Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20)
Special Track on AI in FinTech

SEBF: A Single-Chain based Extension Model of Blockchain for Fintech

Yimu Ji[1,2,3,4,5*], **Weiheng Gu**[1,3], **Fei Chen**[1,3], **Xiaoying Xiao**[1,3], **Jing Sun**[1,3], **Shangdong Liu**[1,2,3,4,5*]
Jing He[6,7*], **Yunyao Li**[6], **Kaixiang Zhang**[8], **Fen Mei**[8] and **Fei Wu**[3,4,5,9*]
[1]School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing, China
[2]Jiangsu High Technology Research Key Laboratory for Wireless Sensor Networks, Nanjing, China
[3]Institue of High Performance Computing and Bigdata, Nanjing University of Posts and Telecommunications, Nanjing, China
[4]Nanjing Center of HPC China, Nanjing, China
[5]Jiangsu HPC and Intelligent Processing Engineer Research Center, Nanjing, China
[6]Institute of Information Technology, Nanjing University of Finance and Economics, Nanjing, China
[7]School of Software and Electrical Engineering, Swinburne University of Technology, Hawthorn, Australia
[8]WeBank Co., Ltd., Shenzhen, China
[9]College of Automation, Nanjing University of Posts and Telecommunications, Nanjing, China
jiym@njupt.edu.cn, {wh.gu, 370696640}@qq.com, {njupt_shaw, sunjingnjupt}@163.com, lsd@njupt.edu.cn, 480245@qq.com, liyunyao1010@163.com, {kxzhang, crisiticmei}@webank.com, wufei_8888@126.com

Abstract

The traditional blockchain has the shortcoming that a single-chain can only deal with one or a few specific data types. The research question of how to make blockchain be able to deal with various data types has not been well studied. In this paper, we propose a single-chain based extension model of blockchain for fintech (SEBF). In the financial environment, we design a four-layer architecture for this model. By employing the external trusted oracle group and a financial regulator agency, a variety types of data can be effectively stored in the blockchain, such that the data type extension based on a single-chain is realized. The experimental results indicate that the proposed model can improve the efficiency of simplified payment verification.

1 Introduction

Since Satoshi Nakamoto published the Bitcoin white paper [Nakamoto, 2008], a brand-new technology called blockchain came into the spotlight. It can solve many challenges, such as data tamper resistance, traceability, and decentralization in untrusted networks [Dai et al., 2020]. At present, Bitcoin that only allows financial transactions has been replaced by Ethereum [Buterin, 2014], which can program the blockchain using smart contracts, enabling users to write more sophisticated and intelligent protocols.

The applications and the commercial value of blockchain have attracted widespread attention in artificial intelligence (AI) and financial communities. Blockchain can help guarantee the credibility of data in AI applications [Sarpatwar et al., 2019]. Blockchain also has broad applicability in financial systems and applications, such as cheque clearance [Kabra et al., 2020], auction system [An et al., 2019], and sharing economy [Abdur Rahman et al., 2019]. Blockchain systems have three characteristics: decentralization, non-tamperability, and traceability. These characteristics allow users to carry out secure transactions along with historical transaction verification or search in a mutually distrusted network. This destines that blockchain will overturn traditional security models. However, there exists a severe problem concerning the scalability of data types for blockchain: the block bodies of most blockchains use one or a few limited types of data for storage. For the blockchains represented by Bitcoin that concerns the maintenance of unspent transaction output (UTXO), the transactions stored within their blocks only support the transaction data of Bitcoin (coinbase transaction is also essentially the data transaction with bitcoin). In Ethereum, full nodes are responsible for maintaining the transaction tree, the state tree, and the receipt tree, which constitute the Merkle tree [Merkle, 1987]. These three trees in Ethereum store three different types of data, however, with the rapid development of blockchain, they may not meet the needs of various specific scenarios in the future. Therefore, there still exists the problem of poor scalability of data types

*Corresponding authors

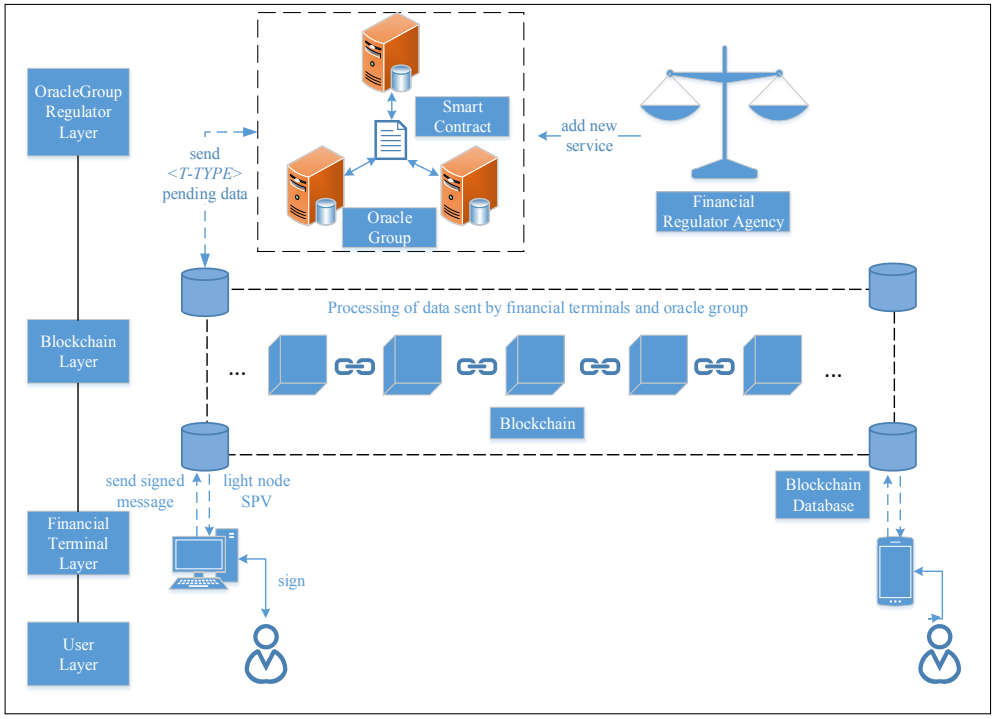

Figure 1. Four-layer model architecture of SEBF.

supported by the underlying blockchain.

1.1 Motivation and Contribution

At present, there exist two kinds of methods that focus on the data scalability of blockchain, including the cross-chain methods [Back et al., 2014; Singh et al., 2020] and the main chain extension methods [Liu et al., 2018]. These methods try to extend the block storage capacity and throughput of blockchain by adopting sidechains, changing block production processes, or optimizing the consensus protocol. However, the problem of how to make blockchain be able to deal with various data types has not been well studied.

Enabling an intelligent system to deal with various data types is an important research topic of AI [Chen et al., 2019]. In this paper, in order to alleviate the above-mentioned shortcomings of blockchain, we propose a single-chain based extension model of blockchain for fintech (SEBF). The contributions of our study can be summarized as follows:

(1) We design a four-layer architecture for our SEBF model, which contains the user layer, financial terminal layer, blockchain layer, and the oracle group regulator layer. To our knowledge, we are among the first to study the problem of single-chain based extension of data type in fintech blockchain.

(2) An external trusted oracle group and a financial regulator agency, e.g., the authorities like a bank, insurance company, etc., are introduced to verify the data type submitted by the blockchain, which makes the successfully verified data be recorded into the sub-tree collection in blockchain, and thus generating the corresponding Merkle tree. In this way, the data type in the blockchain is extended.

(3) Two smart contracts, i.e., extended function embedding and multi-type data processing smart contracts, are designed to facilitate cooperation between the blockchain layer and the oracle group regulator layer.

(4) We build a fintech alliance chain and deploy the oracle group along with the corresponding smart contract on the FISCO-BCOS platform[1]. Experimental results indicate that our SEBF model can improve the efficiency of Simplified Payment Verification (SPV) [Lin and Liao, 2017].

2 Related Works

The poor scalability of blockchain may restrict its rapid development [Liu et al., 2019]. A traditional blockchain can only store one or just a few types of data [Dai et al., 2019]. In order to overcome the scalability issues of blockchain, two types of approaches are adopted in the current mainstream research, including the cross-chain methods [Back et al., 2014] and the main chain extension methods [Lombrozo et al., 2015]. Both kinds of approaches mainly focus on the extension of storage capacity [Jiang and Wu, 2019] or transaction throughput [Sompolinsky and Zohar, 2013].

In terms of the cross-chain technology, Deng et al. [Deng et al., 2018] studied cross-chain technology based on sidechains and hash locks, and proposed a new cross-chain technology to realize communication between different blockchains. Due to the complexity of the cross-chain mechanism, there exist restrictions with regards to asset transfers and data exchange between different blockchains,

[1] https://github.com/FISCO-BCOS/FISCO-BCOS

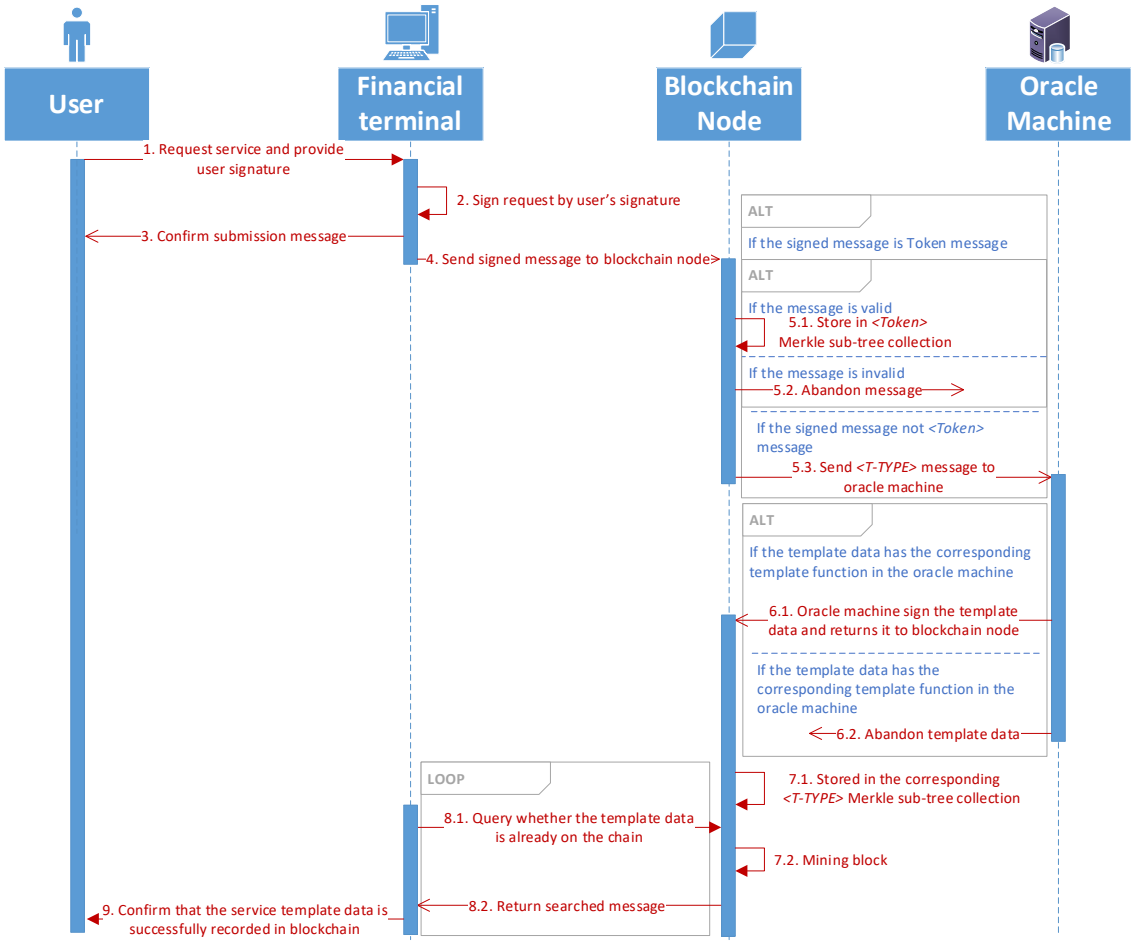

Figure 2. Overall system operation diagram.

which reduce usability and comfort for users. Focusing on this problem, Borkowski et al. [Borkowski et al., 2019] proposed the DeXTT cross-blockchain transfer protocol, which can realize cross-blockchain asset transfer without relying on any single blockchain. Atomic cross-chain exchange [Herlihy, 2018] tries to ensure the security of data transfer between the main chain and sidechain. This kind of works mainly studies the security and robustness of the cooperation between the sidechain and main chain, supporting the data extension indirectly with the help of sidechain.

Against the shortcomings of the cross-chain technology, the main chain extension is another feasible solution. In terms of the scalability on the main chain, bitcoincash[2] adjusts block production time and block size. The increase in block size will deteriorate the efficiency of SPV verification on light nodes. Enterprise Operation System (EOS) [Lee et al., 2017] increases blockchain throughput by using the Byzantine Fault Tolerance-Deligated Proof of Stake (BFT-DPOS) mechanism [Du et al., 2017], and Bitcoin-NG [Eyal et al., 2016] redefines the block production process. This kind of methods changes the block production process or optimizes the consensus protocol of the main chain to make effects similar to an extension.

There exist obvious differences between these two kinds of methods and our SEBF model. (1) These two kinds of methods mainly focus on the extension of block storage capacity and throughput. Contrary to them, we for the first time to study the data type extension problem and provide an effective extension solution. (2) The cross-chain methods promote extension based on the sidechains. Our single-chain based data type extension model is intrinsically different from these cross-chain works. The main differences are summarized in Table 1.

Comparison aspects	Cross-chain methods	Main chain methods	Our model
Single chain	-	√	√
Multiple chains	√	-	-
Extension of capacity and throughput	√	√	-
Extension of data type	-	-	√

Table 1. Differences between related works and our model.

[2] https://www.bitcoincash.org/

3 Systematic Model Design of SEBF

In this section, we first describe the system architecture. Then we introduce the detailed implementation of SEBF model.

3.1 System Architecture

As shown in Fig. 1, our SEBF model contains four layers: the user layer, financial terminal layer, blockchain layer, and the oracle group regulator layer. The first three layers correspond to the roles of user, financial terminal, and blockchain node, respectively. And the last layer corresponds to the roles of financial regulator agency, i.e., the authorities like a bank, insurance company, etc., and oracle group. The users are the participants who intend to purchase or use financial services. The financial terminals refer to the devices, e.g., the light nodes like a personal computer (PC) or mobile phone, that can query, submit, or handle certain financial services. The blockchain refers to an alliance chain established for the financial service, which only allows the authorized nodes to be added into the blockchain. Blockchain is responsible for storing the related template data that the financial institution allows. The financial institution refers to authoritative institutions like bank, insurance company, and securities and exchange commission. Here, we introduce the concept of template data. We use $< T-TYPE >$ to represent the template data, which is a data structure that is filled according to a certain transaction data format. For example, in the Bitcoin system, when a transaction is initiated, the address of the sender, the recipient's address, and the transfer amount need to be filled in, and the data structure containing this three data information is called template data. In our model, the oracle group consisting of several trusted devices called oracle machines serves as a unique bridge between the financial alliance chain and the external data. And it verifies the template data that needs to be verified in the alliance chain.

3.2 Detailed Implementation

The overall operation process of our model can be summarized as follows. As shown in Fig. 2, firstly, the user selects the financial services that he wants to query/process on the financial terminal. In the same time, the user needs to submit his signed authentication. The financial terminal preliminarily verifies the identity of the user and returns the verification result to the user. If the identity of the user is successfully verified, the terminal will then send the signed $< T-TYPE >$ template data message to the blockchain node. When the message is received, the node will verify the signature of the user and check whether the type of the data is $< Token >$. Here, $< Token >$ is the token circulated on the financial blockchain, which can be regarded as a kind of simple template data during transfer transaction. The built-in $< Token >$ template type is solidified in blockchain, and the data with this type can be directly run by blockchain node. If the type of the data is $< Token >$, the data is stored in local $< Token >$ sub-tree collection. Here, $< Token >$ sub-tree is a Merkle tree that specially records the data of $< Token >$. The node will broadcast this message to other nodes. If the type of the data is not $< Token >$, the node will send the data to oracle machine in the oracle group. When the oracle machine receives the message, it will conduct a smart contract and judge whether the type of message has been authenticated by the financial regulator agency, i.e., the authoritative institutions like bank, the insurance company, and securities and exchange commission. If the message is successfully verified, the oracle machine will sign the message, and return the message to the blockchain node. The node will store the data into the corresponding sub-tree collection, and broadcast the message in the blockchain network. After an epoch, the nodes will mine a new block. The financial terminal will calculate the hash value according to the submitted data, perform simplified payment verification (SPV), and return the SPV result to the user. In this way, a single chain is able to deal with multiple data types. In the following, we introduce more details about the implementation process.

The Public-private Key Generation by User

A user first generates a private key P_u locally (P_u can be considered as a point on the curve secp256k1 [Johnson et al., 2001]), and also randomly selects a point G_u on the curve secp256k1, which satisfies $z_u G_u = O$. Here, O is an infinity point, the prime number z_u that meets safety requirements is the order of G_u. Then $K_u = P_u \times G_u$ can be calculated, and the public key (z_u, K_u) corresponds to P_u can be achieved. Here, "$\times$" denotes the multiplication defined on the elliptic curve. In this way, a public-private key can be generated by the user.

Template Data Processing on Financial Terminal

(1) Digest Calculation with the SHA256 Algorithm

The user submits the template data D and the public-private key to the financial terminal. The SHA256 algorithm [Courtois et al., 2014] is used for digest calculation, which is a well-known security hash algorithm that can generate a 256-bit digest. By employing the SHA256 algorithm, we can get the digest dig_u of the template data D.

(2) Digest Signature by the User

When the user applies for a certain service, the financial terminal signs the provided service according to the private key P_u provided by the user. Specifically, it selects a random number a_u within the elliptic curve, and calculates the point on the elliptic curve, i.e., (x_u, y_u), with a_u and G_u

$$(x_u, y_u) = a_u G_u \tag{1}$$

Then, the signed message (r_u, s_u) by P_u can be calculated as

$$r_u = x_u \bmod z_u \tag{2}$$

$$s_u = (dig_u + r_u P_u) a_u^{-1} \bmod z_u \tag{3}$$

Here, "mod" denotes the modulus operation. This process of digest signature by the user is defined in the function digestSign(public key, digest).

The financial terminal extracts the template type T (the abbreviation of $< T-TYPE >$) in template data D. At the

same time, in order to prevent the distributed denial of service attack (DDoS) [Essaid et al., 2019], the user is required to pay deposit $Bond$, and the deposit amount is set by financial regulator agency. Then the financial terminal can obtain the integration package $Package_u$ for transfer by

$$Package_u = ((r_u, s_u), (z_u, K_u), D, T, Bond) \quad (4)$$

This package will be sent to the blockchain node. In this way, the process of interop signature between the user and financial terminal is finished.

Message Processing by Blockchain Node

When the blockchain node receives the data package $Package_u$ sent by the financial terminal, the smart contract locks the $Bond$ of the user, and verifies the user signature. The blockchain node then calculates the digest $dig_{u'}$ with the SHA256 algorithm. After that, the signed message can be calculated with Eqs. (5) and (6)

$$(x_{u'}, y_{u'}) = \left(\left(s_u^{-1} dig_{u'}\right) \bmod z_u\right) G_u + \left(\left(s_u^{-1} r_u\right) \bmod z_u\right) K_u \quad (5)$$

$$r_{u'} = x_{u'} \bmod z_u \quad (6)$$

If $r_u = r_{u'}$, the signature verification can be regarded to be successful. Afterward, the template type T is checked that whether it is built-in $< Token >$ template type or not. Here, $< Token >$ is the token circulated on the financial blockchain, which can be regarded as a kind of simple template data during transfer transactions. The built-in $< Token >$ template type is solidified in blockchain, and blockchain can directly run this type of data. If T is $< Token >$ template type, then $(< Token >, D, Bond)$ will be stored in local $< Token >$ sub-tree collection. And the blockchain node will use Flooding routing protocol [Rahman et al., 2004] to broadcast $Package_u$ to other nodes. If T is not the $< Token >$ template type, the node uses its own private key P_b to sign the integration package and obtains the package data $Package_b$

$$Package_b = ((r_b, s_b), (z_b, K_b), D, T) \quad (7)$$

where (z_b, K_b) is the public key corresponding to P_b, (r_b, s_b) is the signed message by the blockchain node on template data D with P_b. The signature process is the same as the process digestSign(public key, digest). Financial blockchain is an alliance chain, and thus nodes need to be authorized by the financial regulator agency before they can be added. In this secure scenario, it is hardly for nodes to act maliciously. Even in case if the nodes act maliciously, they will be eliminated once they are found out by the financial regulator agency. Thus, in order to save the communication resource, there is no need to attach $Bond$ to data package. The blockchain node will send data package $Package_b$ to oracle machine, and temporarily save the template data $(< T-TYPE >, D, Bond)$ locally. If the oracle machine does not make response or replies that the template data does not exist, the blockchain node deletes the data $(< T-TYPE >, D, Bond)$ and deducts user's deposit.

Algorithm 1. Smart contract of extended function embedding

```
1. Input: FRA signature, integrity hash, and FRA template package
2. ecrecoverDecode(FRA signature)
3. require(SHA256(FRA template package)==integrity hash)
4. require(FRA template package==SHA256(FRA template package))
5. split FRA template package
6. if (checkTemplateTable(template table) && checkTemplateScript(run-validation script)) then
7.     create < T-TYPE > table in oracle group database
8.     store run-validation script in oracle group
9. else
10.    throw
11. end
```

Template Message Processing by the Oracle Group

The oracle system consists of an oracle group and the financial regulator agency. Oracle group runs a smart contract that communicates with the financial regulator agency. When a new service is added to financial terminals, the corresponding $< T-TYPE >$ template data will be added to the oracle group through a smart contract. The oracle machines then deploy corresponding smart contract on themselves.

Algorithm 1 summarizes the smart contract of extended function embedding that is deployed in the oracle machines, which deals with the process that the financial regulator agency adds the function corresponding to $< T-TYPE >$ into the oracle group. The smart contract will verify the signature of the financial regulator agency (FRA), template integrity hash, and the integrity of the FRA template package, including template table and run-validation script, using the functions checkTemplateTable() and checkTemplateScript().

Algorithm 2 summarizes the smart contract of multi-type data processing that is deployed in the oracle machines, which deals with the $< T-TYPE >$ template data to be verified that is sent by the blockchain node. When the smart contract receives the $Package_b$ sent by blockchain node, it first verifies the signature of blockchain node. If the signature verification is successful, T in $Package_b$ will then be compared with $< T-TYPE >$ that is stored in a local oracle group database. If the type of $< T-TYPE >$ corresponding to T is found, it is then considered to meet the condition that the $< T-TYPE >$ data requires to be processed by the smart contract. Afterwards, the smart contract uses the oracle machine's private key P_o to sign D and obtain $Package_o$ by

$$Package_o = ((r_o, s_o), (z_o, K_o), D, T) \quad (8)$$

Algorithm 2. Smart contract of multi-type data processing

1. Input: $Package_b$
2. split $Package_b$
3. require(digestSign((z_b, K_b) , SHA256(D))== (r_o, s_o))
4. if (search($< T-TYPE >$)) **then**
5. conduct run-validation script
6. oracle machine signs D
7. **return** $Package_o$
8. else
9. throw
10. end

where (z_o, K_o) is public key corresponding to P_o, (r_o, s_o) is the signed message by the oracle machine on D with P_o.

Processing of Data Returned by the Oracle Group with the Blockchain Node

After receiving $Package_o$, the blockchain node first verifies the signature of the oracle machine. After successfully verifying that the oracle machine's signature is true, $Package_o$ is then parsed into the $(< T-TYPE >, D, Bond)$ template data for storage, and this data will be compared with temporary template data stored locally in the blockchain node to check whether they are consistent or not. If they are consistent, this template data will be stored in the $< T-TYPE >$ sub-tree collection, and this blockchain node will broadcast it to other nodes through the Flooding protocol.

After a certain period of time, the blockchain node will sort the template data stored locally in all sub-tree collections in the order of timestamps. Taking $< Token >$ as an example, the node will calculate the corresponding hash value $HToken\_DATA_i, i \in \{1, \cdots, I\}$ of the sorted data $Token\_DATA_i$ in the sub-tree collection as follows

$$HToken\_DATA_i = \mathrm{SHA256}(Token\_DATA_i) \quad (9)$$

where I is the number of $< Token >$ template data. Taking $HToken\_DATA_i$ and $HToken\_DATA_{i+1}$ as the input of SHA256 algorithm, their parent node's hash value $HToken\_DATA_{i,i+1}$ can be calculated as

$$HToken\_DATA_{i,i+1} = \mathrm{SHA256}(Token\_DATA_i, Token\_DATA_{i+1}) \quad (10)$$

In this way, the hash value of each two sibling nodes is used to calculate the hash value of their parent node, and after several iterations, the corresponding $< Token >$ sub-tree Merkle root $HToken\_Root$ is finally produced. The calculation of other sub-tree roots $H < T-TYPE > \_Root_j, j \in \{1, \cdots, J\}$ corresponding to $< T-TYPE >$ remains the same as that corresponding to $< Token >$, where J is the number of categories of $< T-TYPE >$. Then, the hash values of the sub-tree roots are stored in the block header.

After the block is produced, the conditions, i.e., the block is mined and template data sent by user is inside the block body, will be met for the smart contract to return bonds, and the corresponding amount of bonds will then be added to the user's account balance.

SPV of the Financial Terminal

After waiting for a period of time, the user can utilize the financial terminal to send a SPV request to the blockchain node. The financial terminal will then calculate the hash value of the triple $(T, D, Bond)$ by

$$HPackage_u = \mathrm{SHA256}(T, D, Bond) \quad (11)$$

and search the corresponding hash value $HPackage_u$ in the sub-tree corresponding to T. After the hash value is found, the financial terminal performs SPV. If verification is successful, the transaction will be successfully recorded in the block, which marks the completion of a template data transaction.

4 Simulation Experiment

In this section, we describe the experimental environment, and provide the results and the corresponding analysis.

4.1 Experimental Environment

In the experiment, the blockchain is deployed locally. Ten personal computers are used to simulate the nodes of the blockchain, a Raspberry Pi (light node) is used to simulate the multi-functional financial terminal, and the multi-group alliance chain that is built on the FISCO-BCOS platform is used as the oracle group. The detailed configurations are shown in Table 2. The main code of our model and the corresponding application demo is released on the webpage https://github.com/sebf2020/ijcai20.

Device	Configuration
Financial terminal	Raspberry Pi 3B+
Blockchain nodes	CPU: E3-1257 v5 Graphics card: NAVIDIA Geforce GTX 1660
Oracle machine	Server CPU: E5-2620 v2 Smart contract is built on the FISCO-BCOS v2.0 platform

Table 2. Experimental configuration.

4.2 Experimental Results and Analysis

In the experiment, we define two kinds of data types. The first type is the built-in $< Token >$ template data type, which records the basic transfer transactions in the blockchain, including the public key of the transfer sender, the public key of the transfer receiver, and the amount of transfer. The other type is $< T-TYPE >$ template data type that is extended by the oracle machine. Three $< T-TYPE >$ template data types are defined, including Insurance, CrossborderPayment, and ElectronicIdentity. The Insurance template data type is responsible for recording healthy financial insurance data,

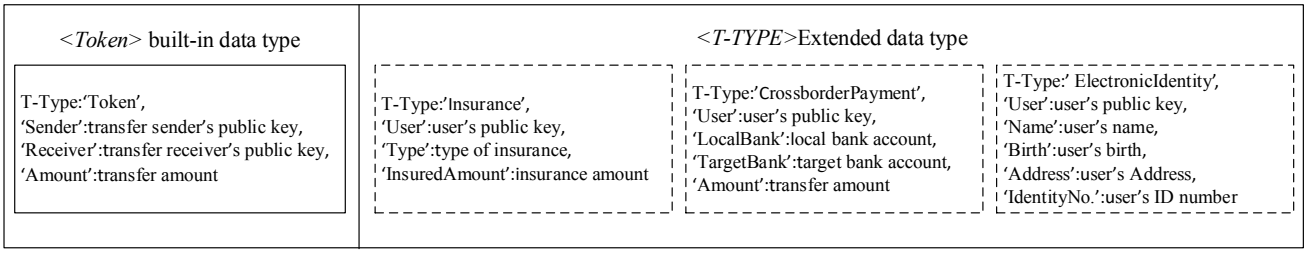

Figure 3. Template data types.

which includes the public key of the user (policy holder), the type of insurance, and the insurance amount, making it easy for the insurance company to call the data quickly. The CrossborderPayment template data type is responsible for recording cross-border transfer transaction data, which includes the public key of the cross-border transfer user, the public key of the local bank account, the public key of the target bank account, and the transfer amount, making it easy for cross-border banks to quickly process user's transfer. The ElectronicIdentity template data type is responsible for recording the user's electronic ID information data, including the public key of user's account, user's real name, birth date, the user's address, and electronic ID number, which enables regulators to make quick calls to this type of data. The above $< Token >$ template data type and three $< T-TYPE >$ template data types are shown in Fig. 3.

In order to verify the effectiveness of our SEBF model, we compare our model with the traditional financial single-chain without extension function. Specifically, in the experiment, we utilize four kinds of blockchains with different settings, including: (1) a single-chain without extension function, only having the built-in $< Token >$ template data type, which is named **B-T**; (2) a single-chain with extension function containing the oracle machine and financial regulator agency, having the built-in $< Token >$ and the Insurance template data types, which is named **B-TI**; (3) a single-chain with extension function containing the oracle machine and financial regulator agency, having the built-in $< Token >$, the Insurance, and the CrossborderPayment template data types, which is called **B-TIC**; (4) a single-chain with extension function containing the oracle machine and financial regulator agency, having the built-in $< Token >$, the Insurance, the CrossborderPayment, and ElectronicIdentity template data types, which is called **B-TICE**.

For a given fixed number of template data, our SEBF model can make single-chain be able to deal with multiple types of data. By using the index established with the data type information, the efficiency of data search is improved by rapidly searching data of a specific type. The simplified payment verification (SPV) time is an intuitive reflection of data search efficiency, and it is also a suitable indicator of the financial blockchain processing delay that directly affects the users' experience. Thus, we use SPV verification time as an evaluation measure. The time complexity of the SPV process is $2N\log_2(N)$, where N is the total number of template data. When there are multiple types of data, the time complexity of the SPV process of our model is $\sum_{j=1}^{J} 2N_j \log_2(N_j)$, where $N_j, j = 1,\cdots,J$ denotes the number of data of the j th data type, and $\sum_{j=1}^{J} N_j = N$. It is obvious that $\sum_{j=1}^{J} 2N_j \log_2(N_j) < 2N\log_2(N)$. Thus, theoretically, our SEBF model can improve the verification efficiency of SPV.

We randomly generate template data and input the data to the above four kinds of blockchains. In practice, $< Token >$ is an important template type in the financial blockchain, and $< Token >$ template data should account for a large proportion of data in the financial blockchain. Therefore, we generate more $< Token >$ template data than that of other types in the second to the fourth kind of blockchains. For example, the $< Token >$ template data accounts for 70 percent of the total input data set, and the other types of data constitutes the remaining 30 percent of the input data set equivalently. For the blockchain B-T, we randomly select a $< Token >$ template data in the input data set to perform SPV and record the verification time. For the second to the fourth kind of blockchains, we randomly select a $< Token >$ template data and a single data for each of the other types in the input data set to perform SPV and record the verification time corresponding to each data type. And then, we compute the average verification time across multiple data types for each kind of blockchain. Specifically, in the experiment, the size of the input data set is investigated in the range of [500,3000] with the step of 500. For each case of data set size, we record the optimal and the worst verification time and report the mean value for each kind of blockchain. In order to make it easy to input data, the mining epoch is set as 10 minutes.

Fig. 4 shows the comparison of SPV verification time of four kinds of blockchains when the $< Token >$ data accounts for 70 percent of the input data set. With the increase of the data set size, the verification time of four kinds of blockchains shows the increasing trend. It is also obvious that the verification time of the second to the fourth kind of blockchains is shorter than that of B-T, i.e., the traditional financial blockchain only containing the $< Token >$ template data type, for a specific data set size. Furthermore, for a specific data set size, B-TIC needs less verification time than B-TI, and B-TICE needs the least verification time.

Table 3 tabulates the average SPV verification time across different data set sizes of four kinds of blockchains under different percentages of $< Token >$ data in the input data set. We can see that with the increase of the percentage of

$< Token >$ data in the input data set, the verification time of four blockchains experiences a slight increase. And in all cases of percentages, B-TI, B-TIC, and B-TICE cost less verification time than that of B-T, and B-TICE costs the least verification time.

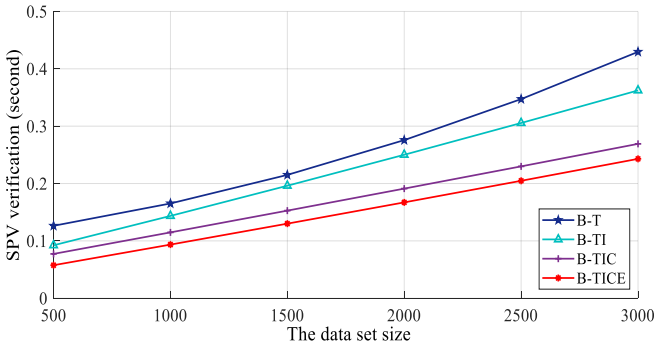

Figure 4. Illustration of SPV verification time of four kinds of blockchains when $< Token >$ template data accounts for 70% of the total input data set.

Percentage of input data set as $< Token >$ data (%)	B-T	B-TI	B-TIC	B-TICE
50	-	0.203	0.160	0.112
60	-	0.215	0.165	0.120
70	-	0.216	0.172	0.134
80	-	0.241	0.177	0.139
90	-	0.242	0.181	0.142
100	0.259	-	-	-

Table 3. Average SPV verification time (second) of blockchains under different percentages of $< Token >$ data.

As a summary, our model is an effective single-chain based extension solution, and it can obviously reduce the SPV verification time as compared with the traditional financial blockchain. With the increase of type of template data, the SPV verification time of our model will be further reduced.

5 Conclusion

In this paper, we design a single-chain based extension model of blockchain for fintech (SEBF). We specially design a four-layer system model architecture. Based on the oracle group and financial regulator agency, we provide a novel data structure extension mechanism and design the corresponding smart contracts, such that the single-chain is able to store multiple types of template data, e.g., Insurance, CrossborderPayment, and ElectronicIdentity. Experimental results indicate that the SEBF model improves the verification efficiency of SPV as compared with traditional financial blockchain, and with the increase of the type of template data, the verification efficiency can be further improved.

For the future work, we will apply our SEBF model in a real bank scenario with Tonghe Chain[3], where a built-in token type of data is owned and new business is developed aperiodically that generates new data types, for further validating the effectiveness and efficiency of the proposed model.

Acknowledgments

This work was supported by the National Key R&D Program of China (Nos. 2017YFB1401300, 2017YFB1401302), National Natural Science Foundation of China (Nos. 61702280, 61902194), Outstanding Youth of Jiangsu Natural Science Foundation (No. BK20170100) , Key R&D Program of Jiangsu (No. BE2017166), Natural Science Foundation of Jiangsu Province (No. BK20170900), Natural Science Foundation of the Jiangsu Higher Education Institutions of China (No.19KJB520046), National Postdoctoral Program for Innovative Talents (No. BX20180146), China Postdoctoral Science Foundation (No. 2019M661901), Jiangsu Planned Projects for Postdoctoral Research Funds (No. 2019K024), Six Talent Peak Projects in Jiangsu Province, CCF-Tencent Open Fund WeBank Special Funding (No. CCF-WebankRAGR20190104), NUPT DingShan Sholar Project, the Innovation and Entrepreneurship Projects of Jiangsu Province and NUPTSF (No. NY219132).

References

[Abdur Rahman et al., 2019] Md Abdur Rahman, George Loukas, Syed Maruf Abdullah, Areej Abdu, Syed Sadiqur Rahman, Elham Hassanain, Yasmine Arafa. Blockchain and IoT-based secure multimedia retrieval system for a massive crowd: sharing economy perspective. International Conference on Multimedia Retrieval, 404-407, 2019.

[An et al., 2019] Baoyi An, Mingjun Xiao, An Liu, Guoju Gao, Hui Zhao. Truthful crowdsensed data trading based on reverse auction and blockchain. International Conference on Database Systems for Advanced Applications, 292-309, 2019.

[Back et al., 2014] Adam Back, Matt Corallo, Luke Dashjr, et al. Enabling blockchain innovations with pegged sidechains.URL:http://www.opensciencereview.com/papers/123/enablingblockchain-innovations-with-pegged-sidechains, 2014.

[Borkowski et al., 2019] Michael Borkowski, Marten Sigwart, Philipp Frauenthaler, Taneli Hukkinen, Stefan Schulte. DeXTT: Deterministic cross-blockchain token transfers. arXiv:1905.06204, 2019.

[Buterin, 2014] Vitalik Buterin. Ethereum: A next-generation smart contract and decentralized application platform. White Paper, 2014.

[Chen et al., 2019] Tian-yi Chen, Lan Zhang, Shi-cong Zhang, Zi-long Li, Bai-chuan Huang. Extensible cross-modal hashing. International Joint Conference on Artificial Intelligence, 2109-2115, 2019.

[3] http://nhpcc.jqsoft.net/

[Courtois et al., 2014] Nicolas T. Courtois, Marek Grajek, Rahul Naik. Optimizing sha256 in bitcoin mining. International Conference on Cryptography and Security Systems, 131-144, 2014.

[Dai et al., 2020] Weiqi Dai, Chunkai Dai, Kim-Kwang Raymond Choo, Changze Cui, Deiqing Zou, Hai Jin. SDTE: A secure blockchain-based data trading ecosystem. IEEE Transactions on Information Forensics and Security, 15: 725-737, 2020.

[Dai et al., 2019] Hong-Ning Dai, Zibin Zheng and Yan Zhang. Blockchain for Internet of things: A survey. IEEE Internet of Things Journal, 6(5): 8076-8094, 2019.

[Deng et al., 2018] Liping Deng, Huan Chen, Jing Zeng, Liang-Jie Zhang. Research on cross-chain technology based on sidechain and hash-locking. International Conference on Edge Computing, 144-151, 2018.

[Du et al., 2017] Mingxiao Du, Xiaofeng Ma, Zhe Zhang, Xiangwei Wang, Qijun Chen. A review on consensus algorithm of blockchain. IEEE International Conference on Systems, 2567-2572, 2017.

[Essaid et al., 2019] Meryam Essaid, DaeYong Kim, Soo Hoon Maeng, Sejin Park, Hong Taek Ju. A collaborative DDoS mitigation solution based on Ethereum smart contract and RNN-LSTM. Asia-Pacific Network Operations and Management Symposium, 1-6, 2019.

[Eyal et al., 2016] Ittay Eyal, Adem Efe Gencer, Emin Gün Sirer, Robbert van Renesse. Bitcoin-ng: A scalable blockchain protocol. {USENIX} Symposium on Networked Systems Design and Implementation, 45-59, 2016.

[Herlihy, 2018] Maurice Herlihy. Atomic cross-chain swaps. ACM Symposium on Principles of Distributed Computing, 245-254, 2018.

[Jiang and Wu, 2019] Suhan Jiang, Jie Wu. Bitcoin mining with transaction fees: a game on the block size. IEEE International Conference on Blockchain, 107-115, 2019.

[Johnson et al., 2001] Don Johnson, Alfred Menezes, Scott Vanstone. The elliptic curve digital signature algorithm (ECDSA). International Journal of Information Security, 1(1): 36-63, 2001.

[Kabra et al., 2020] Naman Kabra, Pronaya Bhattacharya, Sudeep Tanwar, Sudhanshu Tyagi. Mudrachain: Blockchain-based framework for automated cheque clearance in financial institutions. Future Generation Computer Systems, 102: 574-587, 2020.

[Lee et al., 2017] Greg Lee, Josh Lavin, Daniel Larimer, Thomas Cox, Nathan Hourt, William Prioriello. EOS. IO technical white paper. URL: https://github.com/EOSIO/Documentation, 2017.

[Lin and Liao, 2017] Iuon-Chang Lin, Tzu-Chun Liao. A survey of blockchain security issues and challenges. IJ Network Security, 19(5): 653-659, 2017.

[Liu et al., 2018] Han Liu, Chao Liu, Wenqi Zhao, Yu Jiang, Jiaguang Sun. S-gram: towards semantic-aware security auditing for ethereum smart contracts. ACM/IEEE International Conference on Automated Software Engineering, 814-819, 2018.

[Liu et al., 2019] Mengting Liu, Fei Richard Yu, Yinglei Teng, Victor C. M. Leung, Mei Song. Performance optimization for blockchain-enabled industrial internet of things (IIoT) systems: A deep reinforcement learning approach. IEEE Transactions on Industrial Informatics, 15(6): 3559-3570, 2019.

[Lombrozo et al., 2015] Eric Lombrozo, Johnson Lau, Pieter Wuille. Segregated witness (consensus layer). Bitcoin Core Develop. Team, Technical Report BIP, 2015.

[Merkle, 1987] Ralph C. Merkle. A digital signature based on a conventional encryption function. Conference on the Theory and Application of Cryptographic Techniques, 369-378, 1987.

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

[Rahman et al., 2004] Ashikur Rahman, Wlodek Olesinski, Pawel Gburzynski. Controlled flooding in wireless ad-hoc networks. International Workshop on Wireless Ad-Hoc Networks, 73-78, 2004.

[Sarpatwar et al., 2019] Kanthi Sarpatwar, Venkata Sitaramagiridharganesh Ganapavarapu, Karthikeyan Shanmugam, Akond Rahman, Roman Vaculin. Blockchain enabled AI marketplace: The price you pay for trust. IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2857-2866, 2019.

[Singh et al., 2020] Amritraj Singh, Kelly Click, Reza M. Parizi, Qi Zhang, Ali Dehghantanha, Kim-Kwang Raymond Choo. Sidechain technologies in blockchain networks: An examination and state-of-the-art review. Journal of Network and Computer Applications, 149: 102471, 2020.

[Sompolinsky and Zohar, 2013] Yonatan Sompolinsky, Aviv Zohar. Accelerating bitcoin's transaction processing. Fast money grows on trees, not chains. IACR Cryptology ePrint Archive, issue 881, 2013.