# PolyChain: a Generic Blockchain as a Service Platform

Shan Jiang, Jiannong Cao, Juncen Zhu, and Yinfeng Cao

Department of Computing, The Hong Kong Polytechnic University
{cssjiang, csjcao, csjzhu1, csyfcao}@comp.polyu.edu.hk

**Abstract.** In recent years, blockchain technology has been attracting intensive attention from both the industries and academia because of its capability of rebuilding trust in trustless environments. There are increasing demands for developing and delivering blockchain applications and services in an agile and continuous way. To this end, Blockchain as a Service (BaaS) emerges which refers to cloud-based blockchain infrastructure developed by a vendor allowing users to develop, host, and use their own blockchain components, functions, and applications. There are many BaaS platforms developed by industries and academia, e.g., Bitcoin, Ethereum, and Hyperledger Fabric. However, they are either limited in scalability or difficult for configuration and customization. In this paper, we propose and develop PolyChain, a generic BaaS platform with high modularity, flexibility, scalability, reliability, and security, which are achieved with the following three design principles. First, each blockchain node is designed as four modularized components, e.g., network, storage, consensus, and application, based on the functionalities. Second, the components in a logic blockchain node interact via communication interfaces and can be deployed on different physical nodes. Finally, the component deployment is optimized based on the capabilities of the physical nodes. We believe PolyChain may benefit the industries and academia in agile development and continuous delivery of blockchain prototypes and applications.

**Keywords:** Blockchain · Blockchain as a Service · Blockchain Platform · Blockchain Architecture · Blockchain Applications

## 1 Introduction

Blockchain is a technology of distributed ledger for trustless data management with auditability. In 2008, Nakamoto developed Bitcoin [10], which is a kind of cryptocurrency and also the first application of blockchain. Later on, the academic and industries have developed a wide range of applications based on blockchain leveraging its distinctive features of decentralization, transparency, and immutability. The application domains range from cryptocurrencies [14], education [13], healthcare [7], to manufacturing [6], etc.

The development of blockchain technology mainly experienced three stages: 1.0, 2.0, and 3.0+. blockchain 1.0 is a programmable currency, e.g., Bitcoin [10],

related to money transfer, remittance, and digital payment. With the integration of smart contracts in blockchain systems, the blockchain technology enters the 2.0 era, in which the representative application is Ethereum [14]. Since the 2.0 era, blockchain has received unprecedented attention all around the world, and enterprises began to develop blockchain-based applications. As a result, enterprise-customized blockchain solutions (blockchain 3.0+) began to appear, in which Hyperledger Fabric [1] is an outstanding project.

With the popularity of blockchain technology, there are increasing demands for developing and delivering blockchain applications and services agilely and continuously. Moreover, many start-ups are eager to promote blockchain-related entrepreneurship even without enough hardware. The technology push and market pull give birth to the concept of blockchain as a service (BaaS), which refers to cloud-based blockchain infrastructure developed by a vendor allowing users to develop, host, and use their blockchain components, functions, and applications.

At present, many enterprises have launched the BaaS platforms, among which Bitcoin, Ethereum, and Hyperledger Fabric are the most popular ones. However, they are either limited in scalability or difficult for configuration and customization. In academia, the researchers have proposed several BaaS solutions, however, they are either conceptual [11][12], for specific applications [2][3][8][4], not deployed in real-world applications [9][17]. The evolution of blockchain demands a generic BaaS platform while existing platforms can hardly provide.

In this paper, we propose PolyChain, a generic BaaS platform with distinctive advantages of flexibility, scalability, reliability, security, and modularity. The main contributions of this paper are as follows:

- We design PolyChain, a generic BaaS platform that may benefit the industries and academia in agile development and continuous delivery of blockchain prototypes and applications.
- We evaluate PolyChain extensively in terms of flexibility, scalability, reliability, security, and modularity.
- We deploy PolyChain with three example applications in authenticable transcripts, big data sharing, and food traceability.

The rest of this paper is organized as follows. In Sec. 2, we introduce the related work and articulate the motivations of this work. In Sec. 3, we introduce the five design goals and three design principles of PolyChain. Sec. 4 presents the system architecture and component design of PolyChain. Three example applications based on PolyChain are demonstrated in Sec. 5. Finally, Sec. 6 concludes the paper.

## 2 Related Work

In this section, we introduce the existing BaaS platforms in industry and academia separately and analyze their shortcomings.

In industry, Bitcoin [10], Ethereum [14], and Hyperledger Fabric [1] are the three representative BaaS platforms in blockchain 1.0, 2.0, and 3.0+, respectively. Bitcoin, a cryptocurrency invented in 2008, is the first application of

blockchain. Although Bitcoin is a great success whose market capitalization hit 1 trillion US dollars in 2021, it suffers from the issues of poor scalability, no support for smart contracts, and difficulties in customization. As a result, the enterprises seldom employ Bitcoin as the BaaS platform.

In 2014, Ethereum was initiated with the support of smart contracts. The developers can freely develop decentralized applications by writing smart contracts on Ethereum. Microsoft announced in November 2015 that it provides Ethereum Blockchain as a Service (EBaaS) on Microsoft Azure. EBaaS allows financial service customers and partners to quickly build and test their applications at a low cost in a ready-made development/test/production environment. It allows users to use industry-leading frameworks to quickly create private, public, and consortium-based blockchain environments and distribute their blockchain products through Azure's World Wide distributed platform. This makes Azure an excellent development/test/production environment for blockchain applications. However, they still have shortcomings, such as insufficient scalability and difficulties to customize the consensus mechanism.

Hyperledger Fabric is a permissioned blockchain infrastructure providing a modular architecture with a delineation of roles among the nodes in the infrastructure, smart contract execution environments, and configurable consensus and membership services. For example, the IBM blockchain network is built on the Hyperledger Fabric stack. Although Hyperledger Fabric is developing very fast, it merely supports public blockchain, provides insufficient native consensus mechanisms, and is very difficult for inexperienced developers to use.

Besides Bitcoin, Ethereum, and Hyperledger Fabric, large enterprises such as Amazon, Baidu, and Alibaba are also developing their BaaS platforms as parts of their cloud services.

In academia, the researchers have been developing BaaS solutions. In [11], Samaniego et al. propose BaaS for the first time. However, it is conceptual and only shows a set of experimental results. Singh et al. analyze the management, governance, and trust issues of BaaS platforms while merely touch the technical details [12]. In [2] and [3], Alia et al. and Aujla et al. consider the integration of BaaS with end-edge-cloud networks and software-defined networks for applications in unmanned aerial vehicles and smart city, receptively, which are pioneer but lacks design details of BaaS.

The two recent research works introduce two BaaS platforms, i.e., NutBaaS [17] and uBaaS [9]. NutBaaS is a BaaS platform providing blockchain service over cloud computing environments, such as network deployment and system monitoring, smart contracts analysis, and testing. Based on these services, developers can focus on the business code to explore how to apply blockchain technology more appropriately to their business scenarios, without bothering to maintain and monitor the system. Although NutBaaS has a clear system architecture with technical approaches to enhance reliability and security, the design of the different layers remains unclear. In uBaaS, various services including deployment as a service, design pattern as a service, and auxiliary services are provided. The deployment of uBaaS is not bound to the cloud service providers or the

blockchain platform. The proposed solutions are evaluated using a real-world quality tracing use case in terms of feasibility and scalability. UBaaS focuses on the vendor-irrelevant design and provides the implementation details. However, the advantages of uBaaS in terms of security, flexibility, etc. over existing BaaS platforms are unclear.

To conclude, the existing BaaS platforms from academia and industry are not enough in terms of modularity, flexibility, scalability, reliability, and security. The evolution of blockchain technology demands a generic BaaS platform.

## 3 PolyChain Design Goals & Principles

In this section, we introduce five primary goals of PolyChain and three design principles to meet the goals.

### 3.1 Design Goals

PolyChain pursues the following five goals:

- *Modularity*: refers to the degree of decomposition of the components. High modularity makes it possible to reuse the developed components, reduces the development costs, and enables transplantation of the components.
- *Flexibility*: refers to the degree of support for diversified customization. Users can flexibly choose system parameters, components, consensus protocols to build a blockchain system that meets their own development requirements.
- *Scalability*: refers to the ability of the system to remain high-performance when the workload increases. It is embodied in the reasonable arrangement of resources and the optimal allocation of hardware.
- *Reliability*: means that the ability of the system to run continuously and stably even with internal faults.
- *Security*: means that the ability of the system to resist various external attacks. In particular, in the BaaS platform, the main resistance is the single point of failure, that is, the broken of one component will not cause the failure of the entire system.

### 3.2 Design Principles

PolyChain employs three design principles, component modularization, distributed deployment, and resource optimization, to achieve the five design goals.

**Component Modularization** In a blockchain system, normally there are many fixed components and layers for different functionalities that make the entire system complex, bloated and hard to develop, thus making it harder to design applications over it. In Ethereum, there are Ethereum virtual machines, miners, blocks, transactions, consensus algorithms, accounts, smart contracts, mining nodes, etc. The developers almost only can design applications through

smart contracts instead of modifying every component they need like consensus. In Hyperledger Fabric, there are also many fixed components and relationships like peer, orderer, endorser, and membership service provider. The design space of developers is also very limited since they need to choose and combine these components to form their network, which means it is hard to change the consensus algorithm or database choices.

The reason why there are many fixed components, and most of them are unmodifiable is that they do not separate various functionalities well within the entire blockchain system, which brings difficulties and costs for developers to design the application they want.

We address the issue by decomposing each blockchain node into four components, namely application component, consensus component, network component, and storage component, based on the functionalities. Each component focus on one dedicated group of functionality. Such an approach achieves modularity and makes the system easy to understand, develop, and maintain. Meanwhile, since the interface of each module is well-defined, their implementation can be changed independently according to different application/experiment need without rebuilding the rest of the system which achieves flexibility. Because the functions are divided into various components, each component does not affect each other, the security of the system is also guaranteed.

**Distributed Deployment** Deployment is another key issue for the current blockchain platform. In Ethereum, for the smart contract, developers need to compile their Solidity code into byte-codes, use APIs for deployment, and finally use the web3 library to call the contracts. Such procedures are unfriendly for those who not familiar with blockchain.

As a BaaS platform, we provide multiple deployment methods through web interfaces. Developers can transform the components they design into an executable program, install script or Docker image then upload and configure them, and finally form a blockchain network. In this way, developers can also Initialize or update their components or nodes in platforms, with monitoring of them. This deployment method ensures the flexibility of the platform.

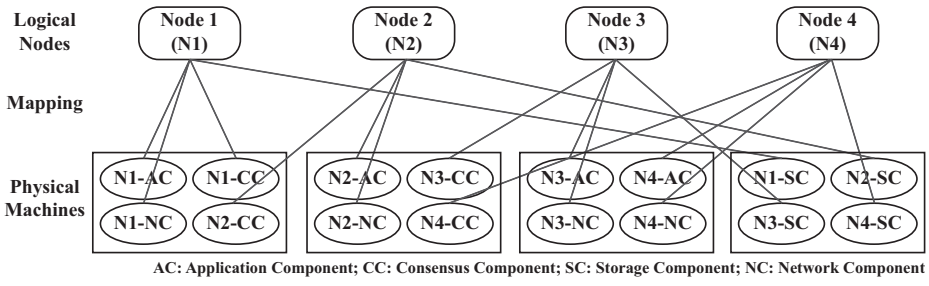


**Fig. 1.** PolyChain Deployment Model

We propose a novel distributed deployment scheme to ensure the reliability and security of the system. First, we identify the concepts of logical nodes and physical machines. Each logical node consists of four components while the components of a logical node can be deployed in different physical machines. As shown in Fig. 1, the application component, network component, and consensus component of logical node 1 are in physical machine 1 while the storage component of logical node 1 is in physical machine 4.

The advantage of the deployment scheme is that it can resist single-point attacks very well. The damage of one component will not affect the operation of the entire blockchain network. Moreover, the physical machines may have different specialized capabilities and are good at hosting different components. For example, storage components can be deployed in physical machines with a high capability of storage.

**Resource Optimization** Finally, we optimize the component deployment based on the functions of physical nodes. We mainly use the following two methods to ensure the scalability of the platform.

Each major component is also a portable distributed sub-system, which means that the components can be implemented in different threads/processes or hosts. Some components, e.g. application component and consensus component, may be implemented in a parallelized manner to maximize transaction processing throughput. In contrast, some components, e.g. storage component, may be implemented in a distributed manner to scale for intensive and complex querying and supporting the huge data volume. The components communicate with each other with fixed protocols and frameworks. Such an approach also enables the scalability and cross-platform capability of the system.

Meanwhile, the components in the platform are reusable. We provide some pre-defined components, e.g., proof-of-work consensus component for developers to use and form their network at the beginning. Developers can also reuse their components built by themselves. This approach will reduce the development cost and make the design of applications more flexible.

## 4 System Architecture and Component Design

In this section, we demonstrate the system architecture and component design of PolyChain in detail.

### 4.1 System Architecture

In PolyChain, there are two kinds of entities, i.e., users and PolyChain nodes. The users generate application data and enjoy the services provided by PolyChain. The PolyChain nodes interact with each other to maintain the blockchain and provide services to the users.

Fig. 2 depicts the system architecture of PolyChain. The blockchain is maintained by a network of nodes connecting with each other. Each PolyChain node consists of four components with functionalities as follows:

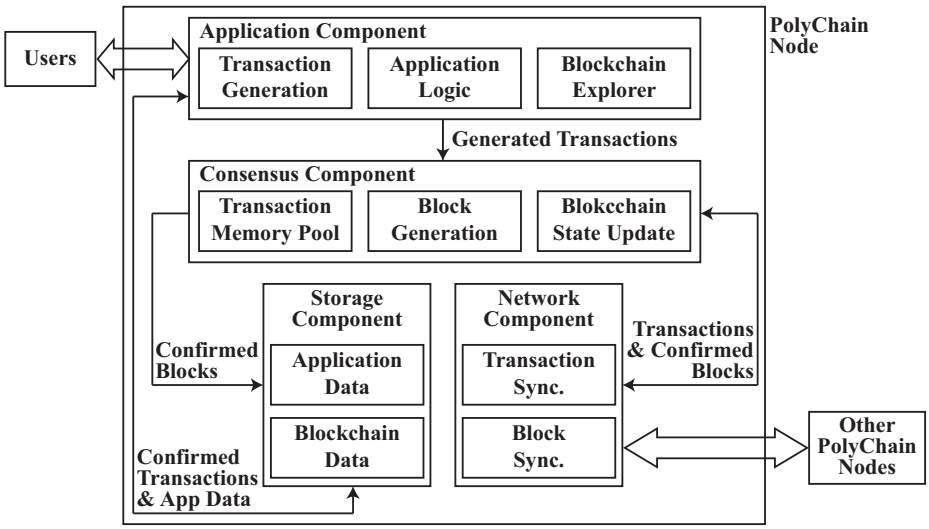


**Fig. 2.** PolyChain System Architecture

- *Application component* (AC) manages the application logic interacting with the users. Particularly, it interacts with the users through predefined application interfaces, generates raw transactions, and applies the confirmed transactions to execute the application logic. Furthermore, it provides interfaces to display the blockchain data, i.e., confirmed blockchain and transactions.
- *Consensus component* (CC) packs the raw transactions into a linked chain of blocks agreed by the PolyChain network. Such a component maintains the pool of raw transactions, packs transactions into blocks, and updates the blockchain state given the blocks from other nodes.
- *Storage component* (SC) manages the application data and blockchain data, which implies the application states and blockchain states, respectively.
- *Network component* (NC) connects the PolyChain node with other ones and synchronizes the transactions and blocks.

The PolyChain system architecture allows the components to be implemented separately. In the following, we explain the design of the four components.

### 4.2 Application Component

As shown in Fig. 3, there are three APIs in the application component: TxGen (invokable by users), OnConfirmedTx (invokable by the storage component), and GetData (invokable by users).

- TxGen takes data from the users as input, generates a transaction based on pre-defined formats, and sends the generated transaction to the consensus component (CC.OnNewTx). In PolyChain, a transaction is a dictionary

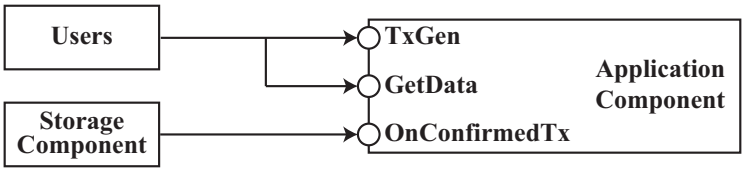


**Fig. 3.** Design of Application Component

with pre-defined fields. The purpose of TxGen is to fill in the fields based on the user data and application logic.
- OnConfirmedTx takes the confirmed transactions from the storage component as input, generates application data by applying the transactions, and submits the application data to the storage component (SC.OnAppData).
- GetData takes parameters from the users as input and responds to the requested data. The requested data can be the application and blockchain data requested from the storage component (SC.GetData, and even the transaction memory pool from the consensus component (CC.GetMemPool) depending on the willingness of the developers.

### 4.3 Consensus Component

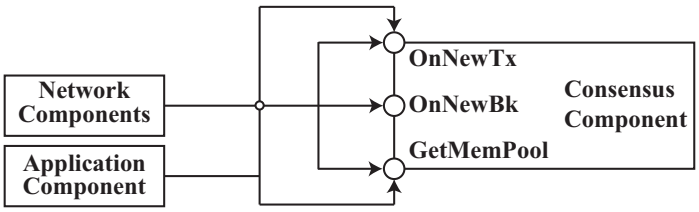


**Fig. 4.** Design of Consensus Component

As shown in Fig. 4, there are three APIs in the consensus component: OnNewTx (invokable by the application and network components), OnNewBk (invokable by the network component), and GetMemPool (invokable by the application and network components).

- OnNewTx takes a new transaction $tx$ as input and updates the transaction memory pool. It checks whether $tx$ is already in the memory pool or the blockchain. If not, it will send $tx$ to the network component (NC.OnNewTx) for broadcasting in the blockchain network.
- OnNewBk takes a new block $bk$ as input and updates the blockchain state. It checks whether the block height is correct and whether the hash values match. If yes, it will store $bk$ in the storage component (SC.OnConfirmedBk)

and send *bk* to the network component (NC.OnNewBk) for broadcasting in the blockchain network.
– GetMemPool responds to the requests with the set of unconfirmed transactions in the memory pool. Such an API is useful for transaction synchronization and state debug.

### 4.4 Storage Component

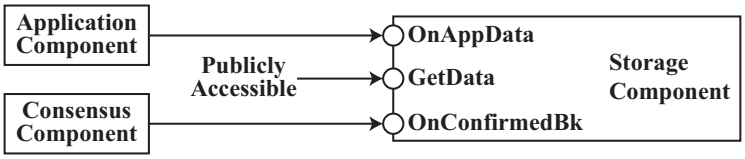


**Fig. 5.** Design of Storage Component

As shown in Fig. 5, there are three APIs in the storage component: OnConfirmedBk (invokable by the consensus component), OnAppData (invokable by the application component), and GetData (publicly accessible).

– OnConfirmedBk takes a block as input, updates the blockchain data, and sends the confirmed transactions in the block to the application component (AC.OnConfirmedTx).
– OnAppData takes the application data as input for storage.
– GetData responds with the corresponding data based on the parameters.

### 4.5 Network Component

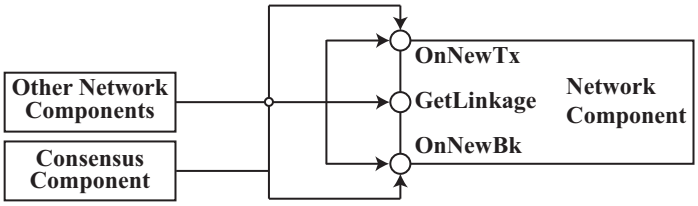


**Fig. 6.** Design of Network Component

As shown in Fig. 6, there are three APIs in the network component: OnNewTx (invokable by the consensus component and other network components), OnNewBk (invokable by the consensus component and other network components), and GetLinkage (invokable by other network components).

– OnNewTx takes a transaction as input and sends it to the consensus component (CC.OnNewTx) and other network components (NC.OnNewTx) for broadcasting purpose.
– OnNewBk takes a block as input and sends it to the consensus component (CC.OnNewBk) and other network components (NC.OnNewBk) for broadcasting purpose.
– GetLinkage responds to the request with the set of connected network components of other blockchain nodes.

## 5 Case Studies

In this section, we demonstrate three case studies, i.e., authenticable transcripts, big data sharing, and food traceability, with the help of PolyChain.

### 5.1 PolyScript: Authenticable University Record Management

In almost all universities, many procedures are complicated and time-consuming because of the numerous data distributed among departments. For example, the confirmation of the examination results may go through the lecturers, departmental general office, faculty general office, research office, and finally the student portal. Blockchain can serve as a secure communication platform connecting various university offices, which can improve the time efficiency of university affairs. Moreover, electronic transcripts and certificates can be generated on the blockchain, which is environmental-friendly, convenient, and anti-counterfeiting. To this end, we propose PolyScript, a PolyChain-based system for authenticable management of university records.

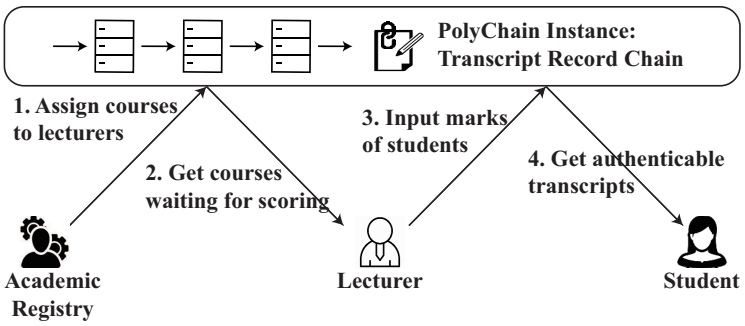


**Fig. 7.** PolyScript System Architecture

Fig. 7 depicts the system architecture of PolyScript. The university (or multiple universities as an alliance) will maintain the PolyChain instance, which is responsible for storing the university records, e.g., student registration information, course information, and examination results.

Each transaction on the PolyChain instance contains the following fields:

- TIME: the timestamp when the piece of record is submitted.
- STUID: the unique identifier of the student.
- COURSEID: the unique identifier of the course.
- LECTURERID: the unique identifier of the lecturer.
- YEARSEM: the academic year and semester of the piece of record.
- GRADE: the examination result.

With PolyChain, a set of university record management functions can be supported as follows:

- *Course assignment* (open to the academic registry): assigning courses to the corresponding lecturers.
- *Marking* (open to the lecturers): input marking results of the students.
- *Transcript generation* (open to the students): generating authenticable transcripts based on the marking results stored on PolyChain.

### 5.2 AI3: PolyChain enabled Big Data Sharing

Big data sharing refers to the act of the data sharers to share big data so that the sharees can find, access, and use it in the agreed ways. In recent years, big data sharing is more and more popular due to its wide applications such as big data trading and cross-domain data analytics. Traditional big data sharing platforms can be classified into data hosting centers and data aggregation centers. However, they suffer from either privacy or authenticity issues. To this end, we cooperate with Huawei Technologies Co., Ltd. to develop AI3, a PolyChain-based big data sharing platform [15][5]. In AI3, we propose to leverage two loosely-coupled blockchains, metadata chain and sharing data chain, to guarantee the privacy of the original data and the authenticity of the sharing records.

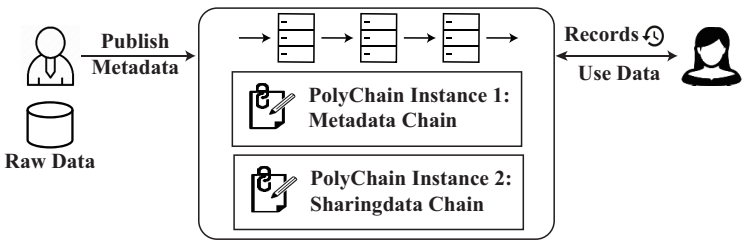


**Fig. 8.** AI3 System Architecture

Fig. 8 depicts the AI3 system architecture. All the data sharers and sharees join the blockchain network and maintain two PolyChain instances, i.e., metadata chain and sharing data chain, which are responsible for storing the shared

metadata and sharing records, respectively. Note that only the metadata will be publicly accessible while the original data is stored by the data sharers locally.

Each transaction on the metadata chain contains fields as follows:

- Time: the timestamp when the metadata is published.
- Publisher: the one who published the metadata.
- URL: a publicly accessible web address to the metadata.
- DataHash: the hash value of the original data.
- FileType: the type of the original file.

Each transaction on the sharing data chain contains the following fields:

- TimeRequest: the timestamp of requesting data.
- timeSharing: the timestamp of sharing data.
- Sharer: the one who share the data.
- Sharee: the one who use the data.
- MetadataHash: the unique identifier of the metadata to be shared (linked with metdata chain).

With the above two PolyChain instances, a set of big data sharing functions can be supported within AI3:

- *Data publishing*: the data sharers publish their data using metadata chain.
- *Data searching*: the data sharees use keywords to search the available data on the metadata chain.
- *Data transfer*: the data sharers transfer the requested data to data sharees with records on sharing data chain.
- *Nearline computation*: the data sharees use predefined functions to perform the calculation on the shared data with permissions from the data sharers. The data sharing records are also stored on sharing data chain.

### 5.3 Food Traceability

Food traceability refers to the ability to follow the trajectories of food products and their ingredients through all steps along the food supply chain. Food traceability helps the stakeholders to better manage the food supply chain and increase of confidence of the customers. Blockchain technology has been widely adopted in food traceability because the transparent and immutable data on blockchain makes the product tracing results remarkably reliable. We cooperate with Alibaba Group Holding Limited to develop a federated blockchain-based solution for ensuring the provenance and authenticity of food items [16].

Fig. 9 depicts the system architecture of the PolyChain-based food traceability system. The stakeholders along the food supply chain will join the blockchain network as federated members and maintain the PolyChain instance, which is responsible for storing the food transactional records and providing query services. There are three kinds of roles, i.e., federated member, administrator, and customer, who have different authorities for querying the blockchain data.

Each transaction on the PolyChain instance contains the following fields:

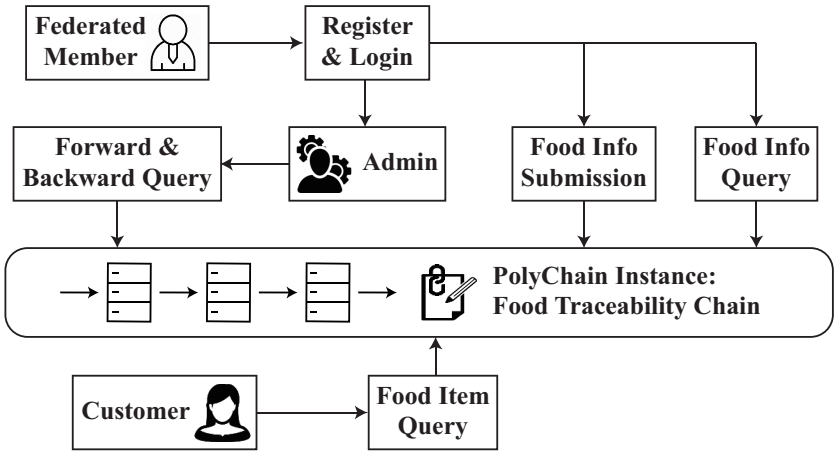


**Fig. 9.** System Architecture of Blockchain Food Traceability

- TIME: the timestamp when the record is submitted.
- LOCATION: the location when the record is submitted.
- PUBLISHER: the one who submitted the record.
- SRCITEMS: the unique identifiers of the source food items.
- DSTITEMS: the unique identifiers of the result food items.
- DESCRIPTION: description information, e.g., access control info, submitted by the publisher.

With PolyChain, a set of food traceability functions can be supported:

- *Food item query* (open to the customers): given the identifier of a food item, query the origin.
- *Food info submission* (open to the federated members): submit the food product records.
- *Food info query* (open to the federated members): given the identifier of a food item, query all the related information along the food supply chain.
- *Forward and backward query* (open to the administrators): given the identifier of an item (it is not necessarily a food item, e.g., can be the ingredients), query all the records about the source and result items.

## 6 Conclusion

In this paper, we propose and develop PolyChain, a BaaS platform with flexibility, scalability, reliability, security, and modularity. We employ the design principles of component modularization, distributed deployment, and resource optimization to provide the above distinctive features. PolyChain is implemented and deployed with applications of authenticable transcripts, big data sharing, and food traceability, which shows its practicability. In the future, we will evaluate the performance of PolyChain extensively, add support of smart contracts, and deploy more real-world applications.

## 7 Acknowledgments

This research is supported by GDSTC Key Technologies R&D Programme with project number 2020B010164002 and Hong Kong RGC Research Impact Fund (RIF) with project number R5034-18.

## References

1. Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., Enyeart, D., Ferris, C., et al.: Hyperledger fabric: a distributed operating system for permissioned blockchains. In: ACM EuroSys. pp. 1–15 (2018)
2. Asheralieva, A., Niyato, D.: Distributed dynamic resource management and pricing in the iot systems with blockchain-as-a-service and uav-enabled mobile edge computing. IEEE Internet of Things Journal **7**(3), 1974–1993 (2019)
3. Aujla, G.S., Singh, M., Bose, A., Kumar, N., Han, G., Buyya, R.: Blocksdn: Blockchain-as-a-service for software defined networking in smart city applications. IEEE Network **34**(2), 83–91 (2020)
4. Chen, Y., Gu, J., Chen, S., Huang, S., Wang, X.S.: A full-spectrum blockchain-as-a-service for business collaboration. In: IEEE ICWS. pp. 219–223 (2019)
5. Jiang, S., Cao, J., McCann, J.A., Yang, Y., Liu, Y., Wang, X., Deng, Y.: Privacy-preserving and efficient multi-keyword search over encrypted data on blockchain. In: IEEE Blockchain. pp. 405–410 (2019)
6. Jiang, S., Cao, J., Wu, H., Yang, Y.: Fairness-based packing of industrial iot data in permissioned blockchains. IEEE Transactions on Industrial Informatics (2020)
7. Jiang, S., Cao, J., Wu, H., Yang, Y., Ma, M., He, J.: Blochie: a blockchain-based platform for healthcare information exchange. In: IEEE SMARTCOMP. pp. 49–56 (2018)
8. Li, D., Deng, L., Cai, Z., Souri, A.: Blockchain as a service models in the internet of things management: Systematic review. Transactions on Emerging Telecommunications Technologies p. e4139 (2020)
9. Lu, Q., Xu, X., Liu, Y., Weber, I., Zhu, L., Zhang, W.: ubaas: A unified blockchain as a service platform. Future Generation Computer Systems **101**, 564–575 (2019)
10. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. Tech. rep. (2019)
11. Samaniego, M., Jamsrandorj, U., Deters, R.: Blockchain as a service for iot. In: IEEE iThings/GreenCom/CPSCom/SmartData. pp. 433–436 (2016)
12. Singh, J., Michels, J.D.: Blockchain as a service (baas): Providers and trust. In: IEEE EuroSP Workshops. pp. 67–74 (2018)
13. Turkanovic, M., Holbl, M., Kosic, K., Hericko, M., Kamisalic, A.: Eductx: A blockchain-based higher education credit platform. IEEE Access **6**, 5112–5127 (2018)
14. Wood, G., et al.: Ethereum: A secure decentralised generalised transaction ledger. Ethereum Project Yellow Paper **151**(2014), 1–32 (2014)
15. Wu, H., Cao, J., Jiang, S., Yang, R., Yang, Y., Hey, J.: Tsar: a fully-distributed trustless data sharing platform. In: IEEE SMARTCOMP. pp. 350–355 (2018)
16. Wu, H., Cao, J., Yang, Y., Tung, C.L., Jiang, S., Tang, B., Liu, Y., Wang, X., Deng, Y.: Data management in supply chain using blockchain: Challenges and a case study. In: IEEE ICCCN. pp. 1–8 (2019)
17. Zheng, W., Zheng, Z., Chen, X., Dai, K., Li, P., Chen, R.: Nutbaas: A blockchain-as-a-service platform. IEEE Access **7**, 134422–134433 (2019)