SDFS: A Scalable Data Feed Service for Smart Contracts

Juan He and Rong Wang
Digital Society & Blockchain Laboratory
Beihang University, Beijing, P. R. China
{xiongbao_hj & wangrong}@buaa.edu.cn

Wei-Tek Tsai
Digital Society & Blockchain Laboratory
Beihang University, Beijing, P. R. China
Andrew International Sandbox Institute, Qingdao, China
IoB Laboratory of Guizhou BigData Experimental Areas
MIIT CCID Blockchain Institue, Qingdao, China
Beijing Tiande Technologies, Beijing, China
tsai@tiandetech.com

Enyan Deng
Beijing Tiande Technologies, Beijing, China
deng@tiandetech.com

Abstract—**Smart Contract (SC) is distributed, trustless, programmable, and tamper-resistant, which can realize secure and efficient information exchange, value transfer and asset management. However, SCs can only access data on their own blockchains (BCs), and can't get external information by themselves. Data feed service is a third-party data source provider that supports a broad range of data requests will thus be critical to SC ecosystems. In this paper, we propose a novel scalable data feed service for SCs, SDFS, that not only ensures that the data acquisition process is reasonable, but also guarantees that the system service is scalable. In addition, we also designed a reputation evaluation strategy for nodes to determine whether each node is a malicious node. After each feedback, the reputation value of each node will be updated. Besides that, SDFS keeps content providers auditable by using BC to preserve evidence of data processing processes. As a data feed service for SCs, it can provide the data of off-chain into the BC truly and accurately, so as to ensure the authenticity of the data on the BC and realize the SC to interact with the outside world.**

Keywords-blockchain; smart contracts; data feed service; data authentication

I. INTRODUCTION

Smart Contracts (SCs) has received attention recently. The original intention of the design is to embed certain digital forms of control into computer transaction protocols for the execution of contract terms without the need for third party credible authority. As a computer transaction protocol to implement contract terms, it embeds some valuable physical entities controlled by digital form, acts as agents trusted by all parties to the contract, performs the contract efficiently and safely, and creates a variety of digital assets.

Until 2008, Nakamoto proposed an encrypted digital currency system, Bitcoin [1], which can carry out peer-to-peer transactions without trust. It was found that its underlying technology BC naturally compatible with SC. BC can encapsulate the complex behavior of distributed nodes with the programmability of SC, and SC can be trusted with the distributed infrastructure of BC. Since then, SC has been revitalized. BC has gradually become the main computing scenario of SC, and SC have been endowed with new meaning. With the continuous maturity of BC technology, some scholars proposed that SC should be legalized and its generation and implementation process should be improved [2] [3].

BC is deterministic, which means that it is a reflection of one specific event that occurs one after another, that is a series of transactions with specific sequence and causal relationship. However, the off-chain information is not in the case, it can be discontinuous, so this information cannot be trusted or used in the BC. This feature of BC gives it invariance, but reduces flexibility and scalability.

The off-chain information is somewhat uncertain, which means that events do not occur in a specific order, causing transparency problems. Because in this process, how to ensure that the data transmitter does not modify the data in the process, how to authenticate whether the node receiving external data is a single node or all nodes receive synchronously, and how to reach consensus in the BC are all the problems faced by data transparency.

However, a SC is a closed system, which is the guarantee execution mechanism under the condition of satisfying conditions. And many SC application scenarios output corresponding results according to some external events. To really function in the outside world, it needs the outside world to input information to it. For example, based on the BC-based digital currency exchange [4] [5], a BC system needs to obtain the price of a digital currency at a specific time. This requirement is simple for traditional Internet systems, which can be achieved by periodically entering price data or accessing data through third-party interfaces. But for BC-based system, the situation is different. In the block chain system, the external

978-1-7281-0945-9/19/$31.00©2019 IEEE

data source as a third party will send to the BC, but because the data transmitter is a centralized participant, the authenticity of the data comes from the sender's subjective judgment, so the SC cannot be directly accessed to obtain relevant external data.

The data feed service is a platform that provides external off-chain information and provides the necessary conditions for SC to meet the runtime. It can access Internet data by establishing a trusted data gateway between BC and Internet. The BC accesses APIs, URLs, search engines through data feed services to access data in the Internet.

SCs rely on data feed services to provide data, but feed services face three main issues: data reliability issue, communication reliability issue and trustee issue.

Data reliability: Data quality is directly affected by the quality of the data source. How to ensure the reliability of data sources is an issue that data feed service designers need to consider. The most typical method is the reputation mechanism used in ChainLink [6], which ranks the data in the data source by quality. However, there is an important issue with distributed data feed services, namely consensus issues. This problem is caused by inconsistency of data obtained by different nodes, and data inconsistency is caused by various reasons, such as data time, network delay and changes in node processing speed. In this case, the system cannot reach a consensus.

Communication reliability: Communication reliability issue is more important in centralized systems. In the distributed system, data is usually negotiated and backed up, but in a centralized system, data cannot be backed up. Therefore, in a centralized data feed service, communication reliability is ensured by designing a communication protocol that satisfies communication security requirements. The most typical secure communication protocol is the TLSnotary protocol [7].

Trustee: The introduction of the predictive machine system is to solve the data credibility problem to prevent data from being tampered with in transmission and to make the system that needs the data believe that the data provided by the data feed service node is reliable. Therefore, trustee is often introduced into data feed service system to prove data sources. Trustees can be centralized or distributed. In a centralized system, some of the communication data is stored as evidence in the trustee for later review. In the distributed system, this problem is easier to resolve. If the BC-based trustee can ensure that the data on the chain is not tampered with.

To address the above problems, this paper proposes SDFS, a novel and scalable data feed service that not only ensures the data collection process is creditable, but also ensures the scalability of system service. This paper is organized as follows: Section 2 introduces relevant technical background; Section 3 presents SDFS, a scalable data feed service for SCs; Section 4 summarizes the work of this paper and presents future work.

II. BACKGROUND

A. BC and SCs

BC is a kind of chain data structure, which combines data blocks in sequence according to time sequence, and guarantees non-tampering and non-forgery distributed ledgers by cryptography. The BC system consists of a data layer, network layer, consensus layer, an incentive layer, SC layer, and application layer. The data layer encapsulates the underlying data blocks and related data encryption and time stamping and other basic data and basic algorithms. The network layer includes a distributed networking mechanism, a data propagation mechanism, and a data verification mechanism. The consensus layer mainly encapsulates all kinds of consensus algorithms of network nodes. The incentive layer integrates economic factors into the technology system of BC, mainly including the distribution mechanism and distribution mechanism of economic incentives. The SC layer encapsulates all kinds of scripts, algorithms and SCs, which is the basis of BC programmability. The application layer encapsulates various application scenarios and cases of the BC.

BC-based SCs, including transaction preservation and state processing, are all done on the BC, as shown in Figure 1. The transaction mainly contains the data that needs to be sent, and time is the description information of these data. When the transaction or event information is passed to the SC, the resource status in the contract resource set is updated, which in turn triggers the SC for state machine judgment. If the event action satisfies the trigger condition, the contract action is automatically and correctly executed by the state machine according to the participant's preset information.

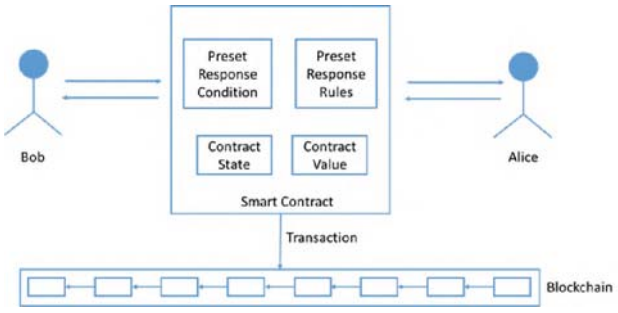

Figure 1. SC Execution Model

B. TLS and TLSnotary Protocol

Secure Transport Layer Protocol (TLS) is used to provide confidentiality and data integrity between two communication applications [8]. The protocol consists of two layers: TLS Record Protocol (TLS Record) and TLS Handshake Protocol (TLS Handshake). TLS consists of three basic phases:

(1) Key algorithm for peer-to-peer negotiation support;

(2) Exchange public key based on private key encryption, identity authentication based on PKI certificate;

(3) Data transmission confidentiality based on public key encryption.

TLSnotary is a service that introduces third-party auditors that validate TLS session data exchanged between the client and server [7]. To provide this functionality, the protocol needs to change the TLS protocol, such as the introduction of a dedicated client audit protocol. The TLSnotary certificate is based primarily on the secure transport layer protocol TLS 1.1,

which is used to provide confidentiality and data integrity between two communication applications. The biggest advantage is that it is independent of the application protocol, and higher layer protocols can be transparently distributed over the TLS protocol. This algorithm allows the auditor to verify that the auditee has not tampered with the data retrieved from the data source by splitting the TLS master key generated by the auditee to retain the secret (this feature applies only to TLS v1.0 and v1.1) until the auditee promises the hash value of result.

C. Existing Data Feed Service Solutions

The most common data feed service currently available is Oraclize, which is the leading oracle service for SCs and BC applications, serving thousands of requests every day on platforms like Ethereum, Rootstock, R3 Corda, Hyperledger Fabric and EOS [9-12]. The Oraclize Engine powers the service for both BC-based and non-BC-based application. The TLSnotary proof leverages a feature of the TLS 1.0 and 1.1 protocols which enable the splitting of the TLS master key between three parties: the server, an auditee and an auditor. In this scheme, Oraclize is the auditee while a locked-down AWS instance of a specially-designed, open-source Amazon Machine Image acts as the auditor. However, this system has some problems, because the TLSnotary protocol can only prove that Oraclize has not tampered with the response data, and cannot prove that there is no tampering with the request data. At the same time, because it creates a centralized control server, it does not meet the high standards of tamper resistance required by untrusted SCs.

The Town Crier system addresses this problem by using trusted hardware, namely the Intel SGX instruction set, a new capability in certain Intel CPUs [13]. Town Crier allows special applications to interact with sites that support HTTPS, such applications are executed within the SGX enclave. The application establishes secure TLS connection with the website and parses its content, which can then be used as send to SC. However, Town Crier has some major limitations. First, it positions Intel as the trusted party needed to perform remote proofs. Second, its security relies on the security of the SGX framework (recently serious attacks [14] [15]) and its proof infrastructure security, which is particularly undesirable because SGX proves that infrastructure is the weakest link security system (i.e., a leaked proof private key allows an opponent to prove any application).

Chainlink's decentralized oracle network provides the same security guarantees as SCs themselves [7], which aims to decentralize Town Crier applications by forming a network of them (to detect and deal with possible inconsistencies). Unfortunately, this design does not solve the main drawbacks of Town Crier [14].

III. SDFS: A Scalable Data Feed Service for Smart Contracts

A. System Architecture

SDFS includes three main components: SDFS SC, SDFS Server, and SDFS Auditors BC. The SDFS SC is deploy on BC, such as Ethereum, Beihang BC, Laoshan BC, Rootstock, Eris, etc. The SDFS Server resides on the server, which consists of many nodes. The SDFS Auditors BC is a BC, which provide audit services.

1) SDFS SC: The SDFS Smart Contract is deployed on BC such as Ethernet, EOS, Beihang BC, Laoshan BC, etc. When users need external data, they only need to refer to the smart contract in their own smart contract, and then invoke the interface according to the method described in the API document. If you want to use SDFS in your own private BC or permissioned BC, you just need to deploy SDFS Smart Contract on your own private BC or permissioned BC. After you deploy it, you can call it as public BC. The SDFS Server provides a variety of data source servers, including API access, URL resources, data search engines, BC content data, etc.

If you want to use SDFS in your own private BC or permissioned BC, you just need to deploy SDFS SC on your own private BC or permissioned BC. After you deploy it, you can invoke it like public BC. SDFS supports a variety of data source servers, including API, URL resources, data search engines, BC content data, etc.

2) SDFS Server: SDFS Server is a data feed service for SCs, which consists of many nodes. It adopts an extensible and distributed architecture and can be deployed dynamically according to the data requirements of SCs. SDFS Server obtains data by accessing Internet data sources. Data integrity and security are guaranteed by TLS protocol. SDFS Auditor BC is used to verify the authenticity of SDFS Server. After being verified, the request data is returned to SDFS SC.

3) SDFS Auditor BC: SDFS Auditor BC provides audit services for SDFS Server. By storing the raw request data of smart control on the BC, TLSNotary algorithm is used to verify that SDFS Server has not tampered with the retrieved data. The algorithm allows SDFS Auditor BC to verify whether the SDFS Server has not tampered with the data retrieved from the data source by dividing the SDFS Server's secret generated by the TLS master key (this function only applies to TLS v1.0 and v1.1) until the hash value of the SDFS Server's promised result is obtained, and then it will not be displayed until the retained secret is obtained. The node checks the content of SDFS Server request and the data of data source response, and consensus the results of the checking. Consensus is issued to authenticate the verification result.

The overall architecture of the SDFS system is shown in the figure 2.

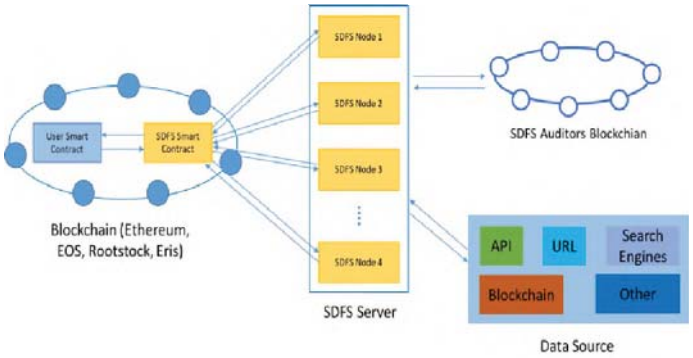

Figure 2. SDFS System Architecture

B. *System Workflows*

The implementation process of SDFS is as follows:

(1) When the data on BC cannot support the normal operation of User SCs, the User SCs use the API sends request messages to SDFS SC;

(2) When SDFS SC received the data request of User SCs, it forwarded the request to SDFS Server node;

(3) To ensure the security of the service, the SDFS Server node should forwarded the hash of request to SDFS Auditors BC as evidence. As the same time, the SDFS Server node sends Client Hello to Web Server;

(4) The SDFS Auditors BC received the hash of request, and after the consensus of all nodes on BC, the hash of request is put on SDFS Auditors BC as evidence. Then each node generates a random number and broadcast it to other node;

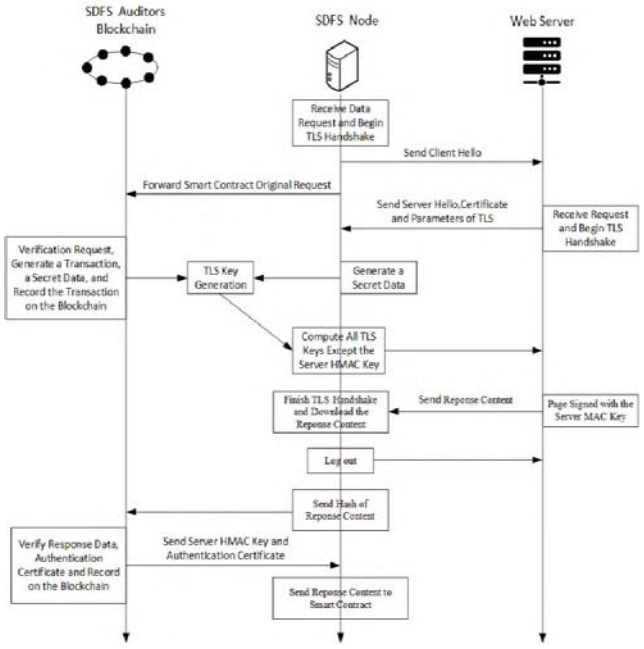

Figure 3. System Workflows

(5) After each node receives random Numbers from other nodes, it will do XOR on all random Numbers to get a final random data S2, and encrypt S2 with RSA algorithm;

(6) The SDFS Auditors BC sends RSA (S2) to SDFS server node. The SDFS Server node generates a random number S1 after it received RSA (S2) from SDFS Auditor BC. Then SDFS Server node sends the (RSA (S1) × RSA (S2)) as a message to the Web Server;

(7) The SDFS Server node computes S1 using an algorithm to generate H11 and H12, and send H12 to SDFS Auditors BC. SDFS Auditors BC generates H21 and H22 from S2 in the same way, and send H21 to SDFS server node.

(8) The SDFS Auditors BC generates Secret_Key2 from H12 and H22, and send part of it to SDFS Server node;

(9) The SDFS Server node uses the part of Secret_Key2, H11, and H21 to generate five keys used to encrypt communication. Because the SDFS Auditors BC still has a part of the key, so there is one key cannot be generating;

(10) The SDFS Server node sends the request to the Web Server. The Web Server receives the request and send the respond to the SDFS Server node. The SDFS Server node forwarded the hash of respond to the SDFS Auditors BC as evidence;

(11) The SDFS Auditors BC received the respond, and after the consensus of all nodes on BC, the hash of respond is put on SDFS Auditors BC as evidence. Then SDFS Auditors BC sends remaining Secret_Key2 to the SDFS Server node.

(12) The SDFS Server node receives the remaining key and generates the last key. And SDFS Server node can use these six keys to communicate securely with the client.

C. *Node Reputation Evaluation*

To prevent malicious data feed service nodes appear, such as the case of being attacked by hackers, we designed a node reputation mechanism, to provide users with a method to assess overall service performance data feed. To reward high reputation service node to ensure its proper operation and ensure high availability and performance. The reputation value of each node will be updated after each feed. Include the following steps:

(1) SDFS SC to receive data SDFS Node returned for verification. Verify that it includes the SDFS Auditor BC certificate and is consistent with the original request.

(2) If the verification passes, increase its reputation value. The reputation value is calculated according to the following formula: $Ri(t) = (1-Z)*Ri(t-1)+n/(n+1)*Z$, where $n> = 1$ is the correct number of times; i is the node No.; $0<Z<1$, when Z is large, the node reputation value increases rapidly, and the Z hour increases slowly;

If not, reduce its reputation value. The reputation value is calculated according to the following formula:

Ri(t) = Y*Ri(t-1)/m, where 0<X<Y<1, m> = 1, m is the number of rounds in which the error occurred. t is the Request counts.

(3) When the node i reputation value is reduced to 0, the service right will be lost and processed offline.

IV. CONCLUSION

This paper proposed SDFS, a data feed service system that provides authenticated data feeds for SCs. This system has three distinct advantages over previous data feed service systems. First, in this system, the data in the trustee are untamable. The auditors as trustee are designed into a distributed system to improve the security of evidence. Second, to verify that the requests of user SC and the responses of the Web Server are matched, the system stored the requests and the responses on the SDFS Auditors BC as evidence. The last point, the SDFS server adopts an extensible and distributed architecture, it can be deployed dynamically according to the data requirements of SCs.

This system has realized the function of secure communication of data and secure storage of communication information. In the future, we can consider how to integrate the reputation mechanism of data sources into SDFS. And expand the function of the SDFS Server and diversify the way it collects data.

ACKNOWLEDGMENT

This work is supported by National Key Laboratory of Software Environment at Beihang University, National 973 Program (Grant No. 2013CB329601) and National Natural Science Foundation of China (Grant No. 61472032), (Grant No. 61672075) and (Grant No. 61690202).

REFERENCES

[1] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

[2] Wei-Tek Tsai, Ning Ge, Jiaying Jiang, Kevin Feng, and Juan He. Beagle: A New Framework for Smart Contracts Taking Account of Law[C]//2019 IEEE International Conference on Service-Oriented System Engineering (SOSE). IEEE, 2019: 134-13411.

[3] Wei-Tek Tsai, and Christine Jiang, "Three Key Principles of Smart Contracts," Jan. 17, 2019. https://mp.weixin.qq.com/s/j5Ec2Jit69lsKOu1iexFUg.

[4] Rong Wang, Wei-Tek Tsai, Juan He, Can Liu, and Enyan Deng. A Distributed Digital Asset-Trading Platform Based on Permissioned Blockchains[C]//International Conference on Smart Blockchain. Springer, Cham, 2018: 55-65.

[5] Wei-Tek Tsai, Zihao Zhao, Chi Zhang, Lian Yu, and Enyan Deng. A Multi-Chain Model for CBDC[C]//2018 5th International Conference on Dependable Systems and Their Applications (DSA). IEEE, 2018: 25-34.

[6] Steve Ellis, Ari Juels, and Sergey Nazarov.Chainlink: A decentralized oracle network. URL:https://www.smartcontract.com/link, 2017.

[7] Tlsnotary - a mechanism for independently audited https sessions. https://tlsnotary.org/TLSNotary.pdf, 2014.

[8] Dierks Tim, and Christopher Allen . The TLS Protocol Version 1.0[M]. RFC Editor, 1999.

[9] Wood Gavin. Ethereum: A secure decentralised generalised transaction ledger[J]. Ethereum project yellow paper, 2014, 151: 1-32.

[10] Brown R G, Carlyle J, Grigg I, et al. Corda: an introduction[J]. R3 CEV, August, 2016.

[11] Androulaki Elli, Barger Artem, Bortnikov Vita, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains[C]//Proceedings of the Thirteenth EuroSys Conference. ACM, 2018: 30.

[12] IO EOS. EOS. IO Technical White Paper v2[J]. 2018.

[13] Fan Zhang, Ethan Cecchetti, Kyle Croman, Ari Juels, and Elaine Shi. Town crier: An authenticated data feed for smart contracts. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2016.

[14] Nick Szabo. Formalizing and securing relationships on public networks. First Monday, 2(9), 1997.

[15] Nick Szabo. A formal language for analyzing contracts. URL: https://web.archive.org/web/20160810220820/http://szabo.best.vwh.n 2002.