LoC - A New Financial Loan Management System based on Smart Contracts

Hao Wang[a,*], Chaonian Guo[b], Shuhan Cheng[b]

[a]*Department of Computer Science, NTNU, Gjøvik, Norway*
[b]*Fujian Rural Credit Union, Wusi Road 517, Fuzhou, Fujian, China*

Abstract

Current financial loan management systems are usually deployed in a single-service mode, also the transactions are not transparent and traceable to most of the roles participating in the process. Their data privacy protection mechanisms are not robust enough facing various cyber attacks. To overcome these challenges, we propose loan on blockchain (LoC), a novel financial loan management system based on smart contracts over permissioned blockchain Hyperledger Fabric. We use the Chinese poverty alleviation loan as the case study. We design a digital account model for the transfer of assets between centralized and decentralized ledgers; and propose locking and unlocking algorithms for smart contracts. We introduce digital signature and oracle to protect the data privacy. Performance evaluations on chaincode and unlocking codes show that our system is applicable in the real financial loan setting.

Keywords: Blockchain, financial loan, smart contracts, decentralized ledger technology

1. Introduction

Information technology has played a key role in the development of modern financial and banking services [1]. To maintain the efficiency and convenience of financial services, banks rely on large and complicated information

*Corresponding author
Email addresses: `hawa@ntnu.no` (Hao Wang), `guochaonian@fjnx.com.cn` (Chaonian Guo), `chengshuhan@fjnx.com.cn` (Shuhan Cheng)

systems and databases to conduct their business today. With the accelerating pace of developments in the fields of mobile devices, the modern analysis methods of big data, the shifting of data into the cloud, the application of AI and blockchain technology, more and more banks begin to apply Fintech [2] to their information and business systems.

China has announced the continuous poverty alleviation program for its rural areas [3], and more than CN¥246 billion is issued for poverty alleviation loan in the end of 2016 [4]. There are few varieties of poverty alleviation loan in the financial industry. Because the service is in single service mode, the loan amount is small for each customer, and the risk management mechanism is not robust, the cost of loan services of financial institutions is higher than other loan products in support of the customer groups. The development of the business requires more investments from all aspects of the local government, financial institution, guarantee agency and regulator. More importantly, there should be a transparent and secure management system providing full services, especially for the rural areas which lack wholesome financial infrastructure.

Also, the poverty alleviation loan is one of the product portfolios of Fujian Rural Credit Union (FRCU) for this program. Poverty alleviation loan, is a special loan product of FRCU set up for poor people lived in rural areas. The roles participated in the loan include bank, customer, financial department of government, civil affairs department of government and regulator of government. The application procedure of this loan should walk through all these roles, and the auditing terms of different roles vary a lot. It is a long time for the applier to get the refund for loan interest. Currently, this loan is maintained in the ordinary loan process management system. However, existing systems are suffering from some limitations that prevent customers from achieving most out of the value of poverty alleviation loan. The major issues are: (i) the management system is centralized and deployed in single service mode, which slows down the efficiency of information exchanges for the loan; (ii) there is not an easy way to trace the data updating process and prevent data tampering (especially for the supervisor), since there are many roles in poverty alleviation loan and the business process is long; (iii) there is not an effective protection for the customer data privacy facing cyber attacks (man-in-the-middle attack [5], DoS attack, fraud, etc.).

The major contributions of this paper are summarized as follows:

- We propose a novel design of poverty alleviation loan management system LoC based on smart contracts over permissioned blockchain;

- We design the digital account model for LoC to ensure asset transfer between traditional bank and LoC;
- Based on the permissioned blockchain we introduce the semantical smart contracts (chaincode) locking and unlocking mechanism for the automatic evaluation/execution of transactions for LoC;
- Event, oracle and signature are introduced for smart contracts to keep the validity of poverty alleviation loan and data privacy.

In section 2, we introduce the related work. Subsequently, in section 3, we describe the overview of LoC, including components, roles and chaincode locking, unlocking process. Section 4 discusses the detailed digital account design, implementation of locking and unlocking process of chaincode, construction of event oracle and the data privacy in LoC. Evaluation on the performance of chaincode and unlocking code is carried out in section 5. Section 6 concludes the paper with an extensive research agenda.

2. Related Work

Blockchain technology is first introduced by Bitcoin as a distributed bookkeeping system to prevent double-spending [6]. Blockchain is an encrypted, distributed database/transaction system where all the peers share information in a decentralized, trusted and secure manner. The ledger holds a complete, and all-agreed transaction record. It has led to an increasing interest in the technical community for using the underlying decentralized ledger of transactions to solve other interesting problems, such as the fairness in information exchange [7, 8]. A number of large industrial companies, such as IBM, Microsoft, Intel and Tencent are currently investing in exploiting blockchain technology in order to enrich their product portfolios. In recent years, there have been a number of blockchain frameworks proposals appearing, such as Ripple [9], Ethereum [10], Corda [11], Hyperledger [12] and Hyperledger Fabric [13], among others.

There are two types of blockchains, one is public or permissionless blockchain (such as Bitcoin) and the other is permissioned blockchain [14, 15] (such as Hyperledger Fabric). The main differences between them are the privacy and consensus algorithms.

In a permissionless blockchain anyone can participate without a specific identity in the process of block verification to create consensus and also create smart contracts. Permissionless blockchains typically involve a native cryptocurrency and often use consensus based on "proof of work" (PoW) and economic incentives. One major issue of permissionless blockchain is the high latency of block generation, which is caused by the expensive PoW process. To make the block generation faster, two different strategies are proposed: "proof of stake" (PoS) and permissioned blockchain. Permissioned blockchain restricts the actors who can contribute to the consensus of the system state. So, it provides a way to secure the interactions among a group of entities that have a common goal but which do not fully trust each other, such as businesses that exchange funds (finance), goods (supply chain), or information (public service). By relying on the identities of the peers, a permissioned blockchain can use traditional Byzantine-fault tolerant (BFT), RAFT, or Paxos consensus.

Table 1 summarizes most of the application scenarios of DLT. In poverty alleviation loan, there may be participants who can only send and receive assets, some others who have (exclusive) rights to validate transactions, and a third group (such as regulator) which has the rights to read only. Therefore, we did not choose the complex and resource-intensive PoW consensus mechanism of permissionless DLT systems. On the contrary, we selected the permissioned blockchain Hyperledger Fabric for the design and implementation of LoC.

Table 1: The application scenarios of DLT

Industry	**Scenarios**
Finance	Payments, Clearing and Settlements, Insurance, Crowdfunding
Supply Chain	Supply Chain Finance, Supply Chain Traceability
Public Service	Intellectual Property Protection, Sharing Economy, File Management
IoT	Traceability, Anti-counterfeiting and Identification of Goods
Public Charity	Public Donation System, Tracking and Management of Money

Core components of decentralized ledger technology (DLT) are widely used in information technology for payments, clearing and settlement for central banks. The world's leading financial institutions are stepping up to explore the blockchain of landing applications: the Bank of Canada has launched project Jasper to evaluate domestic interbank payments settlement and released CAD-coin [16]. In Europe, the European Central Bank and the Bank of Japan have launched a joint research project on DLT [17]. The Monetary Authority of Singapore has announced the successful conclusion of a proof-of-concept project to conduct domestic inter-bank payments using DLT [18]. The People's Bank of China has begun to carry out research on DLT and digital currency [19].

Smart contracts enabled in DLT are virtual agreements encoded on the network that are automatically executed based on logical conditions [20]. The automatic execution is an important attribute for smart contracts [21]. Institutions also focus on the data security and privacy of smart contracts in DLT [22, 23, 24, 25]. Platforms running user-defined smart contracts and executing user-supplied transactions on their objects are also carried out [26]. In recent years, the financial industry is also moving towards expressing financial agreements via financial smart contracts [27], which serves as precise notations for expressing financial agreements among parties.

Efforts are made to handle identity, transaction, debt information and non-performing loans [28, 29, 30, 31, 32]. For poverty alleviation loan, there is no much related work. To address these shortcomings, we propose LoC, a novel poverty alleviation loan management system based on smart contracts leverages the DLT. LoC uses the locking and unlocking of smart contracts to deal with transactions semantically. In LoC, the digital account is used, which makes asset transfer between decentralized ledger and traditional bank possible. Moreover, digital signature and oracle mechanism are introduced to ensure the data privacy and loan assets security. The DLT also makes supervision much easier.

3. Overview of LoC and Locking & Unlocking of Chaincode

In this section, we describe the overview of LoC, and discuss the locking and unlocking of chaincode.

3.1. Components in LoC

There are several components in LoC as shown in Fig. 1:

- Peer – A peer runs smart contracts called chaincode, receives ordered state updates in the form of blocks from the ordering service, and maintains the world state and the ledger. Many peers could be deployed for a single role, and each peer can join in different channels.

- Fabric SDK Node – The roles communicate with a peer through Fabric SDK Node.

- Channels – The system provides channels to roles and peers, offering a broadcast service for messages containing transactions.

- Ordering Service – Ordering Services generate blocks and order the transactions in sequence.

- Membership Service – It provides the membership enrolling service for all roles.

- Roles – Each participant in the system has one role.

- Blockchain – Each block typically contains a cryptographic hash of the previous block, a timestamp and transaction data.

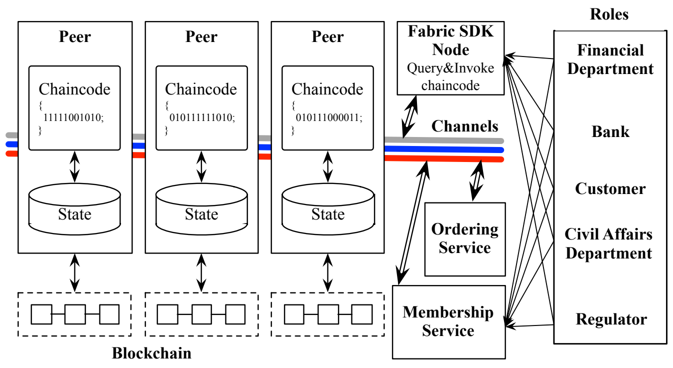

Figure 1: Overview of LoC

3.2. Roles in LoC

In the poverty alleviation loan management system, LoC needs to support all the different types of roles that participate in the process:

- Financial Department – $\mathcal{FD}$. The financial department is the user who represents the government in the poverty alleviation program. Role of financial department in the system is to check the auditing result of identity and application information of the customer for poverty loan from the civil affairs department, and to provide the fund for subsidy of interest to customer via bank according to the governmental poverty alleviation program.

- Bank – $\mathcal{B}$. The bank is the most important part in the system which provides loan to customer. Role of bank is the issuance of loan, including the verification of the identity and application information of the customer for loan, the provision of loan funds to customer, the calculation and collection of loan interests.

- Customer – $\mathcal{C}$. Customer is the issuance target of the poverty alleviation loan. He provides identity and other information to the civil affairs department, and makes an application for the loan to bank.

- Civil Affairs Department – $\mathcal{CAD}$. The civil affairs department audits the identity from the customer and loan application of the customer, then transfers the auditing result to financial department.

- Regulator – $\mathcal{R}$. The regulator is responsible for the monitoring of fund flow among each role and the risk during the loan business. He can inspect the ledger to get a full picture of the system, and challenge any party and transaction if he finds something wrong during the loan business.

3.3. The Locking and Unlocking of Chaincode

3.3.1. Digital signature

A digital signature is a mathematical scheme for demonstrating the authenticity of digital messages or documents [33]. A valid digital signature gives a recipient reason to believe that the message was created by a known sender (authentication), that the sender cannot deny having sent the message (non-repudiation), and that the message was not altered in transit (integrity). Digital signature is a standard element of most cryptographic protocol suites, and are commonly used for software distribution, financial transactions, contract management software, and in other cases where it is important to detect forgery or tampering. In this work, we sign the event and oracle result of the event in the poverty alleviation loan business with a digital signature, which can guarantee the integrity of transactions and prevent any repudiation between applicant and auditor.

3.3.2. The locking and unlocking of chaincode

Transactions of LoC relies on two types of codes: a chaincode and an unlocking code. The chaincode is a type of smart contracts, in which it specifies the requisite conditions that must be met to do some operations. We call the generation procedure of chaincode a locking process. The unlocking code is a piece of code that "solves," or satisfies, the conditions placed on the chaincode and allows the transaction to be executed, which is called the unlocking process.

We define the chaincode space as $\mathcal{CS}$, and let the object $o \in \mathcal{CS}$ like, "*A applies to do something with the grant of B*", where "*do something*" is the atomic *Operations* (such as transfer money, repayment and account query) for object o, the whole object is called *event*, A and B denote roles $r \in \mathcal{RO} = \{\mathcal{FD}, \mathcal{B}, \mathcal{C}, \mathcal{CAD}, \mathcal{R}\}$ that participates in the *event*. Then, the locking process is to translate the object o into chaincode cc semantically. In the loan issuance process, the chaincode locking process could be combined with business flow pre-configured to the LoC system. We construct a 3-tuple ($Operations$, $event$, $Public\ Keys$) to represent the chaincode cc, where *event* should be appended with all the signatures of roles in A for *event*, and $Public\ Keys$ are the keys of all roles in B who grant the *event*. The chaincode will be in effect only when the 3-tuple is valid and complete.

The forming of unlocking code for the chaincode can also be pre-configured in LoC within the loan issuance business. After B receives the *event*, he will give an oracle result to the *event*. The oracle result from any role in B could be $Pass$ or $Failed$ according to the information he receives. Hence, the unlocking code cc should be a 2-tuple ($event$, $Signatures^{ora}$) in unlocking code space $\mathcal{US}$, where *event* is from A, and $Signatures^{ora}$ are the digital signatures of oracle results for *event*. It makes sense when this happens in poverty alleviation loan of real finance business, especially in which there are multiple players participating in the scenario.

Currently, we have shown how to generate chaincode and the corresponding unlocking code. For the unlocking process, we design a dual-stack evaluation mechanism for chaincode transactions by reduction, in which the transaction in chaincode will be executed automatically triggered by the unlocking code. To illustrate the mechanism, we present an informal walkthrough for unlocking process of one chaincode example. Assume that we have generated a chaincode *cc* like this:

$$cc = ((OP_1,\ OP_2),\ event,\ (P_1,\ P_2,\ P_3)), \tag{1}$$

and the corresponding unlocking code *uc*:

$$uc = (event,\ (Sig_1^{ora},\ Sig_2^{ora},\ Sig_3^{ora})). \tag{2}$$

All the items of the codes in (1) and (2) are pushed into respective stacks from left to right, and the unlocking process of the chaincode *cc* is described in Fig. 2.

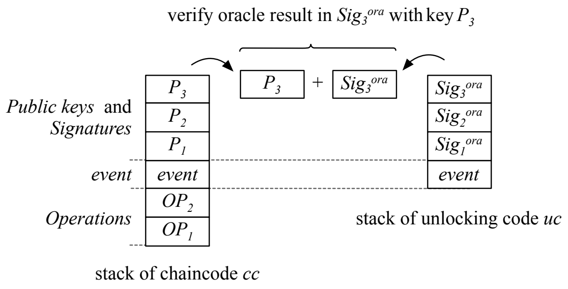

Figure 2: Dual-stack unlocking of chaincode *cc* via *uc*: in the process, all the items are popped out from the stacks in sequence. Then we use the public key in chaincode to verify the signature in the unlocking code and get the oracle results of the event. In case all the oracle results are *Pass* and the *event* in both stack equals, the *Operations* in chaincode is popped out and executed. Thus, the unlocking process of chaincode *cc* completes.

The dual-stack evaluation mechanism guarantees the automatic execution of chaincode and the integrity of transactions, and also prevents the repudiation and man-in-the-middle attack. We summarize this process as event-driven locking and unlocking of chaincode.

4. Design and Implementation of LoC

We now describe in detail the design of digital account for LoC, the implementation of algorithms for chaincode generation, unlocking code generation and the corresponding unlocking process based on business flow. Also, the construction of event oracle and the data privacy for LoC is analyzed.

4.1. Account Design for LoC

In traditional banks, there is an account for every customer keeping the assets of real-world value. These assets are often issued by real world entities. For LoC, we need to cope with this and design the digital account carefully. We record the existence and exchanges of digital assets without issuing them in LoC. The asset value of LoC come from the traditional accounts. LoC targets for non-public settings, in which roles in $\mathcal{RO}$ spin up a network to manage the poverty alleviation loan and trade assets among each other. If other banks or users that are not in $\mathcal{RO}$ want to join in the network, they should get authenticated.

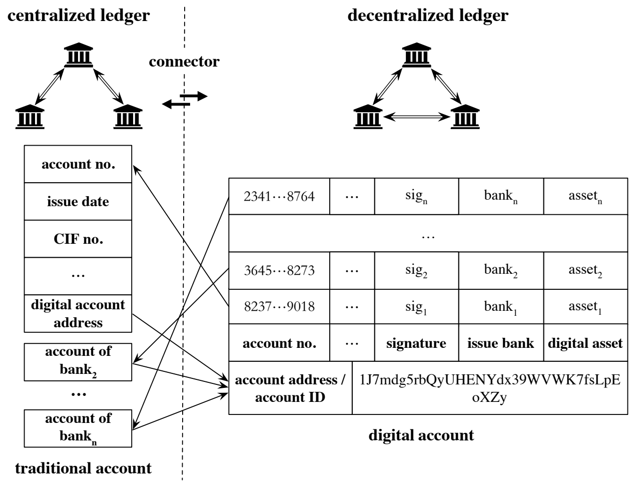

Figure 3: Digital account design of LoC (CIF – customer information)

Fig. 3 shows the digital account structure for LoC. We add an account address (or account ID, the account ID of digital account is the public key of the account owner) link to traditional account and add account number link back in digital account of each asset issued by different banks (In a traditional account, there is usually an account number). They will set up the connection between these two types of accounts, thus making the asset transfer between these two types of accounts possible. The client or customer of banks can have a global digital account $da \in \mathcal{DA}$ and many traditional accounts $tas \in \mathcal{TA}$. For digital accounts, the assets might come from different banks, and the signature of bank is appended to the asset issued by the bank. Hence, the digital assets could be differentiated from each other. For traditional accounts, all the transactions are in centralized ledger. On the other hand, in a decentralized ledger, all the transactions are in peers and each peer keeps the same copy. A connector can be set up between these two ledgers by higher institutions, such as the central bank or government (It is not the focus of this paper). In LoC, the customer can transfer funds between these two accounts.

Hereinafter, we discuss the poverty alleviation loan for the customer who has at least one traditional account ta and one digital account da, with a connection set up between ta and da. If not explicitly stated otherwise, the term "transaction" stands for the transaction inside system LoC.

4.2. Implementation of Chaincode for LoC

As mentioned in the previous section, all roles might propose an *event* $\in \mathcal{ES}$ with related atomic operation in poverty alleviation loan, such as the events shown in Table 2 (not all the events are listed).

Table 2: Events and operations in LoC

Role	**Events**	**Operations**
$\mathcal{C}$	applies for the poverty alleviation loan	transfer
$\mathcal{FD}$	repays interest for customer when loan expires	repay
$\mathcal{R}$	inspects the fund flow of transaction	inspect
$\mathcal{B}$	queries account	query
$\mathcal{CAD}$	queries the information of the customer	query

Algorithm 1 shows the generation/locking process of chaincode. Given the Roles RO_1, RO_2, *Operations* and *event*, it returns corresponding chaincode cc. The atomic *Operations* and *Signatures* of the *event* will be pushed into the chaincode in order, and finally the public keys of RO_2 are pushed into the chaincode.

In the generation process of unlocking code, the *Signatures* in the end of *event* will be verified and checked. If the signature verification passes, roles in RO_2 will judge the event they receive from RO_1 and give the oracle results of the event. The entire process can be described in Algorithm 2.

For the dual-stack unlocking process of chaincode, the chaincode stack S_{cc} and unlocking code stack S_{uc} are combined together. All the *Signatures* in S_{uc} and *Public Keys* in S_{cc} are popped out, then *Signatures* are verified with the keys in sequence and we get event oracle results. When the *event*

```
Algorithm 1: The chaincode generation algorithm
   Input: $RO_1 \subseteq \mathcal{RO}$, $Operations$, $RO_2 \subseteq \mathcal{RO}$, $event \in \mathcal{ES}$
   Output: Chaincode $cc \in \mathcal{CS}$ for $o = (RO_1, Operations, RO_2, event) \in \mathcal{ES}$
 1 $cc \leftarrow \phi$
 2 Assert($Operations$ is valid)
 3 Assert($event$ is valid)
 4 $cc$.append($Operations$)
 5 for $r \in RO_1$ do
 6     $Sig_r^{event} \leftarrow$ sign $event$ with private key of $r$
 7     $event$.append($Sig_r^{event}$)
 8 $cc$.append($event$)
 9 for $r \in RO_2$ do
10     $cc$.append($public\ key\ of\ r$)
11 return $cc$            ▶ locking success
```

is popped out, the $Signatures$ of $event$ and event itself will be checked in both stacks. After the oracle result and event verification pass, $Operations$ are popped out and executed. The unlocking process could be summarized in Algorithm 3. It could be derived that the complexity of each Algorithm is Θ(n), where n is the stack depth of the chaincode.

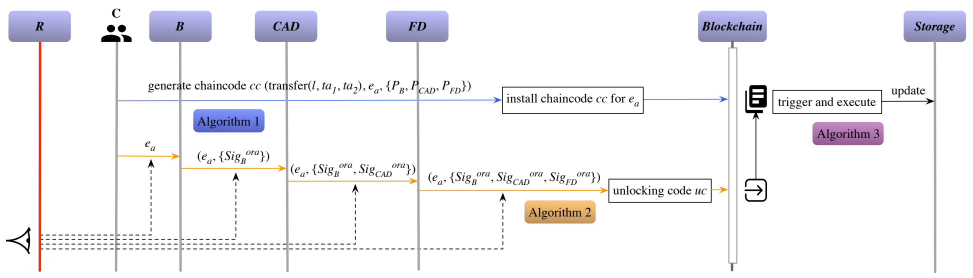
R
C
B
CAD
FD
Blockchain
Storage
generate chaincode cc (transfer(l, ta_1, ta_2), e_a, {P_B, P_{CAD}, P_{FD}})
install chaincode cc for e_a
trigger and execute
update
Algorithm 1
Algorithm 3
e_a
(e_a, {Sig_B^{ora}})
(e_a, {Sig_B^{ora}, Sig_{CAD}^{ora}})
(e_a, {Sig_B^{ora}, Sig_{CAD}^{ora}, Sig_{FD}^{ora}})
unlocking code uc
Algorithm 2

Figure 4: Business flow for chaincode, unlocking code generation and execution

To model the locking and unlocking process for loan application business event, we define the event from customer $c \in \mathcal{C}$ as $e_a = (id_c,\ l,\ d,\ other\ info,\ Sig_c)$, where id_c denotes identity of the customer c, l denotes the loan amount he applies, d denotes the loan duration, $other\ info$ denotes other customer information (such as income, asset, debt, etc.) and Sig_c is the signature of the event. Let $cc = (transfer(l,\ ta_1,\ ta_2),\ e_a,\ (P_{\mathcal{B}},\ P_{\mathcal{CAD}},\ P_{\mathcal{FD}}))$ be the chaincode for the loan application event, where $transfer(l,\ ta_1,\ ta_2)$ means the operation of transferring money from digital account ta_1 to ta_2 with amount

Algorithm 2: The unlocking code generation algorithm

```
Input: $RO_1 \subseteq \mathcal{RO}$, $event \in \mathcal{ES}$, $RO_2 \subseteq \mathcal{RO}$
Output: Unlocking code $uc \in \mathcal{UC}$ for chaincode of $event$
1  $uc \leftarrow \phi$
2  $Sigs \leftarrow \phi$
3  Assert($event$ is valid)
4  for $item \in event$ do
5      if $typeof(item)$ is $Signature$ then
6          $Sigs$.append($item$)
7  for $s \in Sigs$, $r \in RO_1$ do
8      Assert(verify $s$ with public key of $r$)
9  $uc$.append($event$)
10 for $r \in RO_2$ do
11     $result \leftarrow$ oracle($event$) of $r$
12     $s \leftarrow$ sign $result$ with private key of $r$
13     $uc$.append($s$)
14 return $uc$          ▶ unlocking code successfully generated
```

l, $P_{\mathcal{B}}$, $P_{\mathcal{CAD}}$ and $P_{\mathcal{FD}}$ are the public keys of bank, civil affairs department and financial department (the poverty alleviation loan application should be passed to bank, civil affairs and financial department and audited by them). And the unlocking code is also constructed according to the business flow semantically. Let $uc = (e_a,\ (Sig_{\mathcal{B}}^{ora},\ Sig_{\mathcal{CAD}}^{ora},\ Sig_{\mathcal{FD}}^{ora}))$ denotes the unlocking code, where $Sig_{\mathcal{B}}^{ora}$, $Sig_{\mathcal{CAD}}^{ora}$ and $Sig_{\mathcal{FD}}^{ora}$ represent the oracle results for the loan application event e_a with signatures. As shown in Fig. 4, the generation of chaincode and unlocking code could be configured to LoC. After chaincode is installed to the peer, it will be triggered and evaluated when the unlocking code is generated and proposed to peer. Thus, the operation results in the digital asset value of the customer increasing by l. Finally, the world state is updated in peers.

Other events in Table 2 or not shown in the table also follow these three algorithms. One difference between the events is that the business flow is specific for each event. For example, event "$\mathcal{FD}\ repays\ interest\ for\ customer$" will execute a query to the loan information issued by the bank before, then $\mathcal{FD}$ gives an oracle result, finally the bank repays fund from digital account of $\mathcal{FD}$ to the account of $\mathcal{B}$, and the chaincode should be $(\{query(loan),\ repay(interest,\ t_{\mathcal{FD}},\ t_b)\},\ e_r,\ P_{\mathcal{FD}})$. The other difference is that some of their operations will trigger the update of world state (such as transfer, repay)

Algorithm 3: Dual-stack unlocking algorithm for chaincode

```
Input: Chaincode stack $S_{cc}$, unlocking code stack $S_{uc}$, $R \subseteq \mathcal{RO}$
Output: Return code of evaluation result for chaincode $cc \in \mathcal{CS}$
$Sigs_{cc} \leftarrow \phi$
$Sigs_{uc} \leftarrow \phi$
while $S_{cc}$ is not Null and $S_{uc}$ is not Null do
    $item_c \leftarrow S_{cc}$.pop()
    $item_u \leftarrow S_{uc}$.pop()
    switch $typeof(item_c)$ do
        case Public Key do
            oracle($item_u$) $\leftarrow$ verify $item_u$ with key $item_c$
            if oracle($item_u$) is Pass then
                oracle check pass
            else
                return $1 - oracle\ check\ failed$ ▶ unlocking failed
        case event do
            for $i_c \in item_c$, $i_u \in item_u$ do
                if typeof($i_c$) is Signature && typeof($i_c$) is Signature then
                    $Sigs_{cc}$.append($i_c$)
                    $Sigs_{uc}$.append($i_u$)
            for $s_c \in Sigs_{cc}$, $s_u \in Sigc_{uc}$, $r \in R$ do
                Assert(verify $s_c$ with public key of $r$)
                Assert(verify $s_u$ with public key of $r$)
            if $item_c$ equals $item_u$ then
                event check pass
            else
                return $2 - event\ check\ failed$ ▶ unlocking failed
        otherwise do
            return $3 - type\ error$ ▶ unlocking failed
while $S_{cc}$ is not Null do
    $op \leftarrow S_{cc}.pop()$
    if $typeof(op)$ is Operation then
        execute the operation $op$
    else
        return $3 - type\ error$ ▶ unlocking failed
return $0 - success$ ▶ unlocking success
```

while others will not (such as query, inspect).

4.3. The Oracle of Event

In cryptography, oracle is used to make arguments for the security of cryptographic protocols. For oracle, there will be an input and an output. If an input is submitted repeatedly to the oracle, the outputs are always the same. We introduce oracle mechanism with signature for event auditing in LoC to prevent the counterfeit auditing.

Let function f denotes the oracle process:

$$f:\ event\ \rightarrow\ result \tag{3}$$

The roles in $RO_{ora}\ \in\ \{\mathcal{FD},\ \mathcal{B},\ \mathcal{CAD}\}$ will give oracle results independently for specific *event* in LoC. The oracle function for RO_{ora} is $\{f_{\mathcal{FD}},\ f_{\mathcal{B}},\ f_{\mathcal{CAD}}\}$. For $e\ \in event$, we define function f as:

$$f(e) = \begin{cases} Pass & if\ (\odot\ t(i))\ \times\ ex\ =\ 1,\ \forall\ i\ \in\ e \\ Failed & otherwise, \end{cases} \tag{4}$$

where, $t(i)$ is:

$$t(i) = \begin{cases} 1 & if\ i\ meets\ term_i \\ 0 & otherwise. \end{cases} \tag{5}$$

The operation "$\odot$" in equation (4) indicates multiplication mod 2, ex indicates the exception to the event e of role in RO_{ora}. Exception ex always should be 1, and sometimes be 0 if the grant role finds some evidence shows that the loan should not be issued (such as the customer in the blacklist). Some of the terms for role in RO_{ora} of event loan application e_a could be shown in Table 3. All the loan terms could be combined into the oracle functions in this form.

After oracle result $f(e)$ of event e is calculated, role in RO_{ora} will sign the result. Then, the final result will be passed to next role for auditing.

4.4. Data Privacy of LoC

In LoC, users have higher levels of privacy and security compared to the traditional poverty alleviation loan management system. First, we design the LoC system based on permissioned blockchain Hyperledger Fabric, who contains the "Membership Service" and "Ordering Service". The authentication mechanism of the services and other security characteristics makes

Table 3: Terms of the loan for different roles

e_a	$\mathcal{B}$	$\mathcal{CAD}$	$\mathcal{FD}$
id_c	$== identity$	$== identity$	$== identity$
l	$<= loan_{limit}$	NA	NA
d	$<= month_{max}$	NA	NA
$income$	$>= income_{min}$	$<= income_{max}$	NA
$asset$	$>= asset_{min}$	$<= asset_{max}$	NA
$debt$	$<= debt_{max}$	NA	NA
$issue$	0 $issued\ before$ 1 $otherwise$	0 $issued\ before$ 1 $otherwise$	0 $issued\ before$ 1 $otherwise$

the untrusted access and non-deterministic execution from untrusted chaincode DoS attacks impossible in order-execute architectures [13]. Secondly, the messages and events are all appended with signatures, which makes the man-in-the-middle attack easier to happen. Thirdly, all events will be audited and given the oracle results, which means even if the event is tampered, the oracle result will be $Failed$.

With this design, untrusted entities would not be able to get useful information by observing the ledger because they would not have the access of "Membership Service". LoC would be deployed in the Internet zone, always with DMZ (demilitarized zone), of whole banking system. Various of security infrastructure are always deployed in this zone, to keep data away from attack, fraud and virus. These infrastructure include IDS (Intrusion Detect System), firewall, anti-virus, anti-fraud and content filtering system.

5. Evaluation

In this section, we evaluate the generation, execution of three events for chaincode and unlocking code in docker. Also, the scalability of the system is evaluted. The first event is the application for the loan, the second one is the application for the refund, and the final is query of the loan. Table 4 shows the detailed information for the evaluation. These scenarios will trigger the API of Hyperledger Fabric SDK: Invoke, Invoke and Query respectively. The operation Transfer will transfer money from one digital account to another. The operation Query will query the loan information from bank. There are 5 peers on behalf of $\mathcal{C}$, $\mathcal{B}$, $\mathcal{CAD}$, $\mathcal{FD}$, $\mathcal{R}$ and 1 orderer in the container docker. Orderer provides ordering service, who provides a shared communication channel to peers, offering a broadcast service for messages containing transactions.

Table 4: Detailed information of three events

Operation	Event	Business flow	API
Transfer	1: Application of loan	$\mathcal{C} \to \mathcal{B} \to \mathcal{CAD} \to \mathcal{FD}$	Invoke
Transfer	2: Application of refund	$\mathcal{C} \to \mathcal{B} \to \mathcal{FD}$	Invoke
Query	3: Query of loan	$\mathcal{C} \to \mathcal{B}$	Query

After the channel has been created, all the peers join in the channel and listen to the messages broadcasted from the orderer. The chaincode and unlocking code described by go language, which are the output of Algorithms 1 and 2, for all events are installed to these peers in advance. There is no unlocking code for event 3 Query. Then we use fabric-go-sdk and Beego, an open source framework for web application, to send request: Invoke and Query to Fabric. Invoke is the interface for account transfer transaction, and Query is for result querying transaction. Figure 5 shows the installing time for each event (the unit is second). It could be seen from the figure that all the chaincode and unlocking code could be installed within 0.25 second.

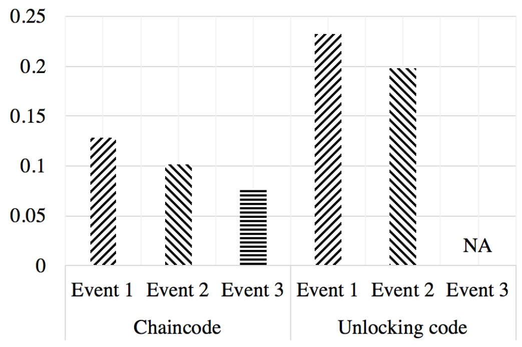

Figure 5: The installing time of chaincode and unlocking code for each event

For the evaluation of chaincode execution, the request of all transactions is increased by 10. We compose all the transactions by the percentages of 40%, 40%, 20% respectively for event 1, event 2 and event 3. It can be derived from Figure 6 that as the load of request rises, the consumption of

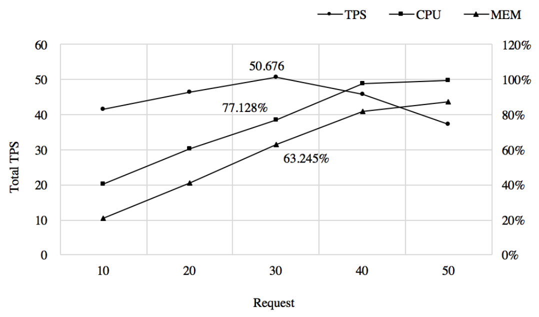

Figure 6: The evaluation results of locking code execution

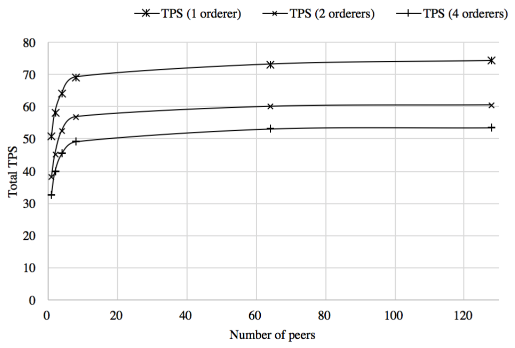

Figure 7: The scalability for large number of peers

CPU and memory for smart contract clearing is increasing too. The total TPS – transaction per second, which is calculated by $request/response$, is growing up gradually and reaching the maximum value 50.676 at request 30. After that, the value drops down to 45.872 at request 40 because of the CPU resource is almost exhausted at 98.001%. When the request is 50, the TPS continues to drop down because of the bottleneck from CPU. At request of 30, the TPS for Transfer of event 1 is 8.2; the TPS for Transfer of event 2 is 10.782; and the TPS for Query of event 3 is 31.694. It shows that event 3 has highest value because it just reads the value state from blockchain, while the other two try to update the state.

For the scalability, we consider three scenarios: 1 orderer, 2 orderers and 4 orderers. Then the number of peers for customer increases gradually. We set the number of peers to 1, 2, 4, 8, 64, 128 respectively. For each set of peers, we could get the final highest TPS (for example, 50.676 is highest TPS for one peer). Figure 7 depicts the results for different number of peers, which shows the total TPS increases fast within 10 peers, and finally the curves get flattening. As the number of peers is large than 200 or more, we find out that the TPS will become to drop and there appears a lot of error or timeout requests. Also, we could see that it takes more time to synchronize the messages and get consensus as the number of order increases. Comparing to current poverty alleviation loan management system, which could reach 200 TPS in FRCU, LoC still need some careful optimization for scalability and performance. For the fault tolerance of LoC, research [34] shows that to tolerate f faulty orderers, the network should consist of at least $n = 3f + 1$ orderers. And this means the network requires $2f + 1$ orderers to get consensus. Also, the authorities for roles $\mathcal{FD}$, $\mathcal{B}$, $\mathcal{C}$, $\mathcal{CAD}$, $\mathcal{R}$ should always be trusted in the network.

The evaluation is carried out on a single server, while more distributed servers could be deployed to pursue higher TPS. The results indicate that our proposed scheme is extendable and applicable for LoC, therefore enabling the transaction management of poverty alleviation loan among different entities. Moreover, in the beginning phase of production, the business volumn of poverty alleviation loan of FRCU is not very large.

6. Conclusion and Future Work

Poverty alleviation loan is an important program for the poverty alleviation strategy. It is urgent to build a management system for this loan to improve the transparency, security and traceability for the business. LoC is introduced in this paper to handle this issue. We design LoC based on the permissioned blockchain, and uses locking and unlocking of smart contracts to execute the transactions automatically. In addition, event, oracle and digital signature are introduced to validate the loan business and provide data privacy. Performance evaluations on chaincode and unlocking code generation, installation and execution show that our system is applicable in the real financial loan setting.

For the future work, we plan to further elaborate on the connector between centralized and decentralized legers. This is also the main limitation of the proposed scheme, that is there is no generally accepted connector between centralized and decentralized ledgers in the market currently. Currently, FRCU is working on the PoC (proof of conception) of the project for poverty alleviation loan. There is a Fintech team consisted of three persons carrying out this project for about one year. Next step, we will deploy this design to the production system and analyze more experimental results in details.

Acknowledgment

The work described in this paper was partially supported by the National Natural Science Foundation of China under Grant No. 61672170 and No. 61871313 and the Science and Technology Planning Project of Guangdong Province under Grant No. 2017A050501035. Also, this work is partially funded by the Fujian Fumin Foundation.

[1] R. J. Shiller, The new financial order: Risk in the 21st century, Princeton University Press, 2009.

[2] P. Schueffel, Taming the beast: a scientific definition of fintech.

[3] L. Meng, Evaluating china's poverty alleviation program: a regression discontinuity approach, Journal of Public Economics 101 (2013) 1–11.

[4] C. B. Association, China banking social responsibility report, http://www.china-cba.net/do/bencandy.php?fid=43&id=16595 (accessed 2 Jan, 2019).

[5] G. N. Nayak, S. G. Samaddar, Different flavours of man-in-the-middle attack, consequences and feasible solutions, in: Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on, Vol. 5, IEEE, 2010, pp. 491–495.

[6] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system.

[7] H. Wang, H. Guo, M. Lin, J. Yin, Q. He, J. Zhang, A new dependable exchange protocol, Computer communications 29 (15) (2006) 2770–2780.

[8] H. Wang, H. Guo, Achieving fairness in wireless environment, in: Emerging Technologies: Frontiers of Mobile and Wireless Communication, 2004. Proceedings of the IEEE 6th Circuits and Systems Symposium on, Vol. 1, IEEE, 2004, pp. 117–120.

[9] Ripple, accessed 2 Jan, 2019.
URL `https://ripple.com/`

[10] G. Wood, Ethereum: A secure decentralised generalised transaction ledger, ethereum Project Yellow Paper.

[11] M. Hearn, Corda - a distributed ledger, corda Technical White Paper.

[12] Hyperledger, https://www.hyperledger.org/ (Accessed 2 Jan, 2019).

[13] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, et al., Hyperledger fabric: a distributed operating system for permissioned blockchains, in: Proceedings of the Thirteenth EuroSys Conference, ACM, 2018, p. 30.

[14] T. T. A. Dinh, R. Liu, M. Zhang, G. Chen, B. C. Ooi, J. Wang, Untangling blockchain: A data processing view of blockchain systems, IEEE Transactions on Knowledge and Data Engineering 30 (7) (2018) 1366–1385.

[15] M. Vukolić, Rethinking permissioned blockchains, in: Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts, ACM, 2017, pp. 3–7.

[16] R. Garratt, Cad-coin versus fedcoin, http://www.r3cev.com/s/cad-coin-versus.pdf (accessed 2 Jan, 2019).

[17] Bank of Japan, ECB and the bank of Japan launch a joint research project on distributed ledger technology, http://www.boj.or.jp/en/announcements/release_2016/rel161207a.htm/ (accessed 2 Jan, 2019).

[18] Monetary Authority of Singapore, MAS working with industry to apply distributed ledger technology in securities settlement and cross border payments, http://www.mas.gov.sg/News-and-Publications/Media-Releases/2017/MAS-working-with-industry-to-apply-Distributed-Ledger-Technology.aspx (accessed 2 Jan, 2019).

[19] Z. Xu, Q. Yao, The proposal of the digital bill trading system, China Finance 17 (2016) 31–33.

[20] A. Savelyev, Contract law 2.0: Smart contracts as the beginning of the end of classic contract law, Information & Communications Technology Law 26 (2) (2017) 116–134.

[21] B. Egelund-Müller, M. Elsman, F. Henglein, O. Ross, Automated execution of financial contracts on blockchains, Business & Information Systems Engineering 59 (6) (2017) 457–467.

[22] J. Liang, W. Han, Z. Guo, Y. Chen, C. Cao, X. S. Wang, F. Li, DESC: enabling secure data exchange based on smart contracts, Science China Information Sciences 61 (4) (2018) 049102.

[23] G. Zyskind, O. Nathan, et al., Decentralizing privacy: Using blockchain to protect personal data, in: Security and Privacy Workshops (SPW), 2015 IEEE, IEEE, 2015, pp. 180–184.

[24] M. Corrales, P. Jurcys, G. Kousiouris, Smart contracts and smart disclosure: Coding a gdpr compliance framework.

[25] H. Halpin, M. Piekarska, Introduction to security and privacy on the blockchain, in: Security and Privacy Workshops (EuroS&PW), 2017 IEEE European Symposium on, IEEE, 2017, pp. 1–3.

[26] M. Al-Bassam, A. Sonnino, S. Bano, D. Hrycyszyn, G. Danezis, Chainspace: A sharded smart contracts platform, arXiv preprint arXiv:1708.03778.

[27] G. W. Peters, E. Panayi, Understanding modern banking ledgers through blockchain technologies: Future of transaction processing and smart contracts on the internet of money, in: Banking Beyond Banks and Money, Springer, 2016, pp. 239–278.

[28] C. Huang, G. Min, Y. Wu, Y. Ying, K. Pei, Z. Xiang, Time series anomaly detection for trustworthy services in cloud computing systems, IEEE Transaction on Big Data, DOI: 10.1109/TBDATA.2017.2711039.

[29] M. Mainelli, M. Smith, Sharing ledgers for sharing economies: an exploration of mutual distributed ledgers (aka blockchain technology).

[30] S. DHAR, I. BOSE, Smarter banking: Blockchain technology in the indian banking system.

[31] Y. Ma, Y. Wu, J. Ge, J. Li, An architecture for accountable anonymous access in the internet-of-things network, IEEE Access 6 (2018) 14451–14461.

[32] X. K. Wang, L. T. Yang, X. Xie, J. Jin, M. J. Deen, A cloud-edge computing framework for cyber-physical-social services, IEEE Communications Magazine 55 (11) (2017) 80–85.

[33] W. Diffie, M. Hellman, New directions in cryptography, IEEE transactions on Information Theory 22 (6) (1976) 644–654.

[34] H. Sukhwani, J. M. Martínez, X. Chang, K. S. Trivedi, A. Rindos, Performance modeling of pbft consensus process for permissioned blockchain network (hyperledger fabric), in: Reliable Distributed Systems (SRDS), 2017 IEEE 36th Symposium on, IEEE, 2017, pp. 253–255.