HDFS: Hadoop Distributed File System, IPFS: InterPlanetary File System..

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 data set 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 data set. 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 be 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. Fig. 2 given below shows the workflow of this scenario.

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 communities 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 are 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 analyze the data set available in both C1DH and C2DH, then the S2 will be given to those Data Hubs. Both Data Hubs 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. Fig. 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, an extended version of the TOTEM architecture shows how to deal with multiple data providers in the TOTEM architecture in the following section.

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 architecture network by the data consumer R. Our goal is to retrieve the

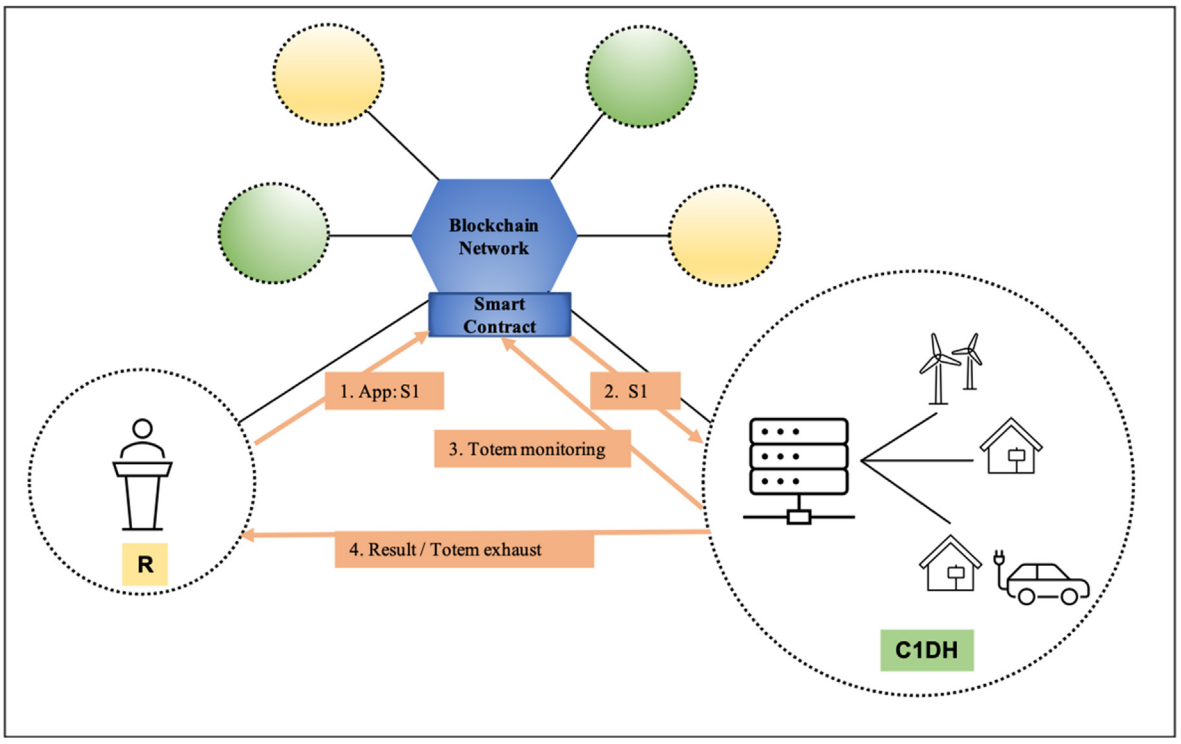

Fig. 2. Scenario 1: With a single data provider.

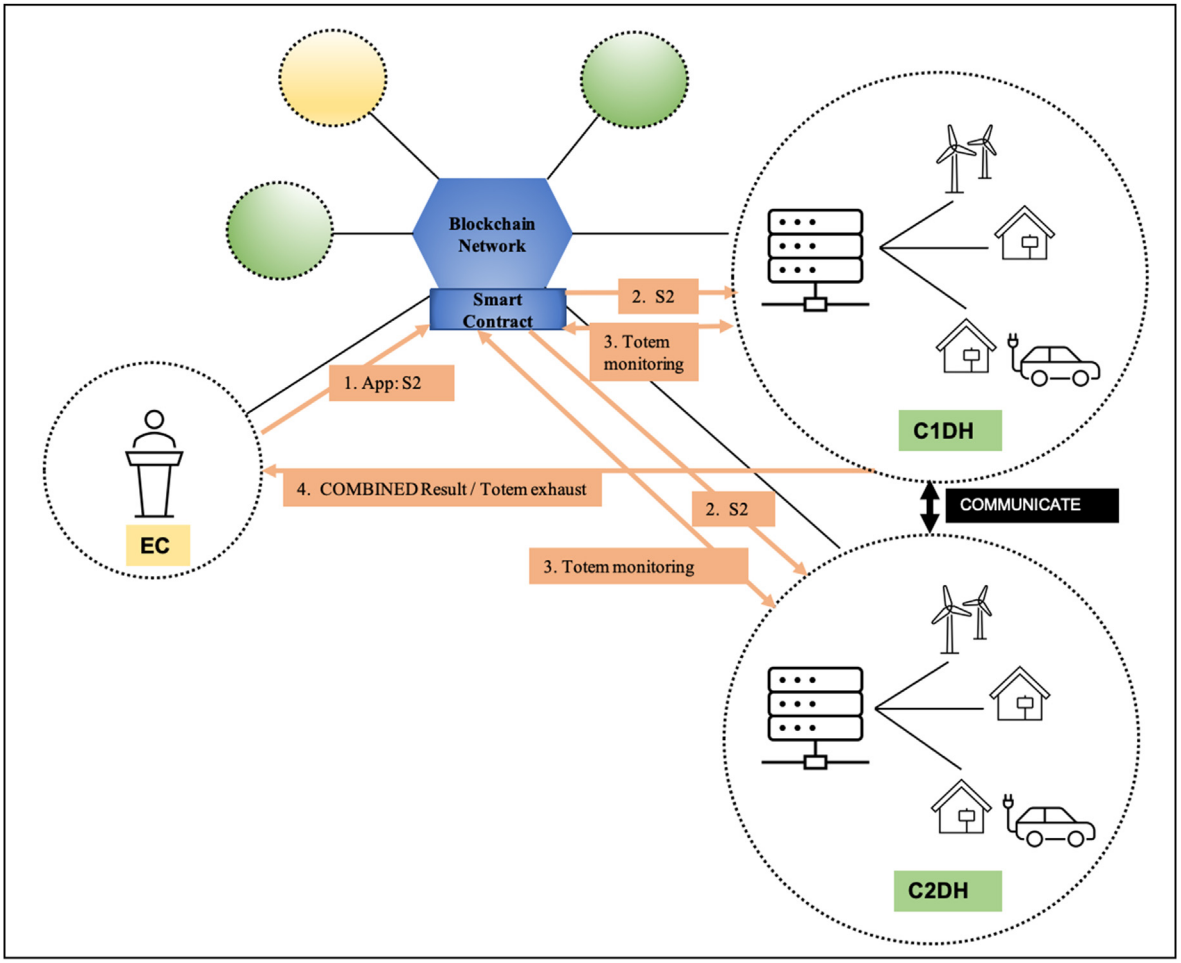

Fig. 3. Scenario 2: With Multiple data provider.

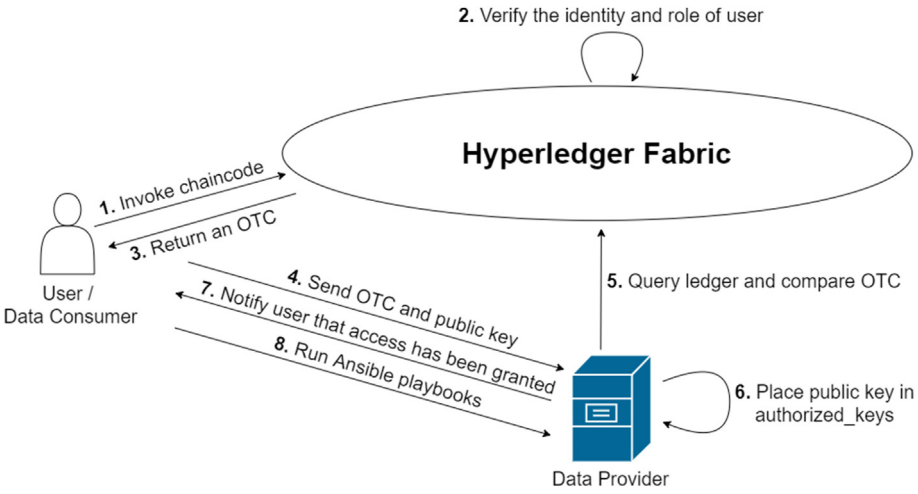

Fig. 4. Architecture for governing remote resource access. OTC: one-time code.

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 storing 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[4] 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 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 that 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 Fig. 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. 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 Fig. 5.

In Fig. 5, we observe 1) two data providers who put the results in their respective private data collections 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 word count job, meaning the function will transform the results to JSON objects, sum values with matching keys, and add key-value pairs that 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 the Microsoft Azure cloud using both a single Kubernetes cluster and a distributed multicluster environment.

6. Implementation and result

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 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[5] 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 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 multicluster distributed Hyperledger Fabric network.

After basic configurations, we specify the size of our node pool. The node pool will contain the nodes that will host the Hyperledger Fabric

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

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

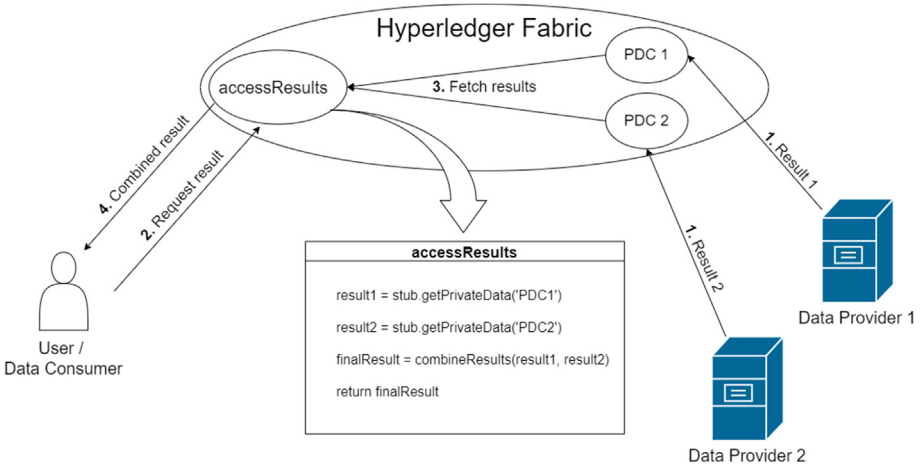

Fig. 5. Scenario 2: Access result with multiple data providers. PDC: private data collection.

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 preinstalled, 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[6]. PIVT provides Helm charts to facilitate launching a Fabric network as well as interacting with it. Helm[7] is a package manager for Kubernetes, which manages charts. The charts provided 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 external services.

In Fig. 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 use PIVT to deploy our Hyperledger Fabric network on two AKS clusters, residing in different regions.

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 Table 1, 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[8]. Following this example, we first create two separate PIVT projects for each cluster by simply copying the

[6] https://github.com/hyfen-nl/PIVT.
[7] https://helm.sh/.
[8] https://github.com/APGGroeiFabriek/PIVT#cross-cluster-raft-network.

```
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
```

Fig. 6. External IP addresses from our services.

Table 1
Overview of clusters in a cross-cluster environment.

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

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. Conclusion

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 access to a remote resource and to the provisional computational resources as Docker containers that form the Hadoop cluster 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 are 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 will consider this into account and will execute the necessary steps. For example, a relevant use of our system would help train machine learning models across several remote data sets. This would require a different approach for combining results or private data sets. 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 problem 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.

Declaration of interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Funding

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] H. Halpin, M. Piekarska, Introduction to security and privacy on the blockchain, 2017 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW); 26–28 Apr 2017; Paris, France, IEEE, Piscataway, NJ, USA, 2017, pp. 1–3.
[2] M. Wu, K. Wang, X. Cai, et al, A comprehensive survey of blockchain: from theory to IoT applications and beyond, IEEE Internet Things J. 6 (5) (2019) 8114–8154.
[3] D.T. Jose, A. Chakravorty, C. Rong, TOTEM: token for controlled computation: integrating blockchain with big data, 2019 10th International Conference on Computing, Commu- Nication and Networking Technologies (ICCCNT); 6–8 Jul 2019; Kanpur, India, IEEE, Piscataway, NJ, USA, 2019, pp. 1–7.
[4] D.T. Jose, A. Chakravorty, C. Rong, Distributed computational framework in TOTEM architecture enabled by blockchain, 2020 15th International Conference on Computer Science and Education (ICCSE); 18–22 Aug 2020; Delft, the Netherlands, IEEE, Piscataway, NJ, USA, 2020, pp. 83–88.
[5] A.M. Pirbazari, A. Chakravorty, C. Rong, Evaluating feature selection methods for short-term load forecasting, 2019 IEEE International Conference on Big Data and Smart Computing (BigComp); 27 Feb–2 Mar 2019; Kyoto, Japan, IEEE, Piscataway, NJ, USA, 2019, pp. 1–8.
[6] A.M. Pirbazari, M. Farmanbar, A. Chakravorty, et al, Short-term load forecasting using smart meter data: a generalization analysis, Processes 8 (4) (2020) 484.
[7] M. Chen, S. Mao, Y. Liu, Big data: a survey, Mobile Network. Appl. 19 (2) (2014) 171–209.
[8] J.S. Hurwitz, A. Nugent, F. Halper, et al. Big Data for Dummies, 1st ed., John Wiley and Sons, Hoboken, NJ, USA, 2013.
[9] P. Chandarana, M. Vijayalakshmi, Big data analytics frameworks, 2014 International Conference on Circuits, Systems, Communication and Information Technology Applications (CSCITA); 4–5 Apr 2014; Mumbai, India, IEEE, Piscataway, NJ, USA, 2014, pp. 430–434.
[10] J. Dean, S. Ghemawat, MapReduce: a flexible data processing tool, Commun. ACM 53 (1) (2010) 72–77.
[11] D. Borthakur, The Hadoop Distributed File System: Architecture and Design, The Apache Software Foundation, 2007.
[12] V.K. Vavilapalli, A.C. Murthy, C. Douglas, et al, Apache hadoop yarn: yet another resource negotiator, Proceedings of the 4th Annual Symposium on Cloud Computing; 1–3 Oct 2013; Santa Clara, CA, USA, ACM, New York, NY, USA, 2013, pp. 1–16.
[13] M. Iansiti, K.R. Lakhani, The truth about blockchain, Harv. Bus. Rev. 95 (1) (2017) 118–127.
[14] S. Haber, W.S. Stornetta, How to time-stamp a digital document, J. Cryptol. 3 (2) (1991) 99–111.
[15] S. Nakamoto, Bitcoin: a peer-to-peer electronic cash system, 2009. Available online: http://www.bitcoin.org/bitcoin.pdf. (Accessed 2 November 2021).
[16] G.W. Peters, E. Panayi, Understanding modern banking ledgers through blockchain technologies: future of transaction processing and smart contracts on the internet of money, in: P. Tasca, T. Aste, L. Pelizzon (Eds.), Banking Beyond Banks and Money, Springer, Cham, Switzerland, 2016, pp. 239–278.
[17] G.-T. Nguyen, K. Kim, A survey about consensus algorithms used in blockchain, J. Inf. Storage Process. Syst. 14 (1) (2018) 101–128.
[18] N. Szabo, Smart Contracts, 1994. Available online: https://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html. (Accessed 2 November 2021).
[19] Vitalik Buterin, A next-generation smart contract and decentralized application platform, white paper 3 (37) (2014).
[20] T. Mitchell. Machine Learning, 1 st, McGraw Hill, New York, NY, USA, 1997.
[21] R.N. Anderson, A. Boulanger, W.B. Powell, W. Scott, Adaptive stochastic control for the smart grid, Proc. IEEE 99 (6) (2011) 1098–1115.
[22] C. Rudin, D. Waltz, R.N. Anderson, et al, Machine learning for the New York City power grid, IEEE Trans. Pattern Anal. Mach. Intell. 34 (2) (2012) 328–345.
[23] Z.M. Fadlullah, M.M. Fouda, N. Kato, et al, An early warning system against malicious activities for smart grid communications, IEEE Netw 25 (5) (2011) 50–55.
[24] Y. Zhang, L. Wang, W. Sun, R.C. Green, M. Alam, Distributed intrusion detection system in a multi-layer network architecture of smart grids, IEEE Trans. Smart Grid 2 (4) (2011) 796–808.
[25] M. Ozay, I. Esnaola, F.T.Y. Vural, et al, Machine learning methods for attack detection in the smart grid, IEEE Transact. Neural Networks Learn. Syst. 27 (8) (2015) 1773–1786.
[26] M. Ozay, I. Esnaola, F.T.Y. Vural, et al, Sparse attack construction and state estimation in the smart grid: centralized and distributed models, IEEE J. Sel. Area. Commun. 31 (7) (2013) 1306–1318.
[27] D.T. Jose, A. Chakravorty, C. Rong, Method for analyzing data using a blockchain, a data provider and a data customer therefor, 2021. US Patent 11121874. 2021.
[28] E. Karafiloski, A. Mishev, Blockchain solutions for big data challenges: a literature review, IEEE EUROCON 2017- 17th International Conference on Smart Technologies; 6–8 Jul 2017; Ohrid, Macedonia, IEEE, Piscataway, NJ, USA, 2017, pp. 763–768.
[29] H. Es-Samaali, A. Outchakoucht, J.P. Leroy, A blockchain-based access control for big data, Int. J. Comput. Networks Commun. Secur. 5 (7) (2017) 137–147.
[30] U.U. Uchibeke, S.H. Kassani, K.A. Schneider, R. Deters, Blockchain access control ecosystem for big data security. 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData); 30 Jul–3 Aug 2018; Halifax, NS, Canada, IEEE, Piscataway, NJ, USA, 2018, pp. 1373–1378.
[31] M.S. Sahoo, P.K. Baruah, HBasechainDB–A scalable blockchain framework on hadoop ecosystem, in: R. Yokota, W. Wu (Eds.), Asian Conference on Supercomputing Frontiers, Springer, Cham, Switzerland, 2018, pp. 18–29.
[32] E. Bandara, W.K. Ng, K. De Zoysa, et al, Mystiko—blockchain meets big data, 2018 IEEE International Conference on Big Data (Big Data); 10–13 Dec 2018; Seattle, WA, USA, IEEE, Piscataway, NJ, USA, 2018, pp. 3024–3032.
[33] K. Yue, Y. Zhang, Y. Chen, Y. Li, L. Zhao, C. Rong, L. Chen, A survey of decentralizing applications via blockchain: the 5G and beyond perspective, IEEE Commun. Surv. Tutorials 23 (4) (2021) 2191–2217.