A Blockchain Framework for Insurance Processes

Mayank Raikwar*, Subhra Mazumdar†, Sushmita Ruj†,
Sourav Sen Gupta†, Anupam Chattopadhyay*, and Kwok-Yan Lam*
*School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798
Email: {mraikwar,anupam,kwokyan.lam}@ntu.edu.sg
†Indian Statistical Institute, 203 Barrackpore Trunk Road, Kolkata 700 108, India
Email: subhra.mazumdar1993@gmail.com,sush@isical.ac.in,sg.sourav@gmail.com

***Abstract*—We design a distributed platform with blockchain as a system service for supporting transaction execution in insurance processes. The insurance industry is heavily dependent on multiple processes between transacting parties for initiating, maintaining and closing diverse kind of policies. Transaction processing time, payment settlement time and security protection of the process execution are major concerns. Blockchain technology, originally conceived as an immutable distributed ledger for detecting double spending of cryptocurrencies, is now increasingly used in different FinTech systems to address productivity and security requirements. The application of blockchain in FinTech processing requires a deep understanding of the underlying business processes. It supports automated interactions between the blockchain and existing transaction systems through the notion of smart contracts. In this paper, we focus on the design of an efficient approach for processing insurance related transactions based on a blockchain-enabled platform. An experimental prototype is developed on Hyperledger fabric, an open source permissioned blockchain design framework. We discuss the main design requirements, corresponding design propositions, and encode various insurance processes as smart contracts. Extensive experiments were conducted to analyze performance of our framework and security of the proposed design.**

I. Introduction

Blockchain technology offers a novel way for constructing secure distributed systems. Initially designed as a system service for detecting double spending in cryptocurrency systems, blockchain is widely applicable to many business applications where there is a requirement of trust among distributed parties. At a high level, a *blockchain* is a distributed ledger service implemented by multiple participants, with each of them storing a local copy of the ledger. The ledger's consistency is achieved by certain consensus protocols involving all participants. Blockchain systems may choose from a wide range of consensus protocols depending on the trust model. Immutability of the ledger is achieved by a combination of cryptographic primitives and open distribution of the ledger [1].

In common practical settings, business rules are encoded as smart contracts [2], [3] that require access to the blockchain. In terms of insurance processes, a smart contract can be written to register clients interested in purchasing policies offered by the company, enable them to file claims, and receive refunds. The smart-contract based distributed ledger will inherently prevents any sort of fraudulent transactions if the requests or actions do not conform to the rules of the contracts.

Existing insurance systems have achieved a certain degree of automated processing. However, due to the lack of a single trusted source of state information of different transactions, they still suffer from performance bottlenecks. The generic insurance systems require manual interactions across different transaction processes, hence resulting in slow processing, and lengthy payment settlement time. Due to similar issues, the insurance industry is also facing the challenges of detecting claim frauds [4], which can otherwise be handled efficiently using blockchain-enabled smart contracts [5], [6].

This paper addresses the performance and security concerns of insurance processes by designing a blockchain based solution (similar to [7]). The main purpose of the blockchain-based solution for the insurance industry is — (a) to automate and speed up business processes in the insurance industry, from client registration and policy issuance to claims handling, (b) to make fraud-detection easier using decentralized digital repository, (c) to make client data confidential and accessible only to the authorized parties, (d) to reduce administrative and operational costs, and (e) to enable regulators and auditors to detect suspicious transaction patterns and market behaviors.

Our contribution in this paper is the design of a blockchain framework for insurance use cases to offer fine-grained access control by specifying different set of endorsers for each smart contract. In reality, different insurance policies have different set of endorsers, which is mimicked by creating different smart contracts for different policies in our model. We used Hyperledger fabric to implement our insurance blockchain framework. Extensive experiments have been conducted by scaling up the network to test the robustness of the system, and a detailed analysis of the latency has been carried out with a varying set of parameters. We also investigated the relationship between transaction latency and network size.

II. Proposed Model

The idea behind our model is to implement the processes of an insurance company as smart contracts, and place the contracts in a blockchain-enabled distributed platform, for both execution of the contracts and storage of the results.

A. Entities in the Model

Primary entities in our model are – the *Client*, who is covered by insurance and requests for insurance policies, submits claim requests, and receive refunds, and the *Agent*, who acts on behalf of the client, and processes the client's requests to the blockchain network. An agent can have multiple clients.

978-1-5386-3662-6/18/$31.00 ©2018 IEEE

B. Components of the Model

The major components of our design include – a distributed *blockchain* ledger $\mathcal{B}$ that logs execution results of all transactions, a *database* $\mathcal{DB}$ (optionally encrypted) that maintains the insurance contracts and transaction results of all the clients in `(Key, Value)` format, a set of *endorsers* E_{SC} (subset of peer nodes) who verify the transaction conditions of smart contract SC, a set of *orderers* O who order the transactions chronologically and create the transaction blocks, a set of *validators* V who validate and store the transaction blocks to the blockchain ledger. Certain cryptographic algorithms are used to authenticate users and to provide the access control.

C. Framework for Insurance

Our insurance framework consists of *assets* that enable the exchange of almost anything with monetary value over the network. The framework has smart contracts that govern the rules for transactions. A block is created in the insurance blockchain network when the peer nodes in the validator set V run consensus over a set of transaction results (key-value pairs) [8]. Each smart contract contains an endorsement (or verification) logic, which specifies under what conditions a transaction can be executed by the smart contract. The endorsement logic is executed by a set of endorsers E_{SC} (may be particular to a smart contract SC) who access the blockchain to determine whether contract conditions are satisfied.

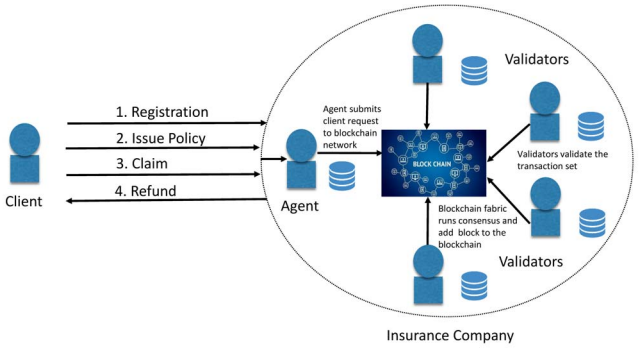

Fig. 1. System model of Insurance blockchain framework

D. Model for Insurance

We consider a simple scenario where the main processes (transactions) are standard insurance operations like — client registration, policy assignment, paying premium, claim submission, processing refunds etc. The blockchain maintains execution and results of each transaction and ensures that the clients do not falsely accuse the insurance company, and that the insurance company is accountable for all services that it provides. Figure 2 shows the basic workflow of the framework.

Each smart contract SC_j has a set of endorsers E_{SC_j} who endorse the transactions for that contract. The term *object* refers to the attributes of the client or the policy. The structure of an object is defined during the instantiation of the smart contract. We use function γ to create an object from its attributes, and function f to generate composite keys (primary) from the ID(s). The purpose of the key is to retrieve specific object(s) corresponding to the ID(s) from the database. We also use partial composite keys (not primary) in our framework through which a set of objects may be retrieved from the database. We discuss each of the contracts in detail, as follows.

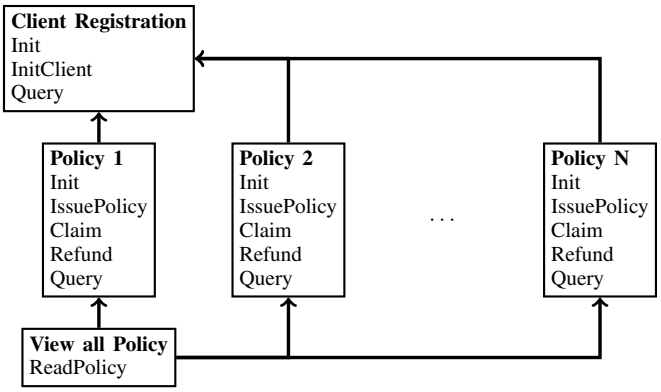

Fig. 2. Smart contracts for Insurance Processes

Client Registration smart contract registers the clients to the insurance system. During its initialization (Algorithm 1), a structure of client object (O_c) is created in the database $\mathcal{DB}$ (as a key-value pair). It contains client attributes like client unique id id_c as the key and other client attributes as values.

Algorithm 1: Client Registration : Init

Input: Peer Nodes : $\{Peer_0, Peer_1, ..., Peer_i\}$
Endorsement Policy : OR$(Peer_0, ..., Peer_i)$

$S_{O_c} \leftarrow (id_c, name_c, age_c, gender_c, contact_c, history_c)$;
Create the structure S_{O_c} in database $\mathcal{DB}$;

To register a client (Algorithm 2), composite key CK_c is created by an agent, and client object O_c is created using CK_c.

Algorithm 2: Client Registration : InitClient

Input: Client structure S_{O_c} and agent id id_A

$CK_c \leftarrow f(id_A, id_c)$;
$O_c \leftarrow \gamma(S_{O_c})$;
Store (CK_c, O_c) in $\mathcal{DB}$;

To retrieve specific client details (Algorithm 3), an insurance agent A has to create a composite key CK_c, as follows.

Algorithm 3: Client Registration : Query

Input : Agent id id_A and Client id id_c
Output: Client object O_c

$CK_c \leftarrow f(id_A, id_c)$;
Search for CK_c in $\mathcal{DB}$;
Retrieve corresponding O_c if it exists, or return Error;

In case an agent requires to retrieve all his/her clients, he/she may generate a partial composite key PK_A using only his/her own id id_A, and query the database $\mathcal{DB}$ with the partial key.

Policy smart contract contains policy issuance, claims, refunds etc. During its initialization (Algorithm 4), policy and policy-client structures $S_{O_p}, S_{O_{pc}}$ are created in database $\mathcal{DB}$, where amt, acc and $date$ are amount claimed, claim acceptance indicator (yes or no) and claim submission date, respectively.

Algorithm 4: Policy : Init

Input: Peer Nodes : $\{Peer_0, Peer_1, ..., Peer_i\}$
Endorsement Policy : OR$(Peer_0, ..., Peer_i)$

$S_{O_p} \leftarrow (id_p, Name_p, premium_p, reimburse_p, term_p)$;
$S_{O_{pc}} \leftarrow (id_p, id_c, amt, acc, date)$;
Create the structures S_{O_p} and $S_{O_{pc}}$ in database $\mathcal{DB}$;

In policy issuance process (Algorithm 5), client c chooses policy P (id id_p) from the available policies, and submits a premium $premium_c$ to the agent A (id id_A). If the transaction passes all standard checks and verifications, a corresponding policy-client object O_{pc} is created and stored in the database.

Algorithm 5: Policy : IssuePolicy

Input: $id_A, id_c, id_p, premium_c$

Query $\mathcal{DB}$ with id_p to check if O_{pc} already exists;
Query client smart contract SC_c to check if client c with id id_c is registered to agent A with id id_A;
Check if client premium matches policy premium;
$CK_{pc} \leftarrow f(id_p, id_c, id_A)$;
$O_{pc} \leftarrow \gamma(id_p, id_c, 0, Yes, date)$;
Strore (CK_{pc}, O_{pc}) in database $\mathcal{DB}$;

To process a claim (Algorithm 6), client c submits his credentials to the respective agent A. Upon verficiation of all necessary conditions, the refund is initiated accordingly. Parameter acc indicates whether then claim is accepted.

Algorithm 6: Policy : Claim

Input: $id_A, id_c, id_p, reimburse_c$

$CK_{pc} \leftarrow f(id_p, id_c, id_A)$;
Query $\mathcal{DB}$ using CK_{pc} to check if O_{pc} exists;
If object O_{pc} exists, check acc in O_{pc};
if $acc = Yes$ **then**
  if $amt + reimburse_c \leq reimburse_p$ **then**
    Refund$(id_A, id_c, id_p, reimburse_c)$;
  end
  else
    Refund$(id_A, id_c, id_p, reimburse_p - amt)$;
    $acc \leftarrow No$, update acc in O_{pc};
  end
end

Refund process is initiated from the claim process. During refund (Algorithm 7), client c's total claimed amount amt in the policy-client object O_{pc} is updated in database $\mathcal{DB}$.

Algorithm 7: Policy : Refund

Input: $id_A, id_c, id_p, reimburse_k$ from claim

$CK_{pc} \leftarrow f(id_p, id_c, id_A)$;
Query $\mathcal{DB}$ using CK_{pc} to check if O_{pc} exists;
Update $amt = amt + reimburse_k$ in O_{pc};

Agent A may retrieve details regarding his/her clients who have purchased a specific policy id_p by querying the database $\mathcal{DB}$ with a partial composite key PK_{Ap}, created by id_A and id_p. This essentially retrieves $\{O_{pc_i}\}_{i \in \{0,...,N\}}$ from $\mathcal{DB}$. An agent can also retrieve the information about any individual policy provided by the insurance company using the Query routine (Algorithm 8) in the policy smart contract.

Algorithm 8: Policy : Query

Input : id_p
Output: O_p

Search for id_p in $\mathcal{DB}$;
Retrieve corresponding O_p if it exists, or return Error;

View all Policy smart contract is an open (without particular authorization) contract that may be used to find the information about all the policies existing in the database of the company, by calling individual smart contracts associated with different policies. In this smart contract, an Agent, a Client, or even a prospective client in the network, may invoke the ReadPolicy routine without any specific access control permission.

E. Transactions in the Framework

In the proposed blockchain network, client c submits transaction request to agent A. The request consists of smart contract method and client attributes needed for the method to execute. The transaction is signed by agent A and further endorsed by the endorsers of that specific smart contract. Upon validation, agent A submits the transaction to the ordering nodes O to chronologically order the transactions. Then, the peer nodes run the core consensus routine with all received transactions, and appends the new records to the blockchain.

III. Security of Proposed Framework

The security of our blockchain framework relies on the realistic assumptions that – (a) the channel is secure during message transmission, and (b) the underlying blockchain network is secure. Table I further depicts how various malicious entities may affect our insurance network, as well as corresponding preventions. Note that the Client is not a part of the insurance blockchain network. Detailed security analysis is beyond the scope of this short paper, and will be a part of the full version.

TABLE I
MALICIOUS ENTITIES, THREATS, AND INBUILT PREVENTIONS

Malicious	Potential Threat	Inbuilt Prevention
Peer	(a) Modify/Delete Client data (b) Send wrong endorsement	(a) Consensus protocol (b) Endorsement policy
Auditor	Wrong auditing result	Consensus protocol

IV. PROTOTYPE AND EXPERIMENTS

Extensive experiments have been performed on the network to figure out the robustness of proposed insurance framework. The chaincode for smart contracts was developed in Golang v1.8 and deployed on hyperledger fabric v1.0.0-beta [9], [10]. The experiments were carried out on a system with a Hexa-core Intel Xeon E5-1650 v2 (3.50 GHz) processor and 16 GB RAM, running Ubuntu 16.04 (64 bit). We computed the *confirmation time* of a set of transactions while varying the number of peer nodes in the network. Results visualized as Figures 3 and 4 depict the relation between the number of peers and the corresponding time taken for confirming N transactions, while Table II presents the basic description and parameters of our blockchain setup for these experiments.

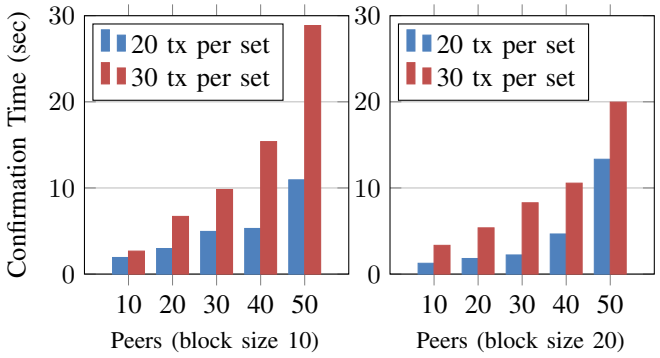

Fig. 3. Confirmation time Vs No. of Peers (Batch Timeout : 2sec)

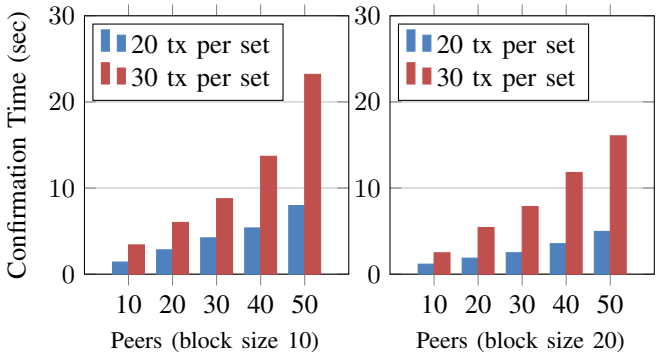

Fig. 4. Confirmation time Vs No. of Peers (Batch Timeout : 4sec)

Note that the network size is directly proportional to the confirmation time. It is clearly noticeable that more the number of nodes, more will be the confirmation time, i.e., slower will be the network. This is caused by the requirement for more number of endorsements and more number of validations.

TABLE II
EXPERIMENTAL PARAMETERS

Number of Transactions	Block Size	Batch Timeout	Consensus Protocol	Number of Iterations
20, 30	10, 20	2, 4 (sec)	Solo	20

Different consensus algorithms [11], [12] result in different confirmation times, and the same happens with different endorsement policies (for example, AND instead of OR will considerably increase the confirmation time).

V. CONCLUSION AND FUTURE SCOPE

This paper proposes a blockchain-based framework for implementing insurance transaction processes as smart contracts. Experiments conducted to study the scalability clearly showed the parameters used during blockchain creation should be chosen carefully, as they have a direct effect on the network latency. Though the database is currently not encrypted, it can be encrypted with fine-grained access control. In our model, each smart contract has its own set of endorsing peers, and this can be extended even to the transaction level, to enable separate set of endorsing peers for each transaction.

VI. ACKNOWLEDGMENT

This paper has been accepted for publication in Blockchains and Smart Contracts workshop (BSC'2018), as a short paper. The authors would like to thank the anonymous reviewers for their constructive criticism and detailed comments.

REFERENCES

[1] Vukolić and Marko, "Rethinking permissioned blockchains," in *Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts*, ser. BCC '17. New York, NY, USA: ACM, 2017.
[2] C. D. Clack, V. A. Bakshi, and L. Braine, "Smart contract templates: essential requirements and design options," *arXiv preprint arXiv:1612.04496*, 2016.
[3] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the internet of things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016.
[4] I. Nath, "Data exchange platform to fight insurance fraud on blockchain," in *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, Dec 2016, pp. 821–825.
[5] W. Li, A. Sforzin, S. Fedorov, and G. O. Karame, "Towards scalable and private industrial blockchains," in *Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts*. ACM, 2017, pp. 9–14.
[6] H. Watanabe, S. Fujimura, A. Nakadaira, Y. Miyazaki, A. Akutsu, and J. Kishigami, "Blockchain contract: Securing a blockchain applied to smart contracts," in *Consumer Electronics (ICCE), 2016 IEEE International Conference on*. IEEE, 2016, pp. 467–468.
[7] F. Lamberti, V. Gatteschi, C. Demartini, C. Pranteda, and V. Santamaria, "Blockchain or not blockchain, that is the question of the insurance and other sectors," *IT Professional*, vol. PP, no. 99, pp. 1–1, 2017.
[8] C. Christian, "Blockchain, cryptography, and consensus," 2017.
[9] Cachin and Christian, "Architecture of the hyperledger blockchain fabric," 2016.
[10] E. Androulaki, C. Cachin, A. De Caro, A. Kind, and M. Osborne, "Cryptography and protocols in hyperledger fabric," 6 January, 2017.
[11] L. S. Sankar, M. Sindhu, and M. Sethumadhavan, "Survey of consensus protocols on blockchain applications," in *2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS)*, Jan 2017, pp. 1–5.
[12] H. Sukhwani, J. M. Martnez, X. Chang, K. S. Trivedi, and A. Rindos, "Performance modeling of pbft consensus process for permissioned blockchain network (hyperledger fabric)," in *2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS)*, Sept 2017, pp. 253–255.