IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 11, 2019, accepted September 10, 2019, date of publication September 17, 2019,
date of current version September 30, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2941905

NutBaaS: A Blockchain-as-a-Service Platform

WEILIN ZHENG[1,2], ZIBIN ZHENG[1,2], XIANGPING CHEN[1,2], KEMIAN DAI[3], PEISHAN LI[3], AND RENFEI CHEN[3]

[1]School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510006, China
[2]Guangdong Key Laboratory for Big Data Analysis and Simulation of Public Opinion, Sun Yat-sen University, Guangzhou 510006, China
[3]Department of Blockchain Development, Badou Financial Blockchain Technology Company Ltd., Guangzhou 510600, China

Corresponding author: Zibin Zheng (zhzibin@mail.sysu.edu.cn)

This work was supported in part by the National Key Research and Development Program under Grant 2016YFB1000101, in part by the National Natural Science Foundation of China under Grant 61722214 and Grant U1811462, in part by the Guangdong Province Universities and Colleges Pearl River Scholar Funded Scheme, in 2016, and in part by the Pearl River S&T Nova Program of Guangzhou under Grant 201710010046.

ABSTRACT Blockchain, originated from Bitcoin system, has drawn intense attention from the academic community because of its decentralization, persistency, anonymity and auditability. In the past decade, the blockchain technology has evolved and became viable for various applications beyond the domain of finance. However, due to the complexity of blockchain technology, it is usually difficult and costly for most developers or teams to build, maintain and monitor a blockchain network that supports their applications. Most common developers or teams are unable to ensure the reliability and security of the blockchain system, which to a certain extent affects the quality of their applications. In this paper, we develop a BaaS platform called NutBaaS, which provides 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.

INDEX TERMS Blockchain, blockchain-as-a-service, cloud computing, smart contracts.

I. INTRODUCTION

Blockchain technology, known as the foundation of Bitcoin [1], has been used in various fields with its rapid development, resulting in the dawn of a new economy [2]. Recently, a wide range of blockchain-based applications and services have emerged. However, most developers still lack a convenient and effective way to deploy, maintain and monitor their applications, and thus they cannot ensure the reliability and security of the applications. There are many reasons for this, but the most important one is the complexity of the blockchain technology itself. When developers are designing business code, they are unaware of the impact from the complex underlying system, so they cannot take precautions to deal with future errors. Besides, due to a lack of professional knowledge, common developers or teams often fail to monitor the running condition of their systems and cannot accordingly identify and take appropriate measures to fix the system errors in time. In order to solve these problems, developers often need to devote a lot of energy to learning the underlying technology of the blockchain, rather than focusing on the design of business code. However, the work involving these underlying technologies is difficult for most developers or teams.

For example, as shown in the Figure 1, a Hyperledger Fabric [3] network running Kafka consensus algorithm consists of three organization, one orderer cluster as well as one Kafka cluster which is supported by Zookeeper [4]. Therefore, in order to build this network to support your application, you should learn how to translate this network topology into a file that can be understood by docker-compose (either in JSON or YAML format). Besides, you need to be skilled in docker and docker-compose, which are used to startup the network through the file mentioned above. During the startup process, there is a high probability that a few errors will occur, such as the mismatch between docker-image version and source code version, Kafka service unavailable and so on. Only if you are equipped with all or more of the above knowledge can you deal with these errors. Furthermore, aimed to monitor the network for more details about each stage of transactions, such as the duration of the endorsement and transaction commitment, you need to have an intimate knowledge of the principle of Kafka consensus algorithm [5] and even modify the source code to change its underlying architecture. In addition to the

The associate editor coordinating the review of this manuscript and approving it for publication was Yassine Maleh.

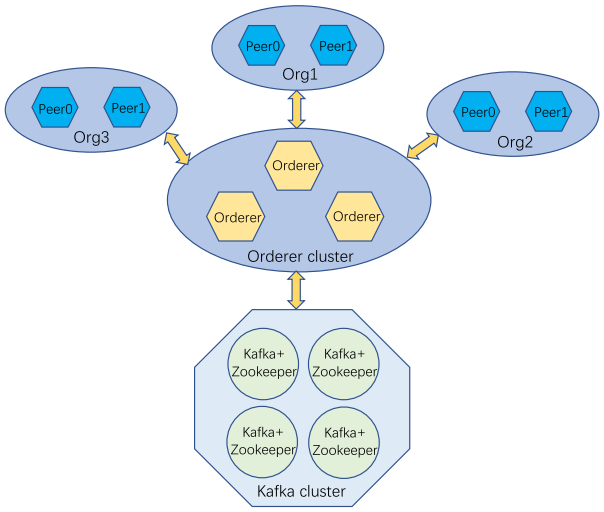


FIGURE 1. A blockchain network based on hyperledger fabric.

above issues, there are still many issues to be considered after successful deployment, such as automatic recovery after system downtime, network environment updating, etc. In short, it is difficult and costly for common developers and teams to deploy, maintain and monitor a blockchain network.

Based on the problems mentioned above, BaaS (Blockchain-as-a-Service) has been proposed in recent years. By embedding the blockchain framework into the cloud computing platform, a BaaS platform can leverage the deployment and management advantages of cloud service infrastructure to provide developers with convenient, high-performance blockchain ecosystems and related services. Through these basic cloud services, developers can quickly startup a blockchain network to support their application ignoring the complexity of the underlying architecture. Besides, some more advanced services such as those related to security and performance, have been gradually introduced into BaaS platform to provide developers with a more complete ecosystem.

Nowadays, many companies, such as IBM and Microsoft, have released their BaaS platforms and achieved good outcomes. Their contribution is mainly to reduce the difficulty of deployment and development, but the services provided by these platforms still have many shortcomings. Especially in system monitoring and support of underlying system types, these platforms have much to be improved. In the paper, we develop a BaaS platform called NutBaaS (which means providing developers with a 'hard' barrier like the nut shell to protect their blockchain applications) to improve the current BaaS platforms, providing a more convenient and safer development environment for developers. Many basic services such as network deployment or advanced services like smart contracts security vulnerability detection are available on the platform through a set of RESTful API. All services are dedicated to enabling developers to focus on business code and leave the rest to NutBaaS.

The remainder of the paper is organized as follows. Section II presents some related works about BaaS platform and our major strengths compared with other platforms, while Section III introduces the background and preliminaries of the work. The structure and the technological innovation of NutBaaS will be highlighted in Section IV and Section V, respectively. Then, some key challenges for the future development of BaaS are discussed in Section VI. Finally, we provide a summary of this paper and some future plans in Section VII.

II. RELATED WORKS AND OUR CONTRIBUTION

Since the concept of BaaS was put forward, many companies in recent years have rushed to release BaaS platforms offering services for blockchain manipulations and building, shipping and running the business logic over blockchain networks. The well-known platforms include but not limited to IBM Blockchain [6], Microsoft Azure Blockchain [7], Ethereum Blockchain as a Service on Azure [8], AWS Blockchain [9] and so on.

In November 2015, Microsoft announced that it would provide EBaaS (Ethereum Blockchain as a Service) services on Azure cloud platform, where developers can quickly create Ethereum blockchain environments in the easiest and most efficient way. The platform was officially opened to the public in August 2016. In February 2016, IBM announced that it would provide blockchain as a service based on Hyperledger Fabric. IBM Blockchain is designed to provide users with an end-to-end blockchain platform solution that quickly builds a highly available blockchain network and provides blockchain security services. In May of the same year, AWS (Amazon Web Services) partnered with the investment company DCG (Digital Currency Group) to provide a blockchain-as-a-service environment for companies invested by DCG. Even though these platforms have brought many benefits to developers and the whole blockchain community, there are still many deficiencies that need to be improved urgently.

In general, every BaaS platform will only choose one blockchain type like Hyperledger Fabric, Ethereum or EOS as a foundation of its underlying architecture. For example, IBM blockchain uses Hyperledger Fabric, while Ethereum Blockchain as a Service on Azure uses Ethereum. However, in various business scenarios, some paying more attention to the token in the network would choose Ethereum while others requiring trading performance would choose Hyperledger. Besides, based on the consideration of file storage, some scenarios may have to choose Filecoin [10], which is considered as a decentralized storage network based on blockchain and IPFS [11]. It can be seen that the existing BaaS platforms have relatively monotonous support for blockchain types. On NutBaaS, we support many types like Hyperledger Fabric, Ethereum, EOS and Filecoin etc. Furthermore, we provide a solution to combine different blockchain types to support complex business scenarios.

In addition to the scalability of the supported blockchain types, we also improve the scalability of smart contracts. Since the Blockchain2.0 era, smart contracts have promoted the development of the whole blockchain ecosystem because

it extends the ability of blockchain to process data. However, the ability of smart contracts to access external data and other chain's data is still a troublesome issue which causes inconvenience to many business scenarios. All the BaaS platforms mentioned above haven't made much improvement in this respect. On NutBaaS, smart contracts are enabled to call external data interfaces through http or https. This gives developers more flexibility in designing smart contracts. Besides, developers can easily write and test the contracts on the platform based on the templates provided.

In system monitoring, most BaaS platforms, such as IBM Blockchain, focus on the overall operation of the peers in the network for achieving rapid recovery after node faults as well as the duration and delay of transactions for a rough statistic of system performance. However, this is not enough for most business scenarios based on Hyperledger Fabric. When there is something wrong in the blockchain system, developers need more details about the peers and the transactions to find out which stage (such as endorsement, commitment and sorting) of the transaction goes wrong and results in system faults. On NutBaaS, we implement a detailed and real-time monitoring framework with a log-based method. It can track service invocation in a transaction and the time-consuming situation of each stage. Based on these statistics, developers can conduct a detailed performance analysis on peers that provide different services in the network and then make a targeted optimization for them.

Apart from what mentioned above, we also introduce some advanced technologies into NutBaaS, such as smart contracts security vulnerability detection and automatic repair. In the future, we hope that through the joint efforts of everyone, we can introduce more advanced technologies to build a safer, more reliable and more convenient BaaS platform.

III. BACKGROUND

A. BLOCKCHAINS

A blockchain can be referred to as a distributed ledger, where the data and transactions are not under the control of any third party. Any transactions are completely recorded in the public ledger in a permanent and verifiable way. The first introduction of blockchain was implemented by Satoshi Nakamoto as a core part of Bitcoin.

With the further development of the technology, there are various blockchains with different goals. For example, Ethereum names its own blockchain as Ethereum Blockchain [12], whereas Hyperledger names its own blockchain as Hyperledger Fabric [3]. However, all of them have some common elements as follows:

- Replicated ledger: All nodes in a blockchain network securely store the history of transactions. The latest transactions are packaged into a block and then the block is append-only with immutable past. All transactions in the blocks are distributed and replicated among all nodes taking part in the network.
- Peer-to-Peer network: All nodes share a public ledger without a centralized control actor over the Internet. In other words, all nodes are connected through a peer-to-peer network. Transactions and blocks are synchronized through this network.
- Consensus: Before the blocks are inserted into the chain, all nodes on the network need to reach a consensus on the validity and the order of transactions within the blocks. The most representative consensus algorithm in public chain is Proof-Of-Work (POW) [1], which is used in Bitcoin System. Other algorithms, like Proof of Stake (POS) [13] and Practical Byzantine Fault Tolerance (PBFT) [14] are used in Ethereum and Hyperledger Fabric respectively.
- Cryptography: The security of blockchain system is based on the knowledge of cryptography. In a blockchain network, the integrity of transactions supports digital signatures and proprietary data structures (e.g.,Merkle tree [15] in Bitcoin, Merkle Patricia Tree [12], [16] in Ethereum). Besides, the authenticity of transactions is supported by digital signatures. The privacy of transactions is supported by asymmetric cryptosystem.

Blockchain technology also has some drawbacks [17], which have attracted our attention. As we know, the blockchain is immutable and append-only, so the storage space of blockchain will continue to grow. For the Bitcoin blockchain, its size reached 219GB on May 18, 2019. At the same time, its distributed storage characteristics have caused a waste of resources. Another crucial issue is transaction throughput of blockchain network, which can be measured by transactions per second. Besides, other issues, such as network congestion, block size, or synchronization mechanism, have been highlighted in many studies. To solve these issues, in the past decade, a lot of researches [18]–[20] have concentrated on the improvement of the four elements mentioned above.

B. CLOUD COMPUTING

Cloud computing is a pay-as-you-go mode of providing scalable, flexible, and shared computing services (e.g., servers, storage, databases, networks, software, analysis, intelligence, etc.) to users over the network. It is attractive to business owners because it has several compelling features: high scalability, high reliability, on-demand services, easy access and extremely low cost.

Nowadays, services offered by clouds can be divided into three categories: software as a service (SaaS), platform as a service (PaaS), and infrastructure as a service (IaaS). IaaS refers to on-demand provisioning of infrastructural resources, while PaaS refers to providing platform layer resources (such as operating system support) and SaaS refers to providing on-demand applications. Examples of different types of services include Amazon EC2 (IaaS) [21], Google App Engine (PaaS) [22], and Rackspace (SaaS) [23]. In recent years, other types of cloud service, such as Function-as-a-Service (FaaS) [24] and Data-as-a-Service (DaaS) [25] have also attracted public attention.

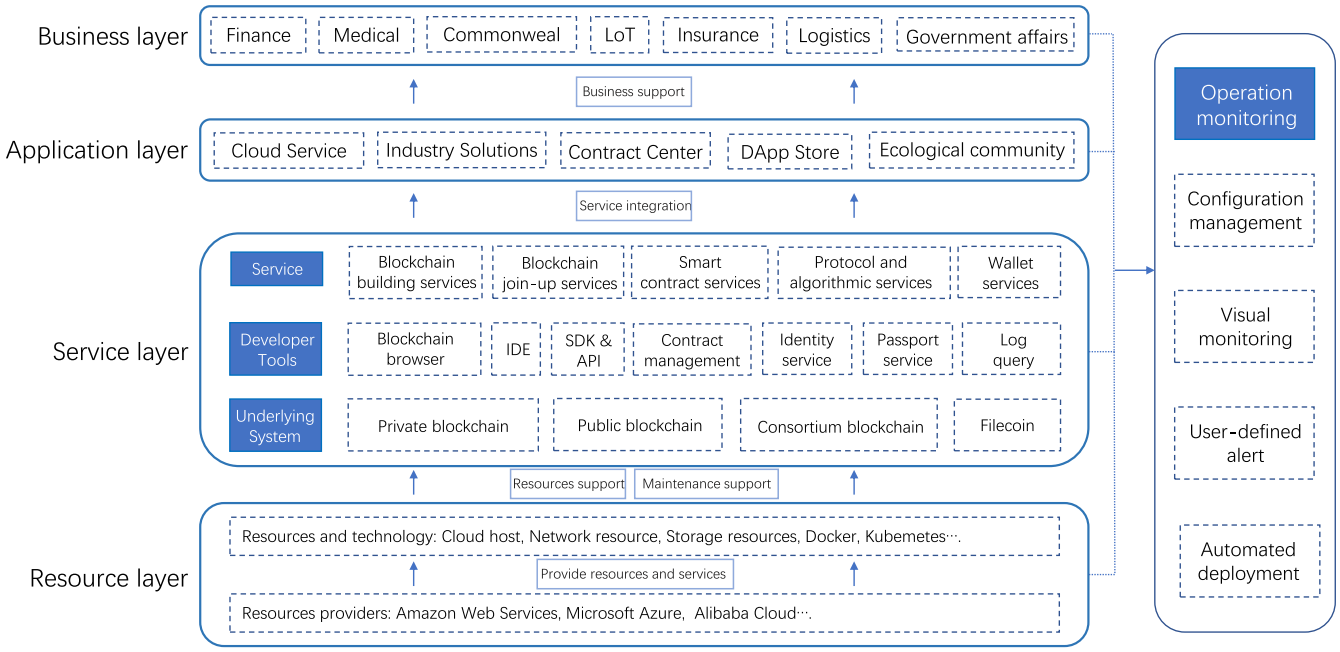


FIGURE 2. The architecture of NutBaaS.

C. BLOCKCHAIN-AS-A-SERVICE (BAAS)

Blockchain-as-a-service (BaaS), the combination of cloud computing and blockchain, is an offering that allows users to leverage cloud-based solutions to build, host and manage their own blockchain apps, smart contracts and functions on the blockchain. The BaaS providers manage all the necessary tasks and activities to keep the infrastructure agile, operational and easily accessible. It is an interesting development in the blockchain ecosystem that is indirectly aiding the blockchain adoption across businesses by helping enterprises simplify operation process and reduce deployment cost. It is based on, and works similar to, the concept of Platform-as-a-Service (PaaS) model.

Clouds can enable the outsourcing of skills and expertise with regard to technology deployment and management. Blockchain is an emerging technology, meaning that experts in the area are limited, and in high demand. As such, BaaS can facilitate technology access, providing abstractions over the lower-level technical details. Currently, a key BaaS marketing focus is on quickly establishing the development environment to support businesses in their desire to explore the blockchain technology's potential. Besides, it can also provide a series of operation services such as search query, transaction submission and data analysis based on blockchain. These services can help developers verify their concepts and models more quickly. The service ability of BaaS platform is embodied in its stronger instrumentality, which facilitates the creation, deployment, operation and monitoring of blockchain. Recently, many large companies have gradually released their BaaS platforms, such as IBM blockchain [6] and Microsoft Azure Blockchain [7], but the services provided by these platforms still have some limitations.

IV. ARCHITECTURE

The architecture of NutBaaS is divided into four layers, which is shown in Figure 2. The lowest layer is the *Resource Layer* which provides the infrastructure (such as storage, databases and networks) needed for blockchain services. The layer above the *Resource Layer* is the *Service Layer*, which is the most important layer on NutBaaS. All blockchain basic services and advanced services are implemented in this layer. By integrating these services, we have constructed some applications (such as DApp Store, Contract center as well as some general industry solutions) at the *Application Layer* that are conducive to the overall ecological development. In the *Application Layer*, people can quickly find solutions to their business scenarios in the *Business Layer*, where some mature solutions and corresponding application examples are shown.

The main goal of the NutBaaS architecture design is to provide a comprehensive and detailed operational monitoring mechanism for the blockchain system, as shown in the right part of Figure 2. The four major layers mentioned above are also working for this goal. The mechanism covers four aspects of blockchain operations: *Configuration management*, *Visual monitoring*, *User-defined alert*, and *Automated deployment*. *Automated deployment* is aimed at providing an integrated service from deploying test networks, writing and testing smart contracts, customizing applications, to experiencing and sharing applications. The key work of *Configuration management* is the maintenance of the network, including the initial configuration of the network, as well as network configuration updates (such as peer upgrades) at runtime. The purpose of *Visual monitoring* and *User-defined alert* is to provide the user with instant visual information, and alarm users according to the alarm threshold set by the

user, so that the user can immediately discover the system problem and solve it. We'll follow up with more details on the four major layers mentioned above.

In the *Resource Layer*, NutBaaS provides various clouds resources and infrastructures for deploying a basic blockchain network. NutBaaS can support private, public and hybrid cloud deployment including Amazon Web Services, Microsoft Azure, Alibaba Cloud etc. In other words, you can deploy different types of peers (e.g., orderer peers, endorsement peers) in different clouds, which can reduce the risk and damage of system collapse when some of the clouds you choose have problems (e.g., Network jitter, system outage). Besides, in the mode of hybrid cloud deployment, it is difficult for service tenants (network participants) to collude with all cloud service providers and unilaterally modify the entire network. Compared to the mode of deploying all peers in the same cloud, hybrid cloud deployment can effectively increase the attack difficulty of malicious attackers. In deploying technology, NutBaaS supports docker-container deployment and other deployment methods. In order to improve the portability and scalability of the system, we also use Kubernetes [26] (an open-source system for automating deployment, scalization, and management of containerized applications) to arrange and manage our containers. Through the services provided by Kubernetes, network participants can quickly update, expand and migrate containers (i.e. peers) or even rearrange the entire network.

The *Service Layer*, the most important layer on NutBaaS, can be divided into three sublayers. The lowest sublayer is the *Underlying System* supporting the current blockchain framework for different types of blockchains. Based on the *Underlying System*, we develop a lot of tools in *Developer Tools*. Then, in *Service sublayer*, we combined these tools to provide some basic and advanced services.

Based on the *Resource Layer*, NutBaaS supports the deployment of multiple underlying systems, including consortium blockchain Hyperledger Fabric 1.0.x to 1.4.x, public or private blockchain Ethereum peers, distributed storage Filecoin and so on. We also enhance the ability of smart contracts to access external data, so different types of chains can access each other's data through smart contracts. Thus, in some business scenarios, users can flexibly combine different types of chains according to their needs of business, making full use of the advantages of different types of chains without worrying about cross-chain data access. On top of the *Underlying System*, we develop a series of tools to facilitate blockchain developer's development including Blockchain browser, IDE, SDK&API, Contract management, Identity service, Passport service and Log query. These tools are designed to simplify the interaction between developers and blockchain. Blockchain browser provides a visual interface for users to interact with the blockchain system. For example, users can see the real-time status (such as transaction throughput, service quality of peers, etc.) of the blockchain system through the browser. IDE and SDK&API provide a set of tools that help developers design smart contracts and deploy blockchain test network. Contract management is responsible for managing the contract on the chain, including contract update, periodic statistics on contract data, and so on. Identity service and Passport service are mainly related to IdentityChain technology (In Section V). Log Query provides an interface that allows users to trace the operational records of the blockchain system.

By combining the functions provided by these tools, we will provide some customized blockchain services to further reduce the uptake barriers of development. In the *Service sublayer*, we provide a series of services which users can easily access through a set of RESTful API. *Blockchain building services* mainly provides one-click deployment and update services of blockchain networks based on user-defined profiles. *Blockchain join-up Services* is dedicated to helping new members quickly join an existing blockchain network. Besides, NutBaaS provides *Smart contract services* to help developers detect security vulnerabilities of smart contracts to avoid economic losses caused by these potential vulnerabilities. In order to promote the development of the DApp ecosystem, NutBaaS also provides *Wallet service* to allow developers to experience and share their blockchain applications. All services in this sublayer are aimed to provide a convenient and safe development environment.

By integrating the underlying blockchain services, we provide some applications in the *Application Layer* to help find suitable solutions faster according to their business scenarios. For example, you can experience all kinds of DApp in the DApp Store, and then know which underlying framework is suitable for your scenario. Besides, you can learn how to design smart contracts in Contract Center by reading some classic examples. In order to create a good learning atmosphere, we also build a community where you can share and discuss some interesting topics with others. On the top of the architecture, the *Business Layer*, our main task is to explore more business scenarios suitable for the use of blockchain technology, and then come up with a specific solution for people to study.

V. TECHNOLOGICAL INNOVATION

A. TRANSACTION BEHAVIOR TRACKING

Performance has been one of the important issues in various blockchain systems and becomes a major constraint of its applications since the emergence of the blockchain. Current blockchain systems, whether public blockchains or consortium blockchains (such as Ethereum and Hyperledger Fabric), suffer from various performance problems, especially when executing complex smart contracts. In the study [27], McCorry et al. introduced an open board voting system which maximizes the privacy of the voters, but the execution time of the contract will become longer as the number of voters grows. In addition, Christidis et al. investigated the application and limitations of smart contracts for IoT, indicating that the efficiency is too low to execute the contracts on IoT [28]. All of these applications have the same problem of long

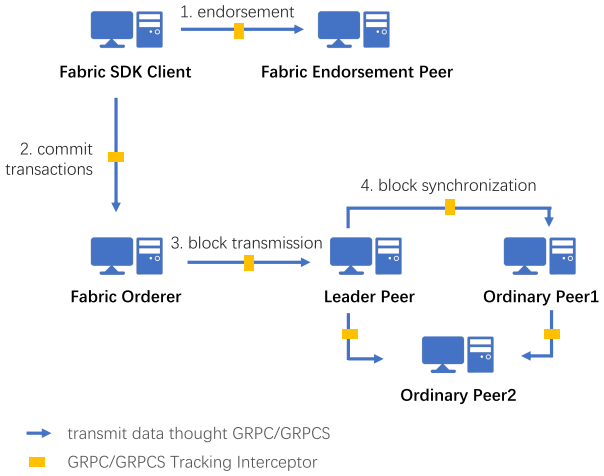


FIGURE 3. Four stages to reach a transaction in hyperledger fabric.

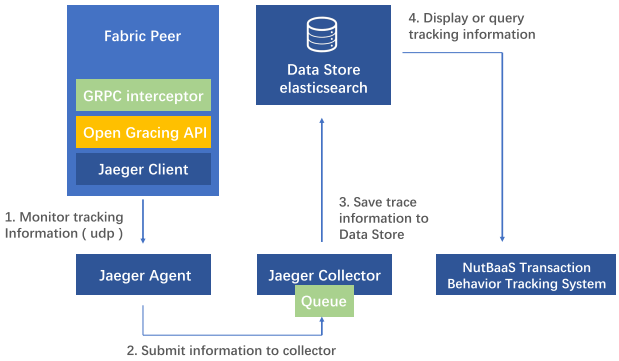


FIGURE 4. Implementation of GRPC/GRPCS tracking interceptor.

execution time and low efficiency, which make them still unable to replace the centralized solution. Performance is one of the key factors restricting the application of blockchain-based smart contacts. Therefore, real-time performance monitoring of a blockchain network is urgently needed, which can help developers find out the mistakes when the system is abnormal as well as optimize the corresponding peers according to the result of the performance analysis.

At present, there are some studies [29], [30] focusing on the overall performance evaluation of blockchain systems. The performance monitoring methods of the current BaaS platform are similar to the methods of these studies. They focus on monitoring the overall transaction situation of the whole network such as transaction throughput, average time to complete a transaction, transaction confirmation delay and so on. However, overall metrics cannot reflect the detailed performance at different stages. Once the network is abnormal, it's hard to pinpoint which stage of the transaction has an impact on the system. In the study [31], Peilin Zheng et al. introduced a Detailed and Real-time Performance Monitoring Framework for different blockchain types, including Ethereum, Parity-PoW, Hyperledger Fabric, CITA and so on. This framework takes different stages such as transaction validation, execution and so on) of a transaction into consideration. In this framework, all peers are treated as the same role in the system, but in Hyperledger Fabric, there are several different roles of peers, which are responsible for different tasks of transaction processing. Besides, as shown in the Figure 3, a transaction in Hyperledger Fabric is divided into several stages, some of which (e.g., endorsement) are not considered in the framework. In general, the framework is not fully applicable to Hyperledger Fabric. It cannot monitor the service invocation and time-consuming situation of a transaction at different stages.

As shown in Figure 3, there are several stages to reach a transaction in a network based on Hyperledger Fabric. The first stage is endorsement according to endorsement strategy and the second one is committing the transactions to Fabric Orderer. After sorting the transactions, Fabric Orderer will package all transactions into a block and transmit it to Leader Peer. Finally, the Leader Peer will synchronize this block to other peers in the same channel. In order to monitor the service invocation and time-consuming situation of each stage, we develop a GRPC/GRPCS Tracking Interceptor based on Jaeger [32]. As shown in Figure 3, transactions and blocks are transmitted between different peers though GRPC/GRPCS, so we install a tracking interceptor between GRPC server and GRPC client. When the endorsement peers, the orderers and the ordinary peers are interacting with GRPC, the tracking interceptor can collect the context information for service invocation as well as transaction information. Finally, all information is transmitted to our system as a source of data for our analysis of system performance or exceptions.

There are four collecting points for tracking information from generation to entry into NutBaaS system, which is shown in Figure 4:

- Jaeger Client: An OpenTracing-compliant SDK is implemented for different languages. Through the API provided by the SDK, the GRPC interceptor writes data to Jaeger Client and then the client passes the tracing information to the Jaeger Agent according to the sampling strategy specified by the application.
- Jaeger Agent: It is a network daemon that listens to receive span data on a UDP port, and it sends data to the Jaeger collector in batch.
- Jaeger Collector: It receives data from Jaeger Agent and then writes the data to the backend storage.
- Data Store: Currently, we use Elasticsearch [33], a distributed, RESTful search and analytics engine capable of solving a growing number of use cases, to store log information. Users can query real-time log information easily through a set of RESTful API.

After collecting the context information, we can track the service invocation at each stage and the time spent on each service in a complete transaction. By comparing the service invocation and time-consuming situation at different stages of the transaction, we can quickly find out which service is the main cause of the system exception when the system is abnormal. Then, we can optimize the peers that provide

this service. In this paper, we optimize the monitoring facility of blockchain systems based on Hyperledger Fabric by introducing a GRPC/GRPCS Tracking Interceptor. At the same time, we introduce this interceptor into the framework in the study [31] and integrate this framework into NutBaaS to provide comprehensive and detailed monitoring services for different blockchain types.

B. IDENTITY-CHAIN TECHNOLOGY

For most business scenarios, the account is the basic configuration of each product, which is no exception in the blockchain system. Whether in the public or consortium chain, a secure account system that supports smart contracts transactions is required. For example, in Ethereum, each common user has an account and each account has a public key and a private key. The security and confidentiality of transactions initiated by the account are protected by this pair of secret keys, i.e., by a symmetric cryptosystem. In addition, a secure account system can withstand many malicious attacks, such as double-spend attack. In the Ethereum account model, the account's nonce field is added to each transaction initiated by the account, so that only one of the transactions with the same nonce value initiated by the account is recognized by the network finally. Therefore, Ethereum can effectively withstand double-spend attack.

However, in the current consortium chain frameworks (such as Hyperledger Fabric), the account system is not complete. In Hyperledger Fabric, the account in Fabric is actually a set of certificates and key files generated according to the PKI specification, and is generally only owned by a member (or a peer) in the organization. Usually the member represents an institution (such as an enterprise), but for common users of the institution, they do not have an account. They can only use the services provided by the institution. Therefore, the security of transactions for all common users depends on the confidentiality of member's certificates and secret key files. Once a member's certificates or keys are leaked, the security of the common user in the member will be threatened. As shown in Figure 5, we provide Identity-Chain technology to solve this problem, allowing enterprises to choose their own encryption algorithm as the public and private key generation algorithm of accounts and build their own identity chain according to their business needs. Thus, enterprises can give their users a secure account (identity) to protect the security and privacy of their transactions.

Firstly, the enterprise needs to customize their identity chain profile based on the templates provided by NutBaaS, such as identity-chain account address prefix, identity-chain account permissions as well as what basic information users need to register. Next, the enterprise uploads the profile to the NutBaaS platform, and the platform will check the validity of the profile. Once the profile's check is passed, the platform transmits the profile to the deployment factory and starts deployment. After successful deploying, the customized identity chain for the enterprise will be generated in which the users of the enterprise can register themselves.

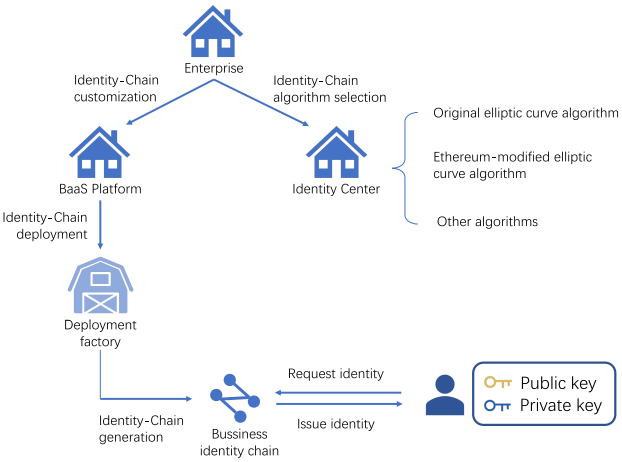


FIGURE 5. The design of the identity chain.

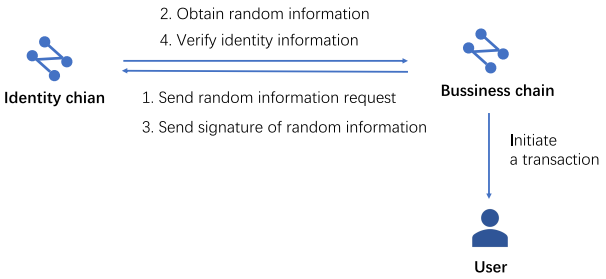


FIGURE 6. The interaction between the identity chain and the business chain.

In order to ensure the security of the chain account, the enterprise can choose the identity chain encryption algorithm (such as Original ECDSA (Elliptic Curve Digital Signature Algorithm) [34] or Ethereum-modified ECDSA [35]) supported by the Identity Center as the public and private key generation algorithm of the account.

Now, in this mode, the user can only successfully submit the transaction to the business chain when authorized by the identity chain. Figure 6 shows the interaction between the identity chain and the business chain after the user initiates a transaction.

1) Send random information request: When the Bussiness-Chain peer (a member of the organization) receives a transaction, and if the transaction needs to be authorized, a request is sent to obtain the random information. (If the authorization is forcibly skipped, the transaction fails due to the transaction verification.)
2) Obtain random information: Business-Chain peer sends a request for random information, including the public identity information of the user. After receiving the request, Identity-Chain peer first checks whether the public identity information sent by Business-Chain peer is legal. If it is legal, a random information is generated and returned to the Business-Chain peer. The generated random information and the public identity of the user are saved for a short period of time.

3) Send signature of random information: The Bussiness-Chain peer receives the random information returned by the Identity-Chain peer, then uses the elliptic curve encryption algorithm to sign the random information. Finally, it sends the signature information and the random information to the Identity-Chain peer.
4) Verify identity information: When the Identity-Chain peer receives the information from the Bussiness-Chain peer, it first checks whether the random information is generated by itself in a short period of time. If the check is passed, the random information is used to obtain the public identity information of the user from the identity chain and then use it to verify the signature. If all verifications pass, the transaction will be successfully submitted, otherwise the transaction will be terminated.

Through the Identity-Chain technology described above, a common user can register an account on the identity chain of the enterprise and initiate a transaction on the network as a separate entity. The privacy of the individual users and the security of the transaction are supported by user's private key. Any transaction needs to go through the authorization process shown in Figure 6 to be successfully executed. Identity-Chain technology optimizes the account system of Hyperledger Fabric, allowing the users of the member peers to interact with the blockchain system as independent individuals rather than relying entirely on the services provided by the member peers. In addition, this technology allows enterprises to customize their user account system, including the choice of the encryption algorithm, the definition of account permission and account prefix, etc.

C. EXTERNAL DATA ACCESS CAPABILITY OF SMART CONTRACTS

With the emergence of smart contracts, blockchain technology is no longer only used in digital currencies, but also widely used in finance, medical, the Internet of Things and so on. However, smart contracts have limited ability to process and store data. For example, in Ethereum, due to the limitation of each block's gasLimit, the amount of data that a smart contract can handle in each transaction is limited. In addition, once the smart contract stores too much data, the speed of querying the data will be low. Therefore, when designing a smart contract, developers usually introduce a cache database to store data, and only store some key data on the chain. However, in the design of most business scenarios (whether based on Hyperledger Fabric or Ethereum), the inability of smart contracts to access external data(from cache database or other applications) is a huge inconvenience to developers.

Oraclize [36] is designed to solve this problem. It is a smart contract in Ethereum, so that it can only work for Ethereum. Besides, callers need to pay a fee based on their use of Oraclize. At the same time, Oraclize uses contract-call-contract to access external data, which is not the most familiar development method for developers. There is still a lack of a common and free way for users to access external data within

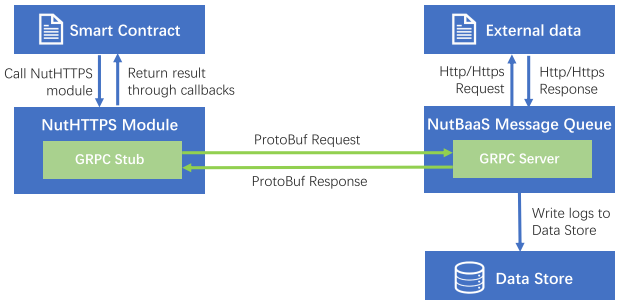


FIGURE 7. The basic implementation logic of NutHTTPS module.

TABLE 1. The parameter format of a request.

Request parameters	Type	Instruction
url	String	The uniform resource locator of the external data source
params	Object	The parameter list of requests
requestType	String	Request type: GET, POST, PUT, DELETE
timeout	long	Timeout unit: milliseconds

contracts. As a result, we develop a module called NutHTTPS to make it easy for smart contracts to access external data through http or https, which can work for various frameworks including Hyperledger Fabric and Ethereum. Figure 7 shows the basic implementation logic of the module:

The NutHTTPS module is primarily responsible for defining data specifications for smart contracts to interact with external data, submitting requests to NutBaaS Message Queue, and processing the results returned from the message queue. In order to support the diversification of programming languages for smart contracts, NutBaaS uses Protocol Buffers (protoBuf) [37] to define the parameter format for requests to interact with external data, and interacts with NutBaaS Message Queue via the GRPC/GRPCS protocol. ProtoBuf is a Google's language-neutral, platform-neutral, extensible mechanism for serializing structured data - like XML, but smaller, faster, and simpler. The data interaction format defined by ProtoBuf can be compiled into GRPC Stub code (interface code) for different programming languages. GRPC [38] is an open-source remote procedure call system initially developed at Google, which uses HTTP/2 for transport and ProtoBuf as the interface description language. NutBaaS uses protoBuf to define the parameter type of a request, as shown in Table 1, and generate different GRPC Stub code for different programming languages.

Developers can import Stub code for the corresponding programming language according to their needs. When it is necessary to interact with external data, the developer only needs to fill in the relevant parameters of Table 1 and send a request to the NutBaaS Message Queue through GRPC. The Stub code converts the parameters represented by the high-level language into a messages in ProtoBuf format and sends a request to the NutBaaS Message Queue. The NutBaaS

message queue mainly converts and verifies the message in ProtoBuf format through a GRPC server and constructs a HTTP/HTTPS request to interact with external data. During this process, all request logs are logged to the backend database for later queries. After the GRPC server gets the result of the request, it also returns the result to NutHTTPS in ProtoBuf format and finally to the smart contract that initiated the request. With the NutHTTPS module, developers can easily access any data outside the contracts, as long as it is accessible via the network. In addition, developers can use NutHTTPS to build a logical channel between different types of chains so that different chains can access each other's data.

D. SMART CONTRACTS SECURITY VULNERABILITY DETECTION

Smart contracts, which run on each node of the blockchain network in the form of chain scripts, provide an application-level extension interface for the blockchain. Based on these interfaces, smart contracts help people apply blockchain technology to more complex business scenarios, and as a result, they often carry a large amount of digital assets or commercial interests in many scenarios. At the same time, smart contracts are highly autonomous, meaning that once a smart contract is deployed, it cannot be changed at will. Therefore, it is especially important to perform security vulnerability detection before deploying smart contracts. Especially after the DAO incident, people further realized the importance of the security issues of smart contracts, which may cause huge economic losses.

In recent years, the identification and detection of smart contracts security vulnerabilities has become a research hotspot in academia. Traditional code defect identification and detection are mainly divided into two types, namely, code defect detection based on test cases [39] and code defect detection based on static analysis [40]. The former is to input test cases to the program under test, simulate the execution process of the program, observe the output of the program and speculate on possible defects in the program; the latter is to identify or find potential defects through automatic scanning and analysis of the program code. Currently, researchers mainly use static analysis methods (such as symbolic execution, model checking) to detect security vulnerabilities in smart contracts. In the studies [41] and [42], symbolic execution technique is used to detect potential types of known vulnerabilities in smart contracts at the bytecode level. In addition, multiple smart contracts verification systems [43]–[46] use model checking techniques to detect potential vulnerabilities in smart contracts. These systems, based on the attributes or assertions given by the users, use model checking technique to verify that whether the target contract implementation has relevant characteristics, thereby discovering potential vulnerabilities in the contract. However, the existing vulnerability detection techniques rely on heuristic strategies, such as depth (or breadth) prioritized search in symbolic execution techniques, which may result in higher computational costs in order to achieve analytical accuracy.

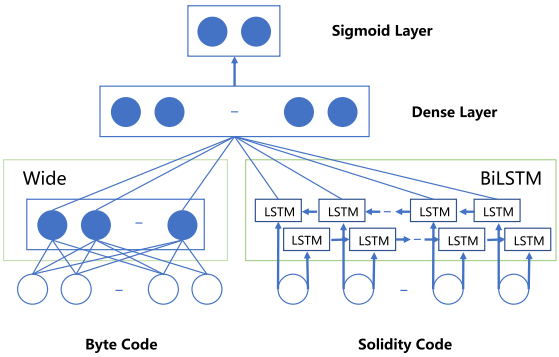


FIGURE 8. The learning model for smart contracts vulnerability detection.

On NutBaaS, we use existing tagged data (many contracts of known defect types) to initially establish a learning model, use machine learning algorithms to determine contract vulnerabilities, and reduce the cost of heuristics strategies.

In the previous work, we have collected the source code of 19,000 smart contracts and their bytecode files from Ethereum. Through manual verification, we found 11 common types of code defects in smart contracts, such as integer overflow, transaction failure exceptions, transaction status dependency, and so on. In the data annotation phase, we have manually selected 2000 smart contracts from the dataset and manually verified them, i.e., detected whether each contract has the above 11 code defects.

In the feature extraction phase, we are ready to incorporate the source code and bytecode files of the smart contracts into the feature consideration. The source code defines the logical behavior of the contract program. We use the way of traversing the code AST (Abstract Syntax Tree) to obtain the behavior characteristics of the code, taking the timing characteristics of the contract bytecode file into account. We will measure contract defects from different perspectives by combining the two types of features.

In the model training phase, considering that the contract source AST has a complex logical structure, we model it using bidirectional LSTM (ie, BiLSTM, Bi-directional Long Short-Term Memory). At the same time, since the contract bytecode file only contains timing information, we use a one-way LSTM (Long Short-Term Memory) to model it. The model architecture is shown in Figure 8. On this basis, we use a manually validated data set for model training, and finally use the trained model to predict contract defects. In the future, we will continue to improve our model from different perspectives to improve the accuracy of forecasts and release it as an official service for users.

VI. DISCUSSION

In recent years, the concept of BaaS has become popular, and a large number of companies have released their BaaS platforms and offered a variety of innovative services to seize the market. The blockchain infrastructure (or service) provided by the BaaS platform aims to bring convenience to developers and ensure the reliability and security of the blockchain

system (or application), but the service providers often ignore the reliability of the infrastructure itself. The availability and reliability of BaaS infrastructure is important for those planning to contract or deliver shared ledgers through these environments, mostly in order to keep up with strict service level agreements (SLAs) [47]. However, current service providers still lack an effective means to evaluate the reliability of these infrastructures, and optimize the relevant components based on the results of the evaluation to improve the overall service quality of the BaaS platform. Carlos Melo et al. used the Dynamic Reliability Block Diagram (DRBD) to evaluate the reliability of the BaaS infrastructure [48]. By understanding system availability and reliability in advance, these service providers can apply high-availability technologies such as redundant or preventive maintenance and improve their SLAs. The study was based solely on the experimental environment of Hyperledger Cello [49], which ultimately resulted in system downtime, but did not perform a comprehensive reliability assessment analysis. In the future, seeking a comprehensive and detailed BaaS reliability assessment through modeling will play an important role in the development of BaaS.

Blockchain has received more and more attention because of its potential to decentralise, disintermediate, and enable 'trustless' interactions. In recent years, BaaS offerings have gradually emerged to provide the underlying supporting infrastructure, aiming to reduce the uptake barriers of the technology. However, an interesting characteristic of BaaS is that it reintroduces an intermediary in the form of a service provider (e.g., IBM, Microsoft, Amazon, etc.), who often has a relationship with certain participants in the network (for example, some participants may have more power to control the infrastructure than others through their arrangements with service providers). This 'recentralisation' introduces new trust considerations as they relate to the provider. As a third-party provider of blockchain services, BaaS seems to run counter to the decentralized trust mechanism of blockchain. In the study [50], Jatinder Singh et al. analyzed the recentralisation and the trust considerations of BaaS in detail, particularly with respect to the role of service providers. At present, the views on this issue introduced by BaaS are mainly divided into two factions. Some people believe that BaaS introduces the role of a third-party service provider, which is contrary to the decentralization and trustless mechanism of the blockchain. Others believe that BaaS reduces the uptake barriers of development and improves the security and reliability of blockchain applications, thus, would drive the development of blockchain technology. They believe that the recentralisation and the trust issue of BaaS can be solved by other means. We prefer the latter view that the emergence of BaaS has more advantages than disadvantages for the development of blockchain. At present, there are also a few studies proposing some solutions to the issue. Study [51] introduces a novel Blockchain-as-a-Service paradigm which adopts deployable components to reconstruct the open and decentralized blockchain service. This paradigm increases the transparency of BaaS when deploying and running blockchain systems to a certain degree, but it cannot prevent collusion between tenants and providers. In the future, we will study this issue and seek a more complete solution.

VII. CONCLUSION AND FUTURE WORK

This paper proposes a more complete platform to make up for the shortcomings of the current BaaS platform, especially in terms of reliability and security. We also introduce some more advanced technical services, such as IdentityChain technology and smart contracts security vulnerability detection. In addition, we discuss the reliability of BaaS infrastructure and the new trust considerations arising from BaaS, which would undermine the security, decentralization and âĂŸtrustless' mechanism of blockchain.

In the future, our work can be extended in different aspects: **(1) Transparency of blockchain deployment and runtime:** Applying a new service paradigm (maybe similar to that described in the study [51]) to NutBaaS. Increasing the transparency of blockchain deployment and runtime through the new service paradigm to reduce the damage of BaaS platform as a third party to the decentralization of blockchain. **(2) Reliability of BaaS infrastructure (service):** Seeking a more detailed and versatile evaluation method for BaaS infrastructure. In this way, the service providers or the service users can optimize the relevant components or take corresponding preventive measures according to the evaluation results. **(3) Security of smart contracts:** Refining the existing machine learning model and conducting research on smart contracts performance optimization (such as reducing the gas consumption of contracts) and automatic repair.

REFERENCES

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Tech. Rep., 2008.

[2] M. Swan, *Blockchain: Blueprint for a New Economy*. Newton, MA, USA: O'Reilly Media, 2015.

[3] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, and Y. Manevich, "Hyperledger fabric: A distributed operating system for permissioned blockchains," in *Proc. 13th EuroSys Conf.*, 2018, p. 30.

[4] P. Hunt, M. Konar, F. P. Junqueira, and B. Reed, "ZooKeeper: Wait-free coordination for Internet-scale systems," in *Proc. USENIX Annu. Tech. Conf.*, vol. 8, no. 9, Boston, MA, USA, 2010.

[5] J. Kreps, N. Narkhede, and J. Rao, "Kafka: A distributed messaging system for log processing," in *Proc. NetDB*, 2011, pp. 1–7.

[6] (2016). *IBM Blockchain*. [Online]. Available: https://www.ibm.com/blockchain

[7] (2017). *Microsoft Azure Blockchain Solutions*. [Online]. Available: https://azure.microsoft.com/en-in/solutions/blockchain/

[8] (2018). *Ethereum Blockchain as a Service on Azure*. [Online]. Available: https://azure.microsoft.com/en-us/blog/ethereum-blockchain-as-a-service-now-on-azure/

[9] *AWS Blockchain Partners*. [Online]. Available: https://aws.amazon.com/partners/blockchain/

[10] J. Benet and N. Greco, "Filecoin: A decentralized storage network," Protocols Labs, Tech. Rep., 2018.

[11] J. Benet, "IPFS-content addressed, versioned, P2P file system," 2014, *arXiv:1407.3561*. [Online]. Available: https://arxiv.org/abs/1407.3561

[12] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, pp. 1–32, Apr. 2014.

[13] S. King and S. Nadal, "Ppcoin: Peer-to-peer crypto-currency with proof-of-stake," Tech. Rep., Aug. 2012.
[14] M. Castro and B. Liskov, "Practical Byzantine fault tolerance," in *Proc. OSDI*, vol. 99, 1999, pp. 173–186.
[15] R. C. Merkle, "A digital signature based on a conventional encryption function," in *Proc. Conf. Theory Appl. Cryptograph. Techn.* Springer, 1987, pp. 369–378.
[16] *Merkle Patricia Tree*. [Online]. Available: https://github.com/ethereum/wiki/wiki/Patricia-Tree
[17] J. Yli-Huumo, D. Ko, S. Choi, S. Park, and K. Smolander, "Where is current research on blockchain technology? A systematic review," *PLoS ONE*, vol. 11, no. 10, 2016, Art. no. e0163477.
[18] J. Kwon, "Tendermint: Consensus without mining," Tech. Rep., May 2014.
[19] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2016, pp. 839–858.
[20] I. Eyal, A. E. Gencer, E. G. Sirer, and R. van Renesse, "Bitcoin-NG: A scalable blockchain protocol," in *Proc. 13th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2016, pp. 45–59.
[21] *Amazon Elastic Computing Cloud*. [Online]. Available: https://aws.amazon.com/ec2/
[22] *Google App Engine*. [Online]. Available: http://code.google.com/appengine
[23] *Dedicated Server, Managed Hosting, Web Hosting by Rackspace Hosting*. [Online]. Available: http://www.rackspace.com
[24] G. C. Fox, V. Ishakian, V. Muthusamy, and A. Slominski, "Status of serverless computing and function-as-a-service (FaaS) in industry and research," 2017, *arXiv:1708.08028*. [Online]. Available: https://arxiv.org/abs/1708.08028
[25] Z. Zheng, J. Zhu, and M. R. Lyu, "Service-generated big data and big data-as-a-service: An overview," in *Proc. IEEE Int. Congr. Big Data*, Jun. 2013, pp. 403–410.
[26] *An Open-Source System for Automating Deployment, Scaling, and Management of Containerized Applications*. [Online]. Available: https://kubernetes.io/
[27] P. McCorry, S. F. Shahandashti, and F. Hao, "A smart contract for boardroom voting with maximum voter privacy," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.* Springer, 2017, pp. 357–375.
[28] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the Internet of Things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016.
[29] I. Weber, V. Gramoli, A. Ponomarev, M. Staples, R. Holz, A. B. Tran, and P. Rimba, "On availability for blockchain-based systems," in *Proc. IEEE 36th Symp. Reliable Distrib. Syst. (SRDS)*, Sep. 2017, pp. 64–73.
[30] T. T. A. Dinh, J. Wang, G. Chen, R. Liu, B. C. Ooi, and K.-L. Tan, "BLOCKBENCH: A framework for analyzing private blockchains," in *Proc. ACM Int. Conf. Manage. Data*, 2017, pp. 1085–1100.
[31] P. Zheng, Z. Zheng, X. Luo, X. Chen, and X. Liu, "A detailed and real-time performance monitoring framework for blockchain systems," in *Proc. IEEE/ACM 40th Int. Conf. Softw. Eng., Softw. Eng. Pract. Track (ICSE-SEIP)*, May 2018, pp. 134–143.
[32] *Jaeger: Open Source, End-to-End Distributed Tracing*. [Online]. Available: https://www.jaegertracing.io
[33] *A Distributed, Restful Search and Analytics Engine Capable of Solving a Growing Number of Use Cases*. [Online]. Available: https://www.elastic.co/products/elasticsearch
[34] D. Johnson, A. Menezes, and S. Vanstone, "The elliptic curve digital signature algorithm (ECDSA)," *Int. J. Inf. Secur.*, vol. 1, no. 1, pp. 36–63, Aug. 2001.
[35] H. Mayer, "ECDSA security in bitcoin and ethereum: A research survey," CoinFaabrik, Buenos Aires, Argentina, Tech. Rep., Jun. 2016.
[36] *A Reliable Bridge Between Smart Contracts and the Internet*. [Online]. Available: https://github.com/oraclize
[37] *Protocol Buffers: A Language-Neutral, Platform-Neutral Extensible Mechanism for Serializing Structured Data*. [Online]. Available: https://developers.google.com/protocol-buffers/
[38] *GRPC: A High Performance, Open-Source Universal RPC Framework*. [Online]. Available: https://grpc.io/
[39] J. Jaffar, V. Murali, and J. A. Navas, "Boosting concolic testing via interpolation," in *Proc. ESEC/SIGSOFT FSE*, Aug. 2013, pp. 48–58.
[40] B. Blanchet, P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Miné, D. Monniaux, and X. Rival, "A static analyzer for large safety-critical software," *ACM SIGPLAN Notices*, vol. 38, no. 5, pp. 196–207, 2003.
[41] L. Luu, D.-H. Chu, H. Olickel, P. Saxena, and A. Hobor, "Making smart contracts smarter," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 254–269.
[42] I. Nikolić, A. Kolluri, I. Sergey, P. Saxena, and A. Hobor, "Finding the greedy, prodigal, and suicidal contracts at scale," in *Proc. 34th Annu. Comput. Secur. Appl. Conf.*, 2018, pp. 653–663.
[43] K. Bhargavan, A. Delignat-Lavaud, C. Fournet, A. Gollamudi, G. Gonthier, N. Kobeissi, N. Kulatova, A. Rastogi, T. Sibut-Pinote, and N. Swamy, "Formal verification of smart contracts: Short paper," in *Proc. ACM Workshop Program. Lang. Anal. Secur.*, 2016, pp. 91–96.
[44] G. Bigi, A. Bracciali, G. Meacci, and E. Tuosto, "Validation of decentralised smart contracts through game theory and formal methods," in *Programming Languages With Applications to Biology and Security*. Springer, 2015, pp. 142–161.
[45] S. Kalra, S. Goel, M. Dhawan, and S. Sharma, "ZEUS: Analyzing safety of smart contracts," in *25th Annu. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, 2018, pp. 1–15.
[46] P. Tsankov, A. Dan, D. Drachsler-Cohen, A. Gervais, F. Buenzli, and M. Vechev, "Securify: Practical security analysis of smart contracts," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2018, pp. 67–82.
[47] P. Patel, A. H. Ranabahu, and A. P. Sheth, "Service level agreement in cloud computing," Tech. Rep., 2009.
[48] C. Melo, J. Dantas, D. Oliveira, I. Fé, R. Matos, R. Dantas, R. Maciel, and P. Maciel, "Dependability evaluation of a blockchain-as-a-service environment," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jun. 2018, pp. 909–914.
[49] HyperCello. (2017). *Setup Cello Platform*. [Online]. Available: https://github.com/hyperledger/cello/blob/master/docs/setup.md
[50] J. Singh and J. D. Michels, "Blockchain as a service (BaaS): Providers and trust," in *Proc. IEEE Eur. Symp. Secur. Privacy Workshops (EuroS&PW)*, Apr. 2018, pp. 67–74.
[51] Z. Wan, M. Cai, J. Yang, and X. Lin, "A novel blockchain as a service paradigm," in *Proc. 1st Int. Conf. Services Conf. Fed. (SCF)*, Seattle, WA, USA, Jun. 2018, pp. 267–273. doi: 10.1007/978-3-319-94478-4_20.

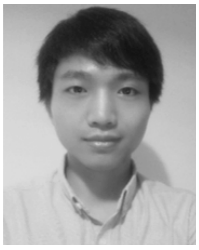
WEILIN ZHENG is currently pursuing the B.A.D. degree with the School of Data and Computer Science, Sun Yat-sen University, Guangzhou, China. His research interests include performance monitoring, blockchain computing power utilization, and blockchain-based decentralized applications.

ZIBIN ZHENG received the Ph.D. degree from The Chinese University of Hong Kong, in 2011. He is currently a Professor with the School of Data and Computer Science, Sun Yat-sen University, China, where he serves as the Chair for the Software Engineering Department. He has published over 120 international journals and conference papers, including 3-ESI highly cited articles. According to Google Scholar, his articles have more than 7000 citations, with an H-index of 42. His research interests include blockchain, services computing, software engineering, and financial big data. He was a recipient of several awards, including the top 50 influential articles in blockchain of 2018, the ACM SIGSOFT Distinguished Paper Award from ICSE 2010, and the Best Student Paper Award from ICWS 2010. He has served on BlockSys 2019 and CollaborateCom 2016. He has served as the PC Co-Chair for IoV 2014 and the General Co-Chair for ICIOT 2018 and SC2 2019.

XIANGPING CHEN received the Ph.D. degree from Peking University, in 2010. She is currently an Associate Professor with the School of Communication and Design, Sun Yat-sen University. Her research interests include software engineering and mining software repositories.

KEMIAN DAI has 14 years of experience in software development. He is the in charge of many ERP systems and other large-scale project implementation. He has been responsible for architecture planning, design, implementation, and problem solving of many large-scale projects and development of many big data projects, including HengTuo Freight Rubik's Cube, the Precision Advertising Platform, and the Cloud Finance Platform. He provides professional open source technology consulting services for many enterprises in China and helps enterprises to improve and enhance their information construction capabilities. He was awarded IBM 2009 to 2010 Technical Elite, IBM 2010 Intellectual Leader, and other titles. At present, he mainly studies about cloud computing, big data, machine learning, intelligent recommendation, and blockchain. At present, he mainly leads the team to develop BD BaaS, which provide many domestic enterprises with blockchain solutions and open source blockchain technologies that provide appropriate scenarios. At the same time, he has published several articles on blockchain, and based on the BaaS platform, he carried out cooperation with a number of colleges and universities on blockchain teaching platform to promote the development of blockchain talent construction.

PEISHAN LI is the main developer of the BaaS Platform. She is an experienced developer who has participated in many blockchain projects, such as DongGuan Blockchain Enterprise Database, the WangAn Trust Certificate Storage Platform, and HitChain distribution. She has also participated in the design and development of BaaS platform and BD blockchain, which obtained a number of technical copyrights and patents. Her current research interests include BaaS development and blockchain services, such as ETH, fabric, IPFS, and stellar.



RENFEI CHEN has nine years of experience in computer software and Internet industry, long-term focus on product quality and implementation areas, proficient in product quality automation construction, and project continuous automation deployment. He has completed a number of high-concurrence and high-availability deployment frameworks supporting provincial government. System performance tuning can support user levels upto 10 000 people. He continues to provide performance tuning for a number of well-known enterprises. At present, his research interests include the blockchain BaaS, Hyperledger Super Accounts, ETH, IPFS, and Bubi blockchain.