Shiping Chen
Harry Wang
Liang-Jie Zhang (Eds.)

LNCS 10974

Blockchain – ICBC 2018

First International Conference
Held as Part of the Services Conference Federation, SCF 2018
Seattle, WA, USA, June 25–30, 2018, Proceedings

ICBC
Blockchain

Springer

Shiping Chen · Harry Wang
Liang-Jie Zhang (Eds.)

Blockchain – ICBC 2018

First International Conference
Held as Part of the Services Conference Federation, SCF 2018
Seattle, WA, USA, June 25–30, 2018
Proceedings

Springer

Editors
Shiping Chen
CSIRO Computational Informatics
Marsfield, NSW
Australia

Liang-Jie Zhang
Kingdee International Software Group Co., Ltd
Shenzhen
China

Harry Wang
University of Delaware
Newark, DE
USA

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-319-94477-7 ISBN 978-3-319-94478-4 (eBook)
https://doi.org/10.1007/978-3-319-94478-4

Library of Congress Control Number: 2018947343

LNCS Sublibrary: SL4 – Security and Cryptology

Printed on acid-free paper

This Springer imprint is published by the registered company Springer International Publishing AG
part of Springer Nature
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

The International Conference on Blockchain (ICBC) aims to provide an international forum for both researchers and industry practitioners to exchange the latest fundamental advances in the state-of-the-art technologies and best practices of blockchain, as well as emerging standards and research topics that would define the future of blockchain.

This volume presents the accepted papers for the 2018 International Conference on Blockchain (ICBC 2018), held in Seattle, USA, during June 25–30, 2018. All topics regarding blockchain technologies, platforms, solutions, and business models align with the theme of ICBC. Topics of interest include, but are not limited to, new blockchain architecture, platform constructions, blockchain development and blockchain services technologies as well as standards, and blockchain services innovation lifecycle including enterprise modeling, business consulting, solution creation, services orchestration, services optimization, services management, services marketing, and business process integration and management.

We accepted 23 papers, including 16 full papers and seven short papers. Each was reviewed and selected by at least three independent members of the ICBC 2018 International Program Committee. We are pleased to thank the authors whose submissions and participation made this conference possible. We also want to express our thanks to the Organizing Committee and Program Committee members, for their dedication in helping to organize the conference and reviewing the submissions. We owe special thanks to the keynote speakers for their impressive speeches.

May 2018

Shiping Chen
Harry Wang
Liang-Jie Zhang

Organization

General Chairs

Chunxiao Xing	Tsinghua University, China
Yong Liu	Zhongguancun Internet Finance Institute, China

Program Chairs

Shiping Chen	CSIRO Data61, Australia
Harry Wang	University of Delaware, USA

Application and Industry Track Chairs

Fabrizio Lamberti	Politecnico di Torino, Italy
Khaled Salah	Khalifa University of Science, Technology and Research (KUSTAR), UAE
Luo Lei	University of Electronic and Science Technology of China (UESTC)

Program Vice Chair

Hanhua Chen	Huazhong University of Science and Technology, China

Services Conference Federation (SCF 2018)

General Chairs

Calton Pu	Georgia Tech, USA
Wu Chou	Essenlix Corporation, USA

Program Chair

Liang-Jie Zhang	Kingdee International Software Group Co., Ltd, China

Finance Chair

Min Luo	Huawei, USA

Panel Chair

Stephan Reiff-Marganiec	University of Leicester, UK

Tutorial Chair

Carlos A. Fonseca	IBM T.J. Watson Research Center, USA

Industry Exhibit and International Affairs Chair

Zhixiong Chen	Mercy College, USA

Organizing Committee

Huan Chen (Chair)	Kingdee International Software Group Co., Ltd, China
Jing Zeng (Co-chair)	Tsinghua University, China
Cheng Li (Co-chair)	Tsinghua University, China
Yishuang Ning (Co-chair)	Tsinghua University, China
Sheng He (Co-chair)	Tsinghua University, China

Steering Committee

Calton Pu	Georgia Tech, USA
Liang-Jie Zhang (Chair)	Kingdee International Software Group Co., Ltd, China

ICBC 2018 Program Committee

Fabrizio Lamberti	Politecnico di Torino, Italy
Allen Wei-Lun Chang	Tamkang University, Taiwan, China
Liqiang Wang	University of Central Florida, USA
Mustafa Canim	IBM T.J. Watson Research Center, USA
Nuno Laranjeiro	University of Coimbra, Portugal
Xumin Liu	Rochester Institute of Technology, USA
Pelin Angin	Middle East Technical University, Turkey
Massimo Mecella	Sapienza Università di Roma, Italy
Marouane Kessentini	University of Michigan, USA
Wubin Li	Ericsson Research, Italy
Yu Qi	Rochester Institute of Technology, USA
Xiaofeng Yu	Nanjing University of Financeand Economics, China
Walid Gaaloul	TELECOM SudParis, France
Sameep Mehta	IBM Research India
Suoratik Mukhopadhyay	Louisiana State University, USA
Guozhu Dong	Wright State University, USA
Akhil Kumar	Pennsylvania State University, USA
Tao Chen	University of Birmingham, UK
Praveen Jayachandran	IBM India

Roberto Natella	Federico II University of Naples, Italy
Pengcheng Zhang	Hohai University, China
Mohamad Reza Hoseiny	The University of Sydney, Australia
Stefan Tai	Berlin University, Germany
Waldemar Hummer	IBM Research, USA
Amjad Gawanmeh	Khalifa University, Abu Dhabi, UAE
Pengcheng Xiong Xiong	Apache Hive PMC, USA
Francois Charoy	University of Lorraine, France
Jun Shen	University of Wollongong, Australia
Latifur Khan	University of Texas at Dallas, USA
Richard Mark Soley	Object Management Group, USA
Nuno Antunes	University of Coimbra, Portugal
John Miller	University of Georgia, USA
Lina Yao	The University of New South Wales, Australia
Xiang Zhao	National University of Defense, China
Chao Wang	University of Science and Technology, China
Hailong Sun	Beihang University, China
Wei Li	The University of Sydney, Australia
Haopeng Chen	Shanghai Jiao Tong University, China
Shi Rui-Sheng	Beijing University of Posts and Telecommunication School of Humanities, China
Luca Cagliero	Politecnico di Torino, Italy
Shigeru Hosono	NEC Corporation, Japan
Ashiq Anjum	University of Derby, UK
Bo Cheng	Beijing University of Posts and Telecommunications, China
Guoquan Wu	Chinese Academy of Sciences, China
Huan Chen	Kingdee International Software Group Co., Ltd, China
Yun Yang Yang	Swinburne University of Technology, Australia
Daphne Soe-Tsy Yuan	National Chengchi University, China
Albert Lam	Hong Kong Baptist University, Hong Kong, SAR China
Han Rui	Chinese Academy of Sciences, China
Winnie Cheng	American Express, USA
Wanchun Dou	Nanjing University, China
Roberto Di Pietro	University of Rome, Italy
Yan Bai	University of Washington Tacoma, USA
Fahimeh Farahnakian	University of Turku, Finland
Jiuyun Xu	China University of Petroleum, China
Ernesto Damiani	University of Milan, Italy
Bruno Cabral	University of Coimbra, Portugal
Salima Benbernou	Paris Descartes University, UK
Rui Andr	CISUC, University of Coimbra, Portugal
Damian Andrew Tamburri	Jheronimus Academy of Data Science, TU/e, The Netherlands

Katt Basel	Norwegian Information Security Laboratory (NISlab) University of Gjøvik, Norway
Joao E. Ferreira	Universidade de São Paulo, Brazil
Georgiana Copil	TU Wien, Austria
Upendra Sharma	IBM T.J. Watson Research Center, USA
Houbing Song	West Virginia University, USA
Onur Altintas	Toyota, Japan
Natalia Kryvinska	University of Vienna, Austria
Ladjel Bellatreche	LIAS/ISAE-ENSMA, France
Shiping Chen	CSIRO, Australia
Yen-Hao Hsieh	Tamkang University, Taiwan, China
Rushikesh Joshi	IIT Bombay, India
Shruti Kunde	TCS, Hong Kong, SAR China
Jian Cao	Shanghai Jiaotong University, China
Jie Xu	University of Leeds, UK
Nagarajan Kandasamy	Drexel University, USA
Jiayu Zhou	Arizona State University, USA
Kisung Lee Lee	Louisiana State University, USA
Banage T. G. S.	Kumara Sabaragamuwa University of Sri Lanka
Bhanu Prasad	Florida A&M University, USA
Stephan Reiff-Marganiec	University of Leicester, UK
Rik Eshuis	Eindhoven University of Technology, The Netherlands
Pradeep Kumar Murukannaiah	Rochester Institute of Technology, USA
Javid Taheri Taheri	Karlstad University, Sweden
Muhammed Younas	Oxford Brookes University, UK
Rami Bahsoon	University of Birmingham, UK
Haytham ElGhazel	Polytech Lyon, France
Xiaofang Zhou	University of Queensland, Australia
Paci Federica Maria	University of Southampton, UK
Markus Kirchberg	VISA Inc. and National University of Singapore, Singapore
Mohamed Mohamed	IBM Research - Almaden, USA
Vijay Naik	IBM T.J. Watson Research Center, USA
Ajay Mohindra	IBM T.J. Watson Research Center, USA
Mohamed Sellami	ISEP
Samir Tata	IBM Research, USA
Ingo Weber	CSIRO Data61, Australia
Qinghua Lu	CSIRO, Australia

Contents

Research Track: Blockchain Research

Using Ethereum Blockchain in Internet of Things: A Solution for Electric Vehicle Battery Refueling

Haoli Sun[1(✉)], Song Hua[1], Ence Zhou[1], Bingfeng Pi[1], Jun Sun[2], and Kazuhiro Yamashita[3]

[1] Fujitsu Research and Development Center, Suzhou, China
{sunhaoli,huasong,zhouence,winter.pi}@cn.fujitsu.com
[2] Fujitsu Research and Development Center, Beijing, China
sunjun@cn.fujitsu.com
[3] Fujitsu Laboratories, Kawasaki, Japan
y-kazuhiro@jp.fujitsu.com

Abstract. Internet of Things (IoT) technology has become more and more popular recently. However, due to the limited resources of IoT devices and the centralized system architecture, some severe issues remain difficult to solve, such as: overload of centralized server, single point of failure, and the possibility of malicious usage of personal information. Blockchain technology has achieved big success in cryptocurrency trading. It has many unique features, such as consensus mechanism, peer to peer communication, implementing trust without a trusted third party, and transaction based on smart contract. Blockchain appears to be suitable to help building a distributed and autonomic IoT system to overcome the aforementioned problems.

In this paper, we introduce an Ethereum blockchain based rich-thin-clients IoT solution to solve the problems caused by limited resources of IoT devices when adopting mining mechanism of blockchain in IoT scenarios. Rich clients and thin clients can both provide blockchain accessing and data collecting functions while only rich clients with more resources can perform mining process. Furthermore, based on the solution, we present an electric vehicle battery refueling system in which battery swapping approach is adopted. We also explain the rationality of our solution by experiments and compare our solution with other blockchain based IoT solutions. Our conclusion is that our blockchain-IoT solution is suitable for various IoT scenarios while avoiding the problems caused by the limited resources of IoT devices.

Keywords: Internet of Things (IoT) · Blockchain · Ethereum Battery refueling

1 Introduction

Blockchain was first introduced by Satoshi Nakamoto in 2008 as the underlying data structure of Bitcoin [1]. As its name suggested, a blockchain is a chain of blocks, in which each block contains a number of transactions which are hashed in a Merkle Tree [2].

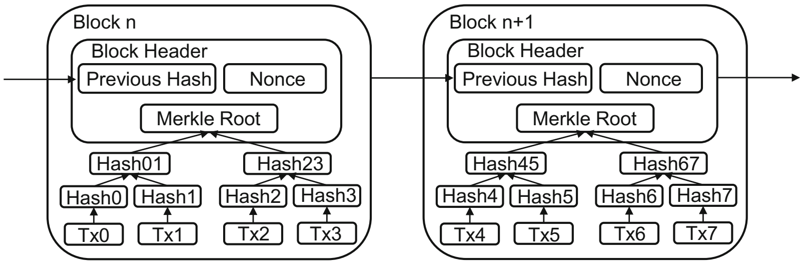


Fig. 1. Typical structure of a blockchain

By storing the hash value of the previous block, each block refers to its previous block, forming a chain structure. Figure 1 shows the typical structure of a blockchain. Together with peer-to-peer communication, consensus between miners such as Proof of Work (PoW), asymmetric encryption and digital signature, a blockchain system can provide a temper-proof and immutable value-transfer network which facilitates the booming of cryptocurrencies. This kind of blockchains which mainly used in cryptocurrencies can be defined as blockchain 1.0 [3].

Although Bitcoin supports scripts to define simple rules, the scripts are Non-Turing-Complete. In order to make blockchain suitable for more scenarios other than cryptocurrency, Ethereum [4] introduced smart contract which can be constructed with Turing-Complete programming languages (e.g. solidity [5]). Smart contracts are executable code stored on blockchain, defining what information to store and what transactions to execute. Theoretically, all transaction based state machines can be built by smart contracts. Figure 2 shows the mechanism of smart contracts. If an application is built only by Ethereum-like blockchain without cyber-physical interaction or other external facilities, it is defined as blockchain 2.0, which means blockchain based economic, market, and financial related applications [3].

Blockchain 3.0 means blockchain based applications beyond the scenarios of blockchain 2.0 [3]. For example, blockchain based naming systems [6, 7], health caring systems, IoT systems [8–11, 21, 22] and so on. As for blockchain based IoT systems, besides the blockchain related factors such as consensus method, smart contract support

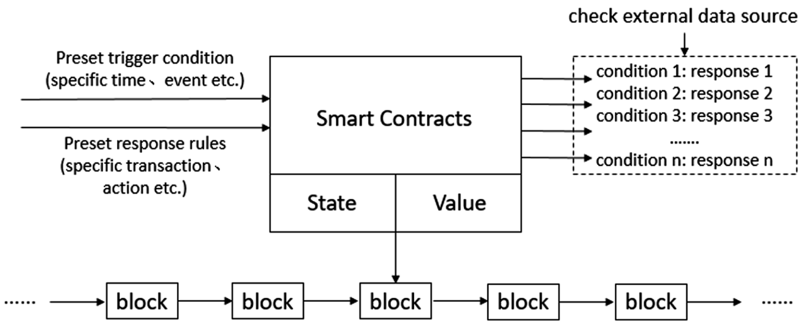


Fig. 2. The mechanism of smart contracts

or not and state machine management, IoT related factors such as the constraint of computing power, memory of devices and network bandwidth, and security of devices must also be considered. However, the problem that the limited resources of IoT devices cannot well support the mining process of blockchain nodes occurs frequently. In this paper, we introduce a rich-thin-clients solution for blockchain based IoT systems to solve the problem.

The remainder of this paper is organized as follows. In Sect. 2, we introduce some blockchain based IoT solutions. In Sect. 3, we propose an Ethereum blockchain based rich-thin-clients IoT solution and an implementation of EV battery refueling system using the proposed solution. A performance experiment is also introduced in Sect. 3 to prove the efficiency of the proposed solution. In Sect. 4, we compare our blockchain based IoT solution with others. In Sect. 5, we provide conclusions and future work.

2 Blockchain Based IoT Solutions

IoT means the Internet to which "things" (devices, sensors, actuators, etc.) are connected. The data from the physical world is gathered by sensors, delivered through the Internet. The users (or control unit) of IoT systems can analyze the gathered data to discover trends or patterns, or change the status of actuators based on the data. IoT technology has been developing rapidly these years. Gartner, Inc. forecasted that by 2020, 20.8 billion IoT devices will be connected to the Internet [15].

However, due to the limited computing power, storage and network bandwidth of IoT devices [16] and the centralized system architecture, some severe issues remain difficult to solve. Minhaj and Khaled listed out the security issues of different layers in IoT systems [16], such as "jamming adversaries", "insecure physical interface", "insecure neighbor discovery", "sinkhole and wormhole attacks", "sybil attacks" and so on. Besides, Nir [17] summarized IoT challenges mainly caused by the centralized architecture of current IoT systems: "costs and capacity constraints", "deficient architecture", "cloud server downtime and unavailability of services", and "susceptibility to manipulation".

Marco et al. [18] suggested that a hierarchical architecture, in which additional applications should be built on top of an underlying blockchain, is suitable for building IoT systems, e.g., the architecture of Blockstack [7] which is proposed by Muneeb et al. Blockstack is a global naming and storage system, by which users can register name-value pairs. The name-value pairs are generated according to particular private keys, and only the owner of the private keys can perform writing or updating operations on the name-value pairs. Blockstack separates "control plane" from "storage plane", which makes Blockstack outperforms Namecoin [6]. Furthermore, the authors believe that various state machines can be built in the "Control plane" of BlockStack.

Ali et al. [8] proposed a blockchain based IoT solution for smart homes, and they conducted further researches about their proposed solution [19, 20]. The system architecture of their solution is also hierarchical, consisting of "smart home", "overlay network", and "cloud storage". "Smart home" contains all smart devices in the home, local blockchain, smart home manager (SHM) [20], and local storage. "Overlay network" connects smart homes and cloud storages, and it provides distributed trust by overlay blockchain. "Cloud storage" is used to store data from smart home in order to provide data to other services on the Internet. Considering IoT devices do not have enough resources to support PoW, the authors designed their proposed system architecture without PoW. In a smart home, local blockchain is centrally managed by the owner through SHM. The nodes of overlay network could be SHMs and other devices with relatively high resource. The nodes are grouped in clusters to decrease network overload, and one of the nodes in each cluster is elected as a cluster head (CH). CHs are responsible for evaluating if other CHs are trustable by maintaining a trust rating based on direct and indirect evidence. The overlay blockchain is maintained by the CHs, since consensus mechanism is not adopted in this system, forking of blockchain is permitted. The authors came to the conclusion that the architecture they proposed keeps the benefits of traditional blockchain such as privacy and security while eliminating PoW of traditional blockchain for better performance.

Seyoung et al. [9] constructed an automatic electricity usage adjustment system based on Ethereum. The system consists of the Ethereum blockchain, a smart phone and three Raspberry Pis respectively representing an electricity meter, an air conditioner and a lightbulb. Since the smart contracts can define the behaviors of IoT devices, a user can set up policies of how the devices work using a smart phone. For example, user can define a threshold of electricity usage, once the threshold is reached, the air conditioner will change to energy saving mode. The authors used RSA algorithm to provide public key and signature functions instead of Ethereum accounts for fine-granularity.

Besides, Kamanashis et al. [10] proposed a general architecture for a blockchain based smart city solution, they utilized different blockchain systems in their solution, such as using both Ethereum and NXT in communication layer and using both permissionless and permissioned distributed ledgers in database layer, which may be an inspiring idea for utilizing blockchain systems in complex IoT scenarios.

In summary, the IoT issues which can be solved by blockchain are mainly the issues caused by the centralized architecture of current IoT systems. Some of the issues are listed:

- Central server failures: failures may be caused by fault of software or attacks.
- Single point of failure: a compromised device can cause failures of entire system.
- Lack of privacy: personal information saved in central server may be abused.

Blockchain can solve the above issues because its decentralized architecture can prevent central server failures or single point of failure, and its public-private key encryption can provide pseudonymity [18] to protect personal information at some degree. However, the issues caused by the limited resources of IoT devices still remain difficult to solve: on one hand, most of the IoT devices do not have enough resources to support PoW, on the other hand, if a device is not a node of blockchain network, its security and identity is difficult to be guaranteed.

Besides, we found that the following factors are essential for constructing blockchain based IoT systems:

- What underlying blockchain system should be used?
  - What consensus method should be used?

  - Is Turing-Complete programming necessary or not? (Requiring smart contracts or not?)
  - How to manage state machines?
  - How should security and privacy be guaranteed?
- How to combine blockchain with IoT?
  - How to enable cyber-physical interaction?
  - If IoT devices need to be blockchain nodes or not?
  - How to design the system architecture? (hierarchical architectures are often adopted [7, 8, 10, 11, 22])
  - How to design the topology of IoT devices and blockchain nodes? (Such as clusters in [8], a cluster is a star topology. And the topology between CHs is P2P)

3 Ethereum Blockchain Based Rich-Thin-Clients IoT Solution

In this section, we propose an Ethereum blockchain based rich-thin-clients IoT solution. We designed a rich-thin-clients architecture to solve the aforementioned dilemma between limited resources of IoT devices and the concerns for centralized architecture. The thin clients, which are responsible for user interaction and IoT data collection, can be considered as IoT devices with constrained resources; the rich clients, which are thin clients plus full blockchain nodes, can be considered as devices that have resources greater than or equal to personal computers. We use a private Ethereum blockchain network built by ourselves as our underlying blockchain system. Not only because it supports smart contracts by which we can design relatively complex interaction between different IoT devices and between human users and devices, but also due to the fact that it can generate a new block faster than Bitcoin [9, 23]. We also utilized the original consensus method: PoW, and the original encryption method: Ethereum account.

3.1 System Architecture and Topology

We designed a rich-thin-clients architecture which differs from the hierarchical architecture in that rich clients and thin clients have some overlapped functions while layers or components in a hierarchical architecture usually perform different functions.

Figure 3 shows the rich-thin-clients architecture. Rich client and thin client can both provide a GUI for users, invoke blockchain (BC) APIs through BC interface which is deployed in a rich client, define business logics and collecting IoT data (optional for rich clients), but only a rich client contains a full BC node which can perform mining and contains all transaction records of the BC system.

Figure 4 shows the overall architecture and the topology formed by rich clients and thin clients. Each rich client contains a fully functional Ethereum BC node which can perform mining and run PoW consensus algorithm with other Ethereum BC nodes. The rich clients form a P2P network, just like public Ethereum BC nodes do. The thin clients connected to one same rich client form a star topology with the rich client. Since only the rich clients with high resources perform mining and consensus algorithm, our network should be similar to the public Ethereum BC network in performance, thus the dilemma between limited resources of IoT devices and the concerns for centralized architecture can be solved.

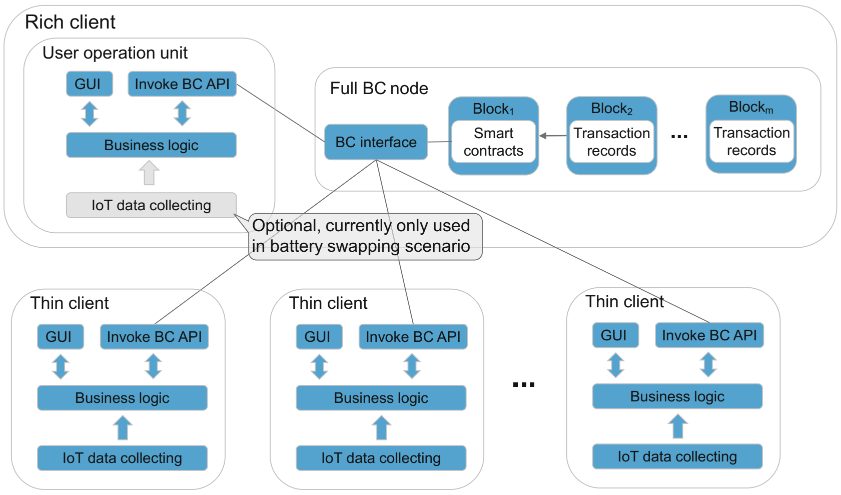


Fig. 3. Rich-thin-clients architecture

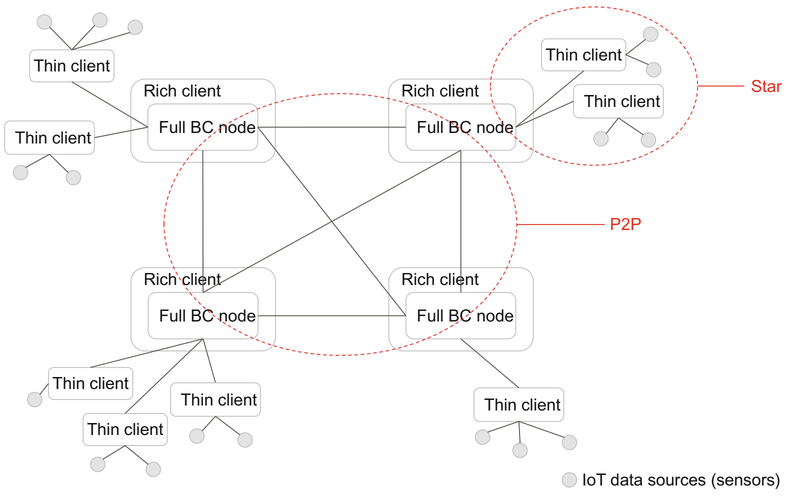


Fig. 4. Overall system architecture

3.2 Privacy and Security

We use Ethereum account mechanism to identify each client and to encrypt transactions. We assign unique Ethereum account for each client, so that they can be uniquely identified in our system. Each client gets its unique public key and private key along with its Ethereum account. And since the accounts are not directly connected to personal information in the real world, this could provide privacy to users.

In an IoT system, IoT devices (thin clients) are the most vulnerable parts facing attacks. If an IoT device is hacked, the attacker may adopt three operations:

1. Steal the property of the account which was used on this device. Unfortunately, this problem is unsolvable by using centralized architecture nor decentralized architecture.
2. Hack other devices through this device. This situation may cause severe issues in a centralized system, because the central servers may get attacked through an IoT device, but it will not happen in a decentralized architecture.
3. Fake an account or send faked data through this device. Since only the rich clients can generate a valid account, the faked account cannot be validated in our system. And thanks to blockchain's validation mechanisms when generating blocks, invalid transactions or accounts will be refused.

Furthermore, even if a rich client is hacked, since it's just a single node in the blockchain network, other rich clients will reject the invalid requests sent from it.

3.3 Implementation of Ethereum Based Cyber-Physical Battery Refueling System

Electric Vehicle Battery Refueling. Due to the development of battery technologies and the environmental awareness, EV technologies have been developing rapidly in last decades. With large-scale utilization of EV technologies, the release of greenhouse gases can be reduced, and the energy utilization can be more efficient. However, battery refueling is still a problem which have not been well solved. There are three major EV battery refueling methods: Alternating Current (AC) charging, Direct Current (DC) charging and battery swapping. AC charging can be adopted in an EV owner's garage. It's convenient but time-consuming. It will cost more than eight hours to fully charge a depleted EV battery by AC charging. DC charging can be provided by charging station. It will cost 1–2 h to fully charge a depleted EV battery by DC charging. However, DC charging may harm EV battery because of the large power. Battery swapping is the least time-consuming one among the three EV battery refueling methods. It will cost only a few minutes to swap a depleted battery for a fully charged one by a battery swapping station [12].

Tesla Inc. and NIO Inc. introduced their EV battery swapping technologies in 2013 and 2017 respectively [13, 14], suggesting that battery swapping may be a promising solution for EV battery refueling. In our previous work [12], a blockchain based EV battery swapping system (in the form of a web application) was proposed to evaluate the batteries to be swapped fairly by smart contracts and to manage the batteries' information such as its manufacturer, brand, power capacity, price and refueling history.

It is actually an IoT scenario to use our previous blockchain based EV battery swapping system in real situation, since the battery swapping stations and EVs have to be connected to the Internet, and the cyber-physical interaction of the battery information has to be involved. In this section, we verify if our blockchain based IoT solution can be utilized in the EV battery swapping scenario.

Architecture of the Battery Swapping System. We implemented the battery swapping system based on our proposed rich-thin-clients architecture. Figure 5 shows how the system is composed.

We use Raspberry Pi (RPi) [24] as the hardware of a thin client, each thin client represents an EV. In each EV, "truffle" [25] is used to invoke Remote Procedure Call (RPC) service of blockchain, and a local "express" [26] server is used to control the cyber-physical-interaction. We use USB disks which can be connected to RPi to represent real batteries, the information of each battery is stored in a file in each corresponding USB disk. Figure 6 shows the static information and the dynamic information of a typical battery information file. (Static information and dynamic information is introduced later in "Smart Contracts" of 3.3.)

A station consists of a battery swaping interface (an RPi works like an EV) and a full Ethereum node. We use "Geth" [27] as the command line interface for running full Ethereum nodes and providing RPC service for invokers.

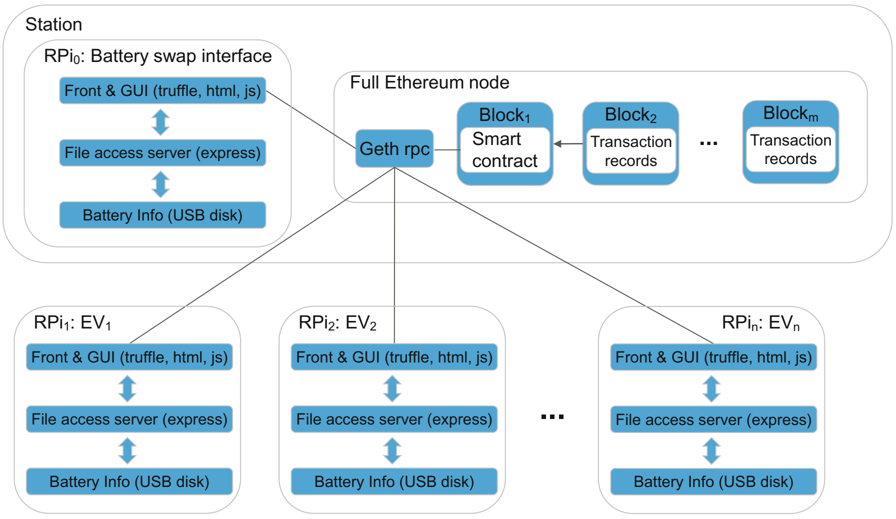
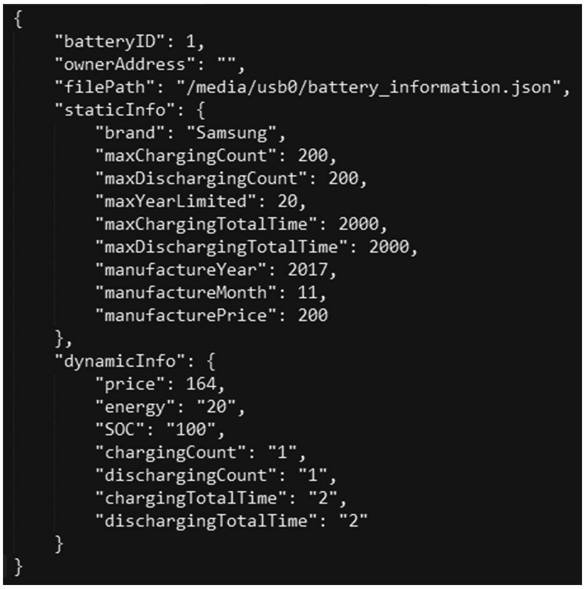


Fig. 5. Architecture of the battery swapping system



```
{
    "batteryID": 1,
    "ownerAddress": "",
    "filePath": "/media/usb0/battery_information.json",
    "staticInfo": {
        "brand": "Samsung",
        "maxChargingCount": 200,
        "maxDischargingCount": 200,
        "maxYearLimited": 20,
        "maxChargingTotalTime": 2000,
        "maxDischargingTotalTime": 2000,
        "manufactureYear": 2017,
        "manufactureMonth": 11,
        "manufacturePrice": 200
    },
    "dynamicInfo": {
        "price": 164,
        "energy": "20",
        "SOC": "100",
        "chargingCount": "1",
        "dischargingCount": "1",
        "chargingTotalTime": "2",
        "dischargingTotalTime": "2"
    }
}
```

Fig. 6. An example of battery information file

Smart Contracts. We implemented three smart contracts to manage the state machine of the battery swapping system [12]:

- "BatteryProcess" smart contract is used to operate and store battery information. It stores the batteries' static information and dynamic information. Static information is determined since a battery was produced and cannot be altered, such as brand, production time, manufacture price, and so on. Dynamic information is used to show a battery's status, such as charge times, state of charge (SOC), price, owner's account, and so on.
- "BalanceProcess" smart contract is used to manage value transfer between accounts. Given that it is not rational to require every EV user to have Ether [4], we defined a token called E-coin as the currency in our system.
- "BatteryInterface" smart contract provides API interfaces for three types of terminal users: station operator, EV owner and super account. EV owners can discharge the battery in their own EV, and they can send a battery swapping request to a battery station and wait for the confirmation. Station operators are the employees of battery station. They can charge, discharge, and recycle batteries belong to the battery station, and they can approve or deny a battery swapping request sent by EV owners. Super account is the system manager of this battery swapping system, he can create other types of accounts, grant E-coins, define the GAS required to invoke smart contracts, and so on.

System Flowchart. Figure 7 shows how the system works. At first, the EV has a depleted battery which belong to the brand of TOY, and the station has two fully

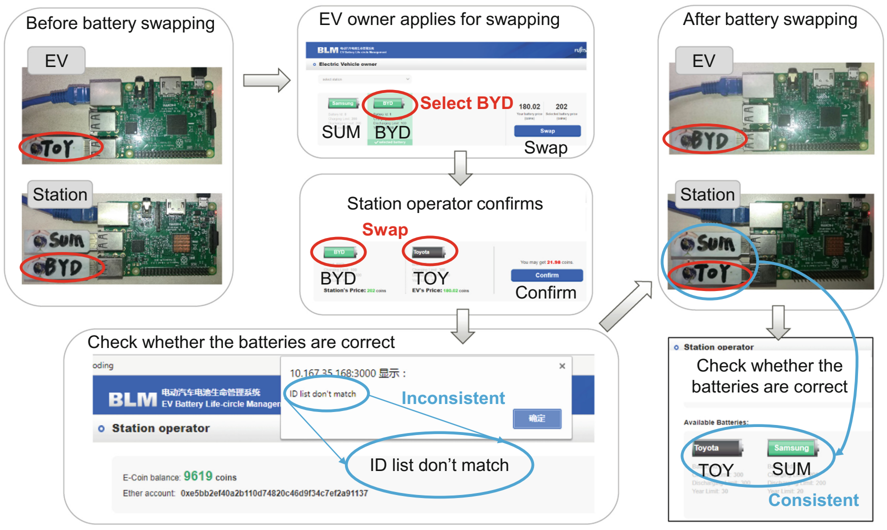


Fig. 7. System flowchart

charged batteries which belong to the brand of SUM and BYD respectively. Then the EV owner accesses the GUI of battery swapping system and send a swapping request to swap his TOY battery with station's BYD battery. After the station operator confirms the swapping request, the ownership information stored in blockchain is changed. However, at this point of time, the corresponding "real" batteries (USB disks) have not been swapped yet, so the inconsistency between cyber information and physical information is alerted. Finally, after the swapping of corresponding batteries, there is no alert any more, and the current battery status can be checked.

There are three kinds of major operations in the working procedure:

- **Initialization.** EV and Station perform initialization respectively. The system on RPi will read the information of all the batteries inserted on RPis' USB interfaces, and the smart contracts will create a battery record based on the static information of each "real battery" and generate a unique battery ID, meanwhile, the generated battery ID will be written into the battery information file inside the USB disk.
- **Battery information consistency checking.** Every time an EV owner or a station operator accesses the index page which shows the information of batteries belong to the EV or station, the system will check whether the batteries' static information recorded in USB disks are consistent with the batteries' static information on blockchain. If any inconsistency is found, an alert box will pop up. After an EV owner submits a swapping request, and a station operator confirms the request, the corresponding USB disks must be changed to remove inconsistency.
- **Battery charging and discharging.** When a user performs charging operation or discharging operation on a battery, the system will change the dynamic information of the battery on blockchain while changing the dynamic information recorded on USB disk according to the battery ID.

3.4 Experiments

In order to find a proper way to utilize Ethereum in the rich-thin-clients architecture, we have tried three different kinds of implementations, which are presented in Fig. 8.

The 1st implementation (Fig. 8(a)) is deploying a Geth client and a full Ethereum node on a Raspberry Pi which is utilized as the rich client, and starting mining on the Raspberry Pi. The 2nd implementation (Fig. 8(b)) is deploying Geth client without mining on each Raspberry Pi and deploying Geth and a full Ethereum node on a PC, the Ethereum nodes on Raspberry Pis synchronize data from the mining node. The 3rd implementation (Fig. 8(c)) is deploying a Geth client and a full Ethereum node only on the PC of a rich client, the Raspberry Pi of the rich client works only as a battery swapping interface.

The 1st implementation caused system crash soon after the starting of mining, because the resources of a Raspberry Pi cannot support the mining of a full Ethereum node, so it was excluded from the performance comparison.

We conducted quantitative comparison between the 2nd implementation and the 3rd implementation. As for experimental environment, Raspberry Pi 3B+ with ARMv7 1.2 GHz CPUs, 1 GB RAM and Raspbian OS, and PC with Intel 2.0 GHz 64 bit CPUs, 66 GB RAM and Ubuntu 16.04.2 OS were used. We monitored the CPU and memory usage percentages of the Raspberry Pi in thin clients of the 2nd implementation and the 3rd implementation. Furthermore, we timed the EV discharging as an index to show the efficiency of each implementation. We conducted the experiments by repeating the battery swapping system's typical scenario which consists of the processes of initialization, EV discharging batteries, battery swapping and station charging batteries. Figure 9 shows how the CPU, memory and network usage change during one time of experiment, indicating that the 2nd implementation clearly requires more CPU and memory usages than the 3rd implementation. Figure 10 shows the statistical results of 10 times of repeated experiments, indicating that the 3rd implementation requires less resources and costs less time when performing EV discharging process. We can tell from the results that the 3rd implementation outperforms the 2nd implementation, thus we used the 3rd implementation in our previously proposed solution.

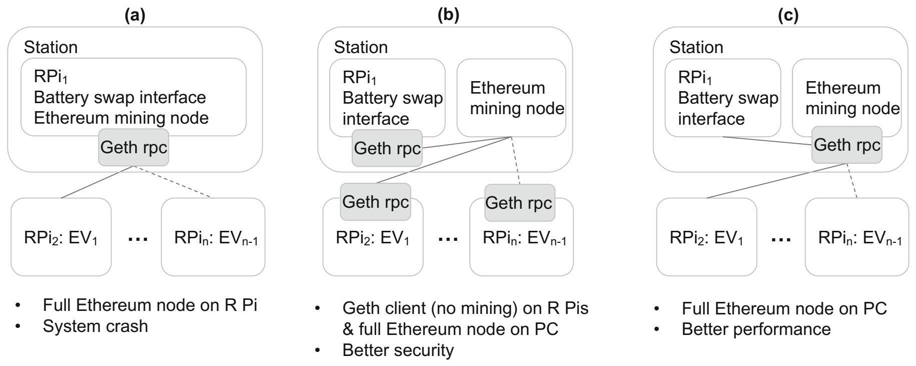


Fig. 8. Different implementations of rich-thin-clients architecture

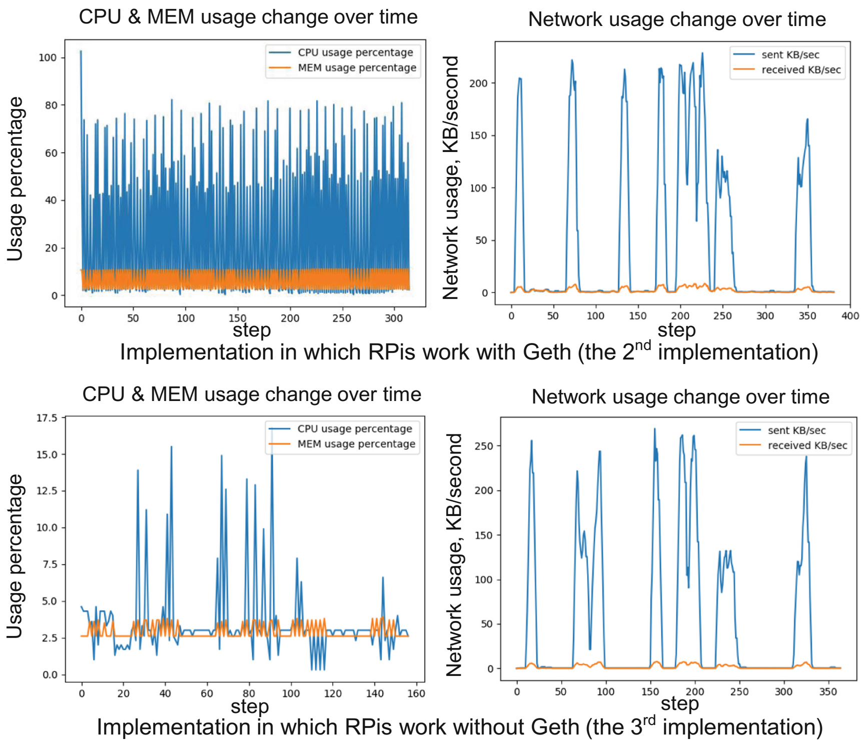


Fig. 9. Comparison of CPU, memory and network usages for two different implementations

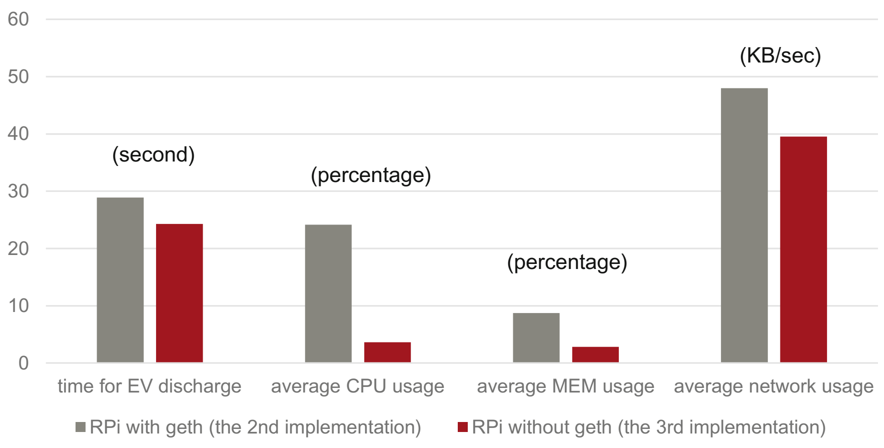


Fig. 10. Statistical performance comparison between two different implementations

4 Comparison Between Blockchain Based IoT Solutions

We made a comparison between our solution and other blockchain based IoT solutions. Table 1 shows the comparison result of some major features. Due to the usage scenarios of the solutions are totally different, so that the performance comparison between different solutions is absent in this paper.

Table 1. Comparison between blockchain based IoT solutions

Solution or Application	BC related features					Major data storage	Architecture
	Underlying BC system	Consensus method	Support smart contract?	State management	Security & Privacy		
Battery swapping (Our solution)	Non-public Ethereum	PoW	Yes	By Ethereum	Ethereum account mechanism	In Ethereum	• Rich-thin-clients • P2P + star topology
Automatic electricity usage adjustment	Public Ethereum	PoW	Yes	By Ethereum	Public key & private key based on RSA algorithm	In Ethereum	P2P topology
Smart city	Hybrid (Ethereum + other BC, e.g.NXT)	Unspecified	Yes	Unspecified	• By blockchain protocols • Recommend private ledgers	On BC	Hierarchical
Blockstack (A global naming and storage system)	Public Bitcoin	PoW	No	By “control plane”	Depend on the underlying blockchain	In “storage plane”	Hierarchical
BC-based smart home	Bitcoin-like	No consensus	No	By BC and Policy headers	• Public/private keys, shared key • By PK lists	*Local storage *Cloud storage	Hierarchical
LoRaWAN IoT (A low power wide area technology)	Bitcoin-like	PoW	No	By BC	• LoRaWAN is already safe • Public-key cryptography and digital signature	On BC	Hierarchical

Since we use Ethereum as underlying blockchain system, our solution surpasses the solutions which use Bitcoin as underlying blockchain system [7, 8, 11] in terms of flexibility of constructing state machines (supporting smart contract). Although Blockstack [7] also supports various state machines, there is no evidence to indicate that its state machine constructing mechanism is more convenient than smart contract of Ethereum. As for the automatic electricity usage adjusting system proposed by Seyoung et al. [9], although Ethereum blockchain is used, there is no value-transfer mechanism involved in their system, so Ethereum’s advantages are not fully utilized in their solution.

5 Conclusions and Future Work

This paper proposes an Ethereum blockchain based IoT solution, introduces a battery swapping system implemented based on the solution and experiments to test the system's performance indexes, and compares our solution with other blockchain based IoT solutions. We use Ethereum (a blockchain system which supports smart contracts) to solve the issues caused by traditional centralized IoT architecture. And by adopting a rich-thin-clients architecture, the dilemma between limited resources of IoT devices and the concerns for centralized architecture can be solved. Besides, the usage of our solution is not limited to battery swapping application, other IoT applications such as trading systems for sensor data or other digitalized property [21, 22] may also benefit from our proposed architecture.

In future studies, we would like to build a more practical battery swapping system with actual vehicle system instead of Raspberry Pi and to include multiple stations to simulate real usage scenarios.

References

1. Satoshi, N.: A peer-to-peer electronic cash system (2008)
2. Merkle, R.C.: Protocols for public key cryptosystems. In: 1980 IEEE Symposium on Security and Privacy, pp. 122–122. IEEE (1980)
3. Melanie, S.: Blockchain: blueprint for a new economy. O'Reilly Media Inc., Sebastopol (2015)
4. Wood, G.: Ethereum: a secure decentralised generalised transaction ledger. Ethereum Project Yellow Paper (2014)
5. Solidity Documentation, http://solidity.readthedocs.io/-en/latest/index.html. Last accessed 07 Feb 2018
6. Namecoin Homepage. https://namecoin.org. Last accessed 07 Feb 2018
7. Muneeb, A., Jude, N., Ryan, S., Michael, J.F.: Blockstack: a global naming and storage system secured by blockchains. In: USENIX Annual Technical Conference, pp. 181–194. (2016)
8. Ali, D., Salil, S., Raja, J.: Blockchain in internet of things: challenges and solutions. arXiv preprint, arXiv:1608.05187 (2016)
9. Seyoung, H., Sangrae, C., Soohyung, K.: Managing IoT devices using blockchain platform. In: 2017 19th International Conference on Advanced Communication Technology (ICACT), pp. 464–467. IEEE (2017)
10. Kamanashis, B., Vallipuram, M.: Securing smart cities using blockchain technology. In: 2016 IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS), pp. 1392–1393. IEEE (2016)
11. Jun, L., Zhiqi, S., Chunyan, M.: Using blockchain technology to build trust in sharing LoRaWAN IoT. In: Proceedings of the 2nd International Conference on Crowd Science and Engineering, pp. 38–43. ACM (2017)
12. Song, H., Ence, Z., Bingfeng, P., Jun, S., Yoshihide, N., Hidetoshi, K.: Apply blockchain technology to electric vehicle battery refueling. In: Proceedings of the 51st Hawaii International Conference on System Sciences (2018)

13. Tesla Bttery Swap Event. https://www.tesla.com/videos/battery-swap-event. Last accessed 08 Feb 2018
14. NIO Power. https://www.nio.com/en/nio-power. Last accessed 08 Feb 2018
15. Gartner Says 6.4 Billion Connected "Things" Will Be in Use in 2016, Up 30 Percent From 2015, https://www.gartner.com/newsroom/id/3165317. Last accessed 11 Feb 2018
16. Khan, M.A., Salah, K.: IoT security: review, blockchain solutions, and open challenges. In: Future Generation Computer Systems (2017)
17. Nir, K.: Can blockchain strengthen the Internet of Things? IT Prof. **19**(4), 68–72 (2017)
18. Marco, C., Antonio, V., Juan, C.D.M.: Blockchain for the Internet of Things: a systematic literature review. In: 2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA), pp. 1–6. IEEE (2016)
19. Ali, D., Salil, S., Raja, J., Praveen, G.: Blockchain for IoT security and privacy: the case study of a smart home. In: 2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), pp. 618–623. IEEE (2017)
20. Ali, D., Salil, S., Raja, J.: Towards an optimized blockchain for IoT. In: Proceedings of the Second International Conference on Internet-of-Things Design and Implementation, pp. 173–178. ACM (2017)
21. Dominic, W., von Bomhard, T.: When your sensor earns money: exchanging data for cash with bitcoin. In: Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication, pp. 295–298. ACM (2014)
22. Yu, Z., Jiangtao, W.: An IoT electric business model based on the protocol of bitcoin. In: 2015 18th International Conference on Intelligence in Next Generation Networks (ICIN), pp. 184–191. IEEE (2015)
23. Cryptocurrencies with much faster block times than bitcoin. https://themerkle.com/4-cryptocurrencies-with-much-faster-block-times-than-bitcoin. Last accessed 24 Feb 2018
24. Raspberry Pi Homepage. https://www.raspberrypi.org. Last accessed 27 Feb 2018
25. Truffle. https://github.com/trufflesuite/truffle. Last accessed 27 Feb 2018
26. Express. https://github.com/expressjs/express. Last accessed 27 Feb 2018
27. Geth. https://github.com/ethereum/go-ethereum/wiki/geth. Last accessed 27 Feb 2018

A Simulation Approach for Studying Behavior and Quality of Blockchain Networks

Bozhi Wang[1(✉)], Shiping Chen[1,2], Lina Yao[1], Bin Liu[1,2], Xiwei Xu[1,2], and Liming Zhu[1,2]

[1] University of New South Wales, Sydney, NSW 2062, Australia
bozhi.wang@student.unsw.edu.au
[2] CSIRO Data61, Sydney, NSW 2110, Australia

Abstract. Blockchain, as the fundamental technology of Bitcoin, tends to be an-other technology revaluation of the Internet, due to its unique security, trustworthiness and reliability. However, due to the massive deployment and high running cost, it is hard for researchers to study the dynamic behavior of a large and alive Blockchain network, e.g., Bitcoin. In this paper, a proposal on a simulation approach to study Blockchain protocols in sizable Blockchain networks is made. First, several key requirements for software tools to simulate Blockchain are identified. Secondly, the focus is on the selection and definition of key performance metrics to quantify Quality of Blockchain (QoB). We aim to demonstrate the proposed idea by using an existing simulation tool to duplicate a simplified Blockchain Proof of Work (PoW) protocol with different parameters and observations. Our case study shows that it is possible and practical to use a simulation approach to study Blockchain networks with different network sizes and protocols.

Keywords: Blockchain · Proof of Work (PoW) · Discrete-event simulation
Performance metric · Quality of Blockchain (QoB)

1 Introduction

For the past few years, people can do almost everything on the Internet. However, most of the online transactions from person to person, no matter email, money, music or anything else, rely on big intermediaries, such as email services, banks and telecommunications operators. There are growing problems with such centralized systems. For example, our privacy is under infringement; the cost of sending money overseas turns out to be too high and takes days. Things started to change in 2008, when Satoshi Noakmoto published his paper, "Bitcoin: A Peer-to-Peer Electronic Cash System" [1]. The key technology of Bitcoin, called Blockchain, shows its strength in the future Internet to enable decentralization as a new paradigm for large-scale distributed systems. While TCP/IP is the communicating protocol between computers, Blockchain is its trust mechanism and their cooperation protocol. Blockchain offers some unique capabilities/features, such as decentralized, reliability and immutability, which enabled people to establish trust and do transactions without relying on a trusted third party. Till now, Blockchain is starting to show its potentials in many interesting domains, such as transnational payment, distributed storage, food provenance and etc.

S. Chen et al. (Eds.): ICBC 2018, LNCS 10974, pp. 18–31, 2018.
https://doi.org/10.1007/978-3-319-94478-4_2

Before Blockchain could be widely adopted in other areas, it still has a number of issues to be solved. First, the current mining (PoW- Proof of Work) mechanism of Blockchain are wasting a large amount of computing power on computation, which is necessary for PoW but meaningless and costly. Second, every node in the network has a complete ledger. As time passing by, the ledger will become larger and larger. And when a miner verifies a transaction, it should track all the historical transactions recorded on the blockchain. The performance issue is getting worse and worse. Then, though Blockchain is an anonymous system, all the transactions are publicly available, which may cause privacy issue. Lastly, in the current Bitcoin network, a common security strategy is to wait six blocks to confirm the transactions in the last block, which last around one hour, which significantly limits the adoption and applications of the current Blockchain technologies. With these problems, many ideas are proposed yet still under developing, such as IOTA [19], EOS.

On the other hand, most current blockchain systems are driven by electronic money. For example, the open source Etherum, once creating a smart contract, it costs some ETH, the cryptocurrency on Ethereum blockchain. When conducting some tests and/or updating, it is also very expensive and has side-effects and/or interruptions on the main blockchains. So we need an approach to testing new ideas in different scopes and sizes of blockchain networks with little costs and impacts on the existing real blockchain networks. Simulation is a good way out. It doesn't cost much, and could find out any tiny change in the system.

In this paper, we propose a simulation approach to study the large-scale Blockchain networks. First, we identify key requirements for the simulation software for implementing PoW protocols. Second, we collect and define a number of performance metrics to quantify the Quality of Blockchain (QoB). Then, we demonstrate the proposed idea using a simple simulation tool to duplicate a simplified Bitcoin PoW protocol using different configurations. Our case study shows that it is possible and practical to study a large-scale Blockchain network using simulation software.

The rest of this paper is organized as follows. Section 2 gives an overview of the PoW protocol in Bitcoin and defines some metrics for QoB. In Sect. 3, we identify the key criteria for functionality and capabilities required by Blockchain simulators. Section 4 shows a simple Blockchain simulator by leveraging SimPy. Section 5 presents some related work. We conclude the paper in Sect. 6.

2 Overview of Blockchain

2.1 Blockchain

Essentially, Blockchain is a peer-to-peer distributed ledger database, which consists of connected data blocks. The connection pointer between blocks is the Header Hash processed by cryptographic hash function that protects the transactions in every block, as well as the connected blocks. Thus, any of the historical transaction cannot be changed without invalidating a chain of Header Hash.

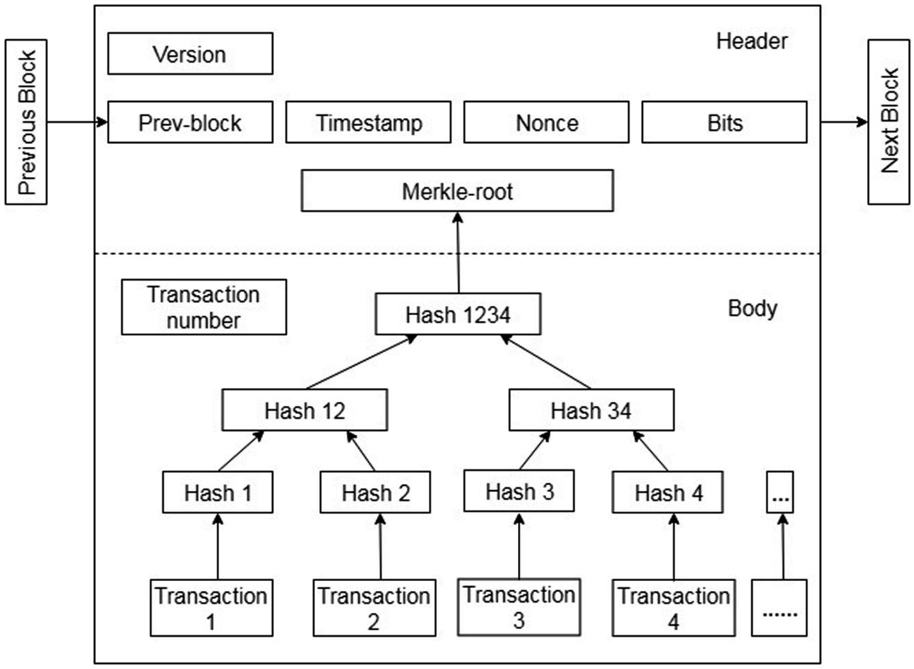


Fig. 1. The internal structure of block

In Bitcoin system, one block is created about every ten minutes. Transactions happening in the Bitcoin system are saved in the blocks. A block contains a Header and a Body, as shows in Fig. 1. The Header contains metadata of the block, such as Version, Prev-block pointing to the previous block, Timestamp, Nonce, Bits, Merkle-root etc. The body mainly includes the details of the transactions in the structure of Merkle Tree. These transactions form a publicly global ledger in Blockchain system, where the transactions recorded in the ledger could be queried by anyone who can access Internet. Every transaction is signed by a digital signature of the sender to ensure they are un-forged and not duplicated. The Merkle Tree with all the transactions has a unique Merkle-root calculated by the Hash procedure, which is recorded in the Header.

The Block is created through mining process, which is an exhaustive random number algorithm. During this process, the miners package the hash value of the previous block and all the transactions have happened in the latest ten minutes, and find a value for Nonce to calculate a hash value with 256 bits. The mining process is aimed to find the value of Nonce that makes the hash value meets some requirements, such as having a certain number of zeros in the first bits. The miner, who successfully find such a value, gets the right to write the new block to the blockchain by broadcasting the new block to the Bitcoin other nodes to verify (Fig. 2).

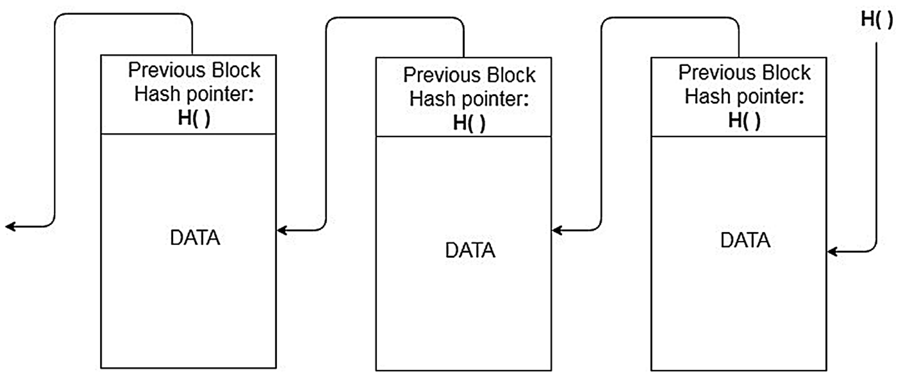


Fig. 2. The structure of blockchain

The system has a competition mechanism for miners to compete for the right of writing, which calls Proof of work (describe in the next part). During the process, the more computing power a miner spends, the larger possibility the miner can get the right. If the new block is successfully added into the blockchain, the miner will get some reward. Also, it is possible for two different miners to find new blocks almost at the same time. The two blocks might be verified and accepted by a subset of the Bitcoin network, which form a fork. In such a case, the other miners need to choose the fork which with larger number of blocks (implying heavier work).

2.2 Proof of Work Protocol in Bitcoin Blockchain

The distributed system consensus algorithm which appeared in 1980s is the foundation of Blockchain Consensus. There are several Byzantine Fault Tolerance protocols which used in different Blockchain platforms, such as PBFT (Practical Byzantine Fault Tolerance), Raft, PoW (Proof of Work), PoS (Proof of Stake), DPoS (Delegated Proof of State). Bitcoin Blockchain uses PoW (Fig. 3).

The main work of PoW is to do a lot of SHA256 hash calculations during the mining process. The process to reach consensus is discussed as below.

1. A user initiates and signs a new transaction, broadcasts to the whole network asking for accounting.
2. The miner, which receives the transaction, puts it in the Mempool.
3. For every round of computation, the miner add all the transactions in the Mempool to create the body of a new block, and works on PoW procedure, to find out one proof of work with enough difficulty.
4. The miner who wins the competition broadcasts the new block to the Bitcoin network.
5. Other miners who receive the new block verify all the transactions in the block to ensure they are all valid and new.
6. Once the new block is accepted by the whole network, it is added at the end of the blockchain.

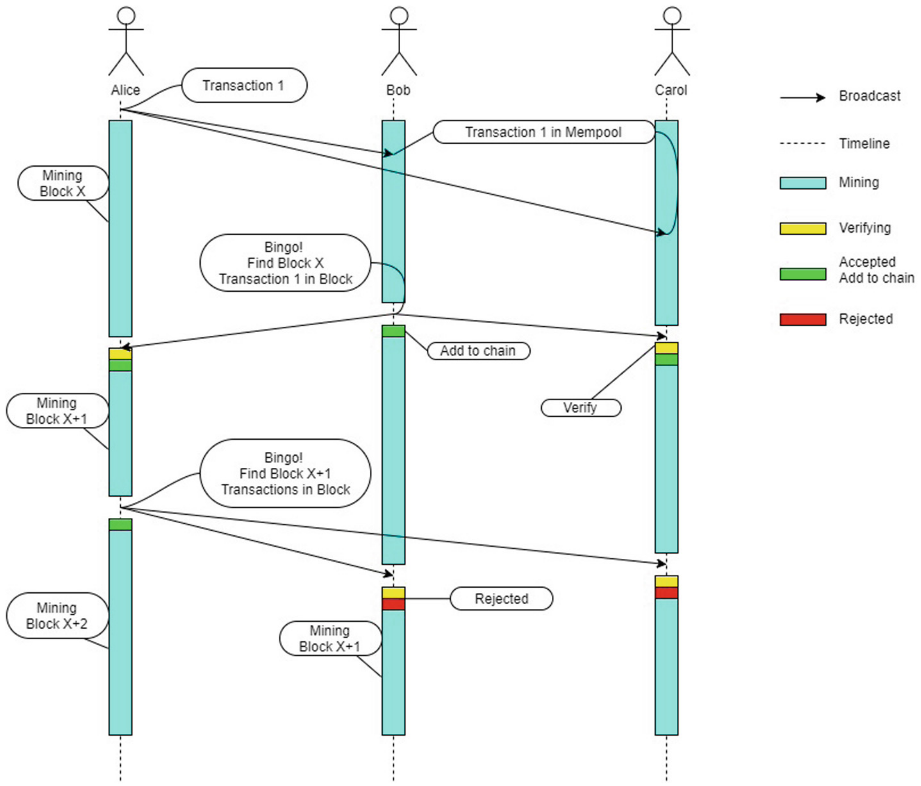


Fig. 3. Proof of work procedure

Through the mining process discussed above, the transactions from different users are written in the blockchain, which is hosted on every single node within the network. So we can get a distributed and high reliable and consistent global ledger.

2.3 Parameters and QoS Metrics for a Blockchain Network

In this section, we collect and define some metrics for QoS of Blockchain (Fig. 4). We can define and simulate a Blockchain Network using these metrics. We ignored both orphaned blocks and network delay in our simulator. It will be fixed in the future research.

TBN (Total Block Number). The number of blocks that have been mined in a specific time period
ABS (Average Block Size). The average block size in MB.
BCT (Block Commit time). The average time needed to commit a block to the main chain since being created.
TPB (Transactions per Block). The average number of transactions per block
ATS (Average Transaction Size). The average transaction size in Byte.

TCT (Transaction Confirmation Time). The average time for a transaction to be accepted into a mined block.

TPD (Transactions per day). The number of daily confirmed transactions.

MS (Mempool Size). The aggregate number of transactions waiting to be confirmed.

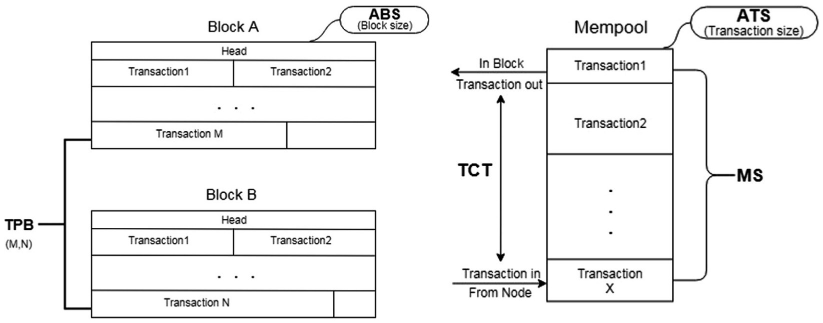


Fig. 4. QoS metrics for a blockchain network

We use these metrics to measure the quality of the Blockchain system. TBN directly related to simulation time. BCT is approximately equals to 6 times of mining time. Block Size includes the header and some transactions, and is limited by 1000000B (defined by Bitcoin source code). ABS is an indicator that shows whether the block is filled. One transaction needs to use the storage 4 times the simple message. After the update ‘Segregated Witness’ on 8/24/2017 [20], the extra 3 times does not calculate in Block size, which expand TPB. Transaction size range and transactions per seconds are the settings we want to change to measure the performance, which turns to be ATS and TPD after statistics. TCT shows how long one transaction is accepted, MS shows the waiting list of transactions.

In the system sight, we want the block to be filled (ABS as large as possible, TPB as many as possible), so we can transport more information at the same time. In client’s view, TCT is the most important QoB, i.e. the less TCT is, the more effective the Blockchain network.

3 Requirements for Blockchain Simulation

In order to realize a Blockchain simulator, there are several requirements we should consider, as shown below:

- **Timing simulation:** The Blockchain’s scale rises with time. Mining time, transaction time and network delay affect the performance of the system a lot.
- **Broadcasting:** In reality, we use multicast to realize IP communication. In the simulation, it turns out to be an ideal situation. So we need to broadcast all the transactions.

- **Event-driven:** A miner can change its status once some event happens during mining. The simulator should be event-driven to fit this pattern.
- **Message Processing:** Once a miner finds a new block, the miner broadcasts its block, at the same time other nodes receive the new block (which packaged as a message) and make its own response. So the software should support message processing.
- **Concurrency:** The Blockchain network is comprised of many doing different but similar things (mining, verifying, receiving or sending messages) at the same time. Concurrency is required as well.

4 Case Study – Simulate Blockchain Using SimPy

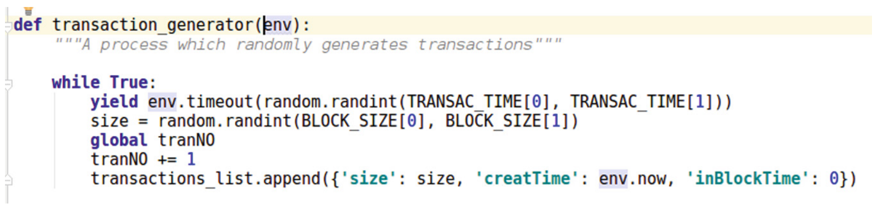
In this section, we demonstrate how to simulate a simple blockchain PoW by leveraging SimPy's capabilities and using the criteria above.

SimPy is a bare simulation API implementation written in Python. In SimPy, the basic simulation entities are processes. These processes can execute in parallel and may exchange Python objects among each other. Most processes include an infinite loop in which the main actions of the process are performed. The simulator is written by SimPy3.0.10 under python 3.6.0.

Computer parameter:
OS: Ubuntu 14.04 64 bit
CPU: Inter(R) Core(TM) i7-7700 CPU@ 3.60 GHz
Memory: 16G

Consider the PoW procedure we have mentioned in Sect. 2.2, our simulator is working with three processes:

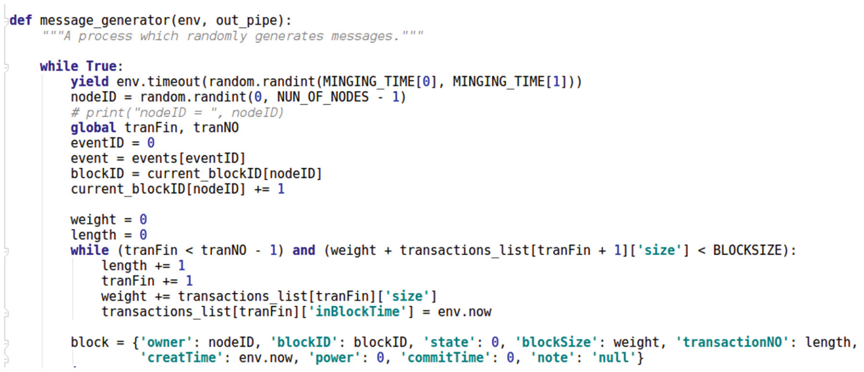
1. Producing transactions. As showed in Fig. 5, we create a transaction with its size and creating time randomly, then set it in the Mempool.



```python
def transaction_generator(env):
    """A process which randomly generates transactions"""

    while True:
        yield env.timeout(random.randint(TRANSAC_TIME[0], TRANSAC_TIME[1]))
        size = random.randint(BLOCK_SIZE[0], BLOCK_SIZE[1])
        global tranNO
        tranNO += 1
        transactions_list.append({'size': size, 'creatTime': env.now, 'inBlockTime': 0})
```

Fig. 5. Transaction process

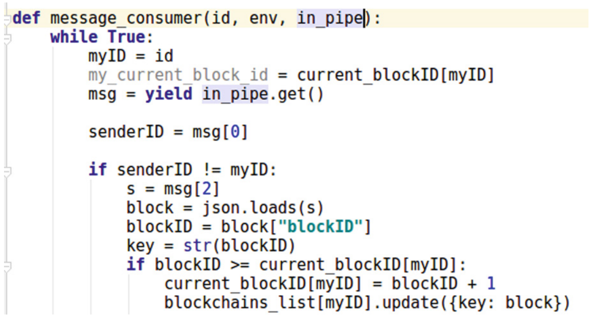
2. Producing blocks. As showed in Fig. 6, we randomize a node as the miner of the block, and then set it mining time randomly. Then calculate the number of transactions could be written into the block, package all these transaction details and then broadcast. Confirm the block before six blocks if it is the branch with biggest computing power, add it into Blockchain (As we didn't consider the situation of orphaned blocks, it is just a confirmation without comparing).



```
def message_generator(env, out_pipe):
    """A process which randomly generates messages."""

    while True:
        yield env.timeout(random.randint(MINGING_TIME[0], MINGING_TIME[1]))
        nodeID = random.randint(0, NUN_OF_NODES - 1)
        # print("nodeID = ", nodeID)
        global tranFin, tranNO
        eventID = 0
        event = events[eventID]
        blockID = current_blockID[nodeID]
        current_blockID[nodeID] += 1

        weight = 0
        length = 0
        while (tranFin < tranNO - 1) and (weight + transactions_list[tranFin + 1]['size'] < BLOCKSIZE):
            length += 1
            tranFin += 1
            weight += transactions_list[tranFin]['size']
            transactions_list[tranFin]['inBlockTime'] = env.now

        block = {'owner': nodeID, 'blockID': blockID, 'state': 0, 'blockSize': weight, 'transactionNO': length,
                 'creatTime': env.now, 'power': 0, 'commitTime': 0, 'note': 'null'}
```

Fig. 6. Block process

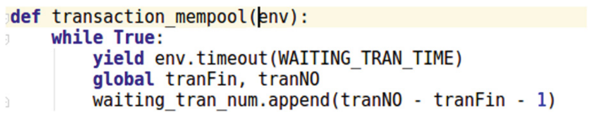
3. Every miner is a process (Fig. 7): once a miner receives a transaction or block, it verifies it. The process adds the block at the end of its own chain and confirm the block before six blocks.



```
def message_consumer(id, env, in_pipe):
    while True:
        myID = id
        my_current_block_id = current_blockID[myID]
        msg = yield in_pipe.get()

        senderID = msg[0]

        if senderID != myID:
            s = msg[2]
            block = json.loads(s)
            blockID = block["blockID"]
            key = str(blockID)
            if blockID >= current_blockID[myID]:
                current_blockID[myID] = blockID + 1
                blockchains_list[myID].update({key: block})
```

Fig. 7. Message process

In order to have a clear look at the mempool, we set a fourth process (Fig. 8) just for recording.



```
def transaction_mempool(env):
    while True:
        yield env.timeout(WAITING_TRAN_TIME)
        global tranFin, tranNO
        waiting_tran_num.append(tranNO - tranFin - 1)
```

Fig. 8. Mempool process

There are several parameters we can change during simulation (Table 1):

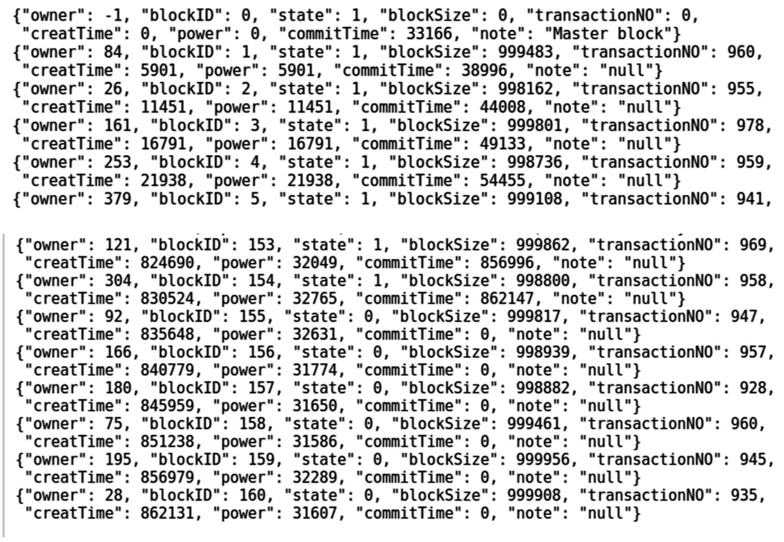
Table 1. Parameters in simulation

Parameters	Description	Default
SIM_TIME	The amount of time that simulation runs	24 h
NUN_OF_NODES	The number of nodes in simulation	512
MINGING_TIME	The rank of time that a block could be mined	[8, 10] min
BLOCKSIZE	The limited size of a single block	1000000B
TRANSAC_SIZE	The rank of size that a single transaction could be	[100, 2000] B

The time in the simulation is calculated by 0.1 s, as the transaction is 1 Byte.

The parameters above configure how the simulator runs. In the real Bitcoin network, most of these parameters are formulated or limited. We can change these parameters in order to find out how it affects the Bitcoin network. In this version of simulator, we ignored network delay and orphan blocks.

With this simulator, we can get some important information about the Blockchain in each node. Due to Blockchain's working method, every node gets a similar but not the same chain. Once we consider the network delay, the chain on different nodes may have a little different. In our case, the only different is commitTime caused by the block creator. One Blockchain in node 0 create with default settings shows below. The first diagram in Fig. 9 shows the first few blocks in the network which includes Master Block, the second diagram shows the latest blocks includes some Blocks haven't been committed yet.



```
{"owner": -1, "blockID": 0, "state": 1, "blockSize": 0, "transactionNO": 0,
 "creatTime": 0, "power": 0, "commitTime": 33166, "note": "Master block"}
{"owner": 84, "blockID": 1, "state": 1, "blockSize": 999483, "transactionNO": 960,
 "creatTime": 5901, "power": 5901, "commitTime": 38996, "note": "null"}
{"owner": 26, "blockID": 2, "state": 1, "blockSize": 998162, "transactionNO": 955,
 "creatTime": 11451, "power": 11451, "commitTime": 44008, "note": "null"}
{"owner": 161, "blockID": 3, "state": 1, "blockSize": 999801, "transactionNO": 978,
 "creatTime": 16791, "power": 16791, "commitTime": 49133, "note": "null"}
{"owner": 253, "blockID": 4, "state": 1, "blockSize": 998736, "transactionNO": 959,
 "creatTime": 21938, "power": 21938, "commitTime": 54455, "note": "null"}
{"owner": 379, "blockID": 5, "state": 1, "blockSize": 999108, "transactionNO": 941,
```

```
{"owner": 121, "blockID": 153, "state": 1, "blockSize": 999862, "transactionNO": 969,
 "creatTime": 824690, "power": 32049, "commitTime": 856996, "note": "null"}
{"owner": 304, "blockID": 154, "state": 1, "blockSize": 998800, "transactionNO": 958,
 "creatTime": 830524, "power": 32765, "commitTime": 862147, "note": "null"}
{"owner": 92, "blockID": 155, "state": 0, "blockSize": 999817, "transactionNO": 947,
 "creatTime": 835648, "power": 32631, "commitTime": 0, "note": "null"}
{"owner": 166, "blockID": 156, "state": 0, "blockSize": 998939, "transactionNO": 957,
 "creatTime": 840779, "power": 31774, "commitTime": 0, "note": "null"}
{"owner": 180, "blockID": 157, "state": 0, "blockSize": 998882, "transactionNO": 928,
 "creatTime": 845959, "power": 31650, "commitTime": 0, "note": "null"}
{"owner": 75, "blockID": 158, "state": 0, "blockSize": 999461, "transactionNO": 960,
 "creatTime": 851238, "power": 31586, "commitTime": 0, "note": "null"}
{"owner": 195, "blockID": 159, "state": 0, "blockSize": 999956, "transactionNO": 945,
 "creatTime": 856979, "power": 32289, "commitTime": 0, "note": "null"}
{"owner": 28, "blockID": 160, "state": 0, "blockSize": 999908, "transactionNO": 935,
 "creatTime": 862131, "power": 31607, "commitTime": 0, "note": "null"}
```

Fig. 9. Blockchain information in node 0

We do some simulations with different settings, the result shows below.

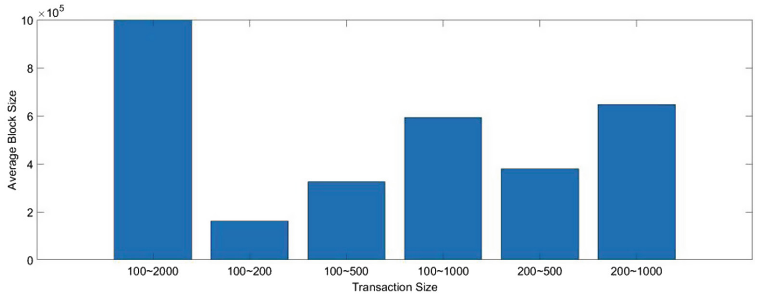


Fig. 10. ABS (change transaction size)

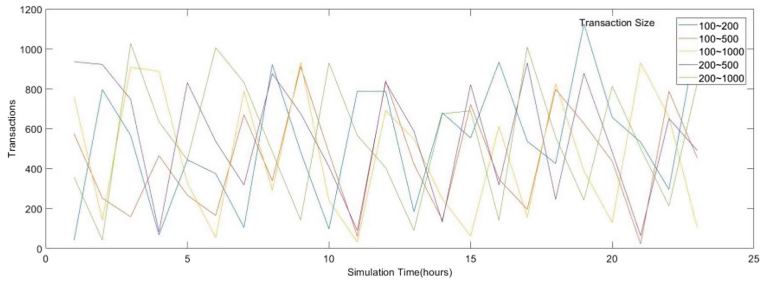
In Fig. 10, as the number of transactions per day and the block size keeps, only when the transaction size is from 100B to 2000B, transactions make the block full. When the transaction is small, more transactions could be recorded in one block. Once there are not enough transactions, the block won't be filled. In real system, mining a block cost a lot, so we want it contain as much information as possible, but not racing.

In Fig. 11(a), it shows details during one day that the mempool size keeps in a low level. The transactions do not need to wait for a long time to be set in a block. Also in Fig. 11(b) and (c), when the transaction per second changes, the mempool size may keep in a low level or go straight high. Once the mempool size goes high, the system will become more and more redundant. Just like in Fig. 13, the transaction commit time rises with time. In the real network, the transaction number depends on different events, which have peak and off-peak time. As well as the transaction size. In further research, we will test the network with huge transactions in a limited time and the ability the network could solve with the stacked Mempool.

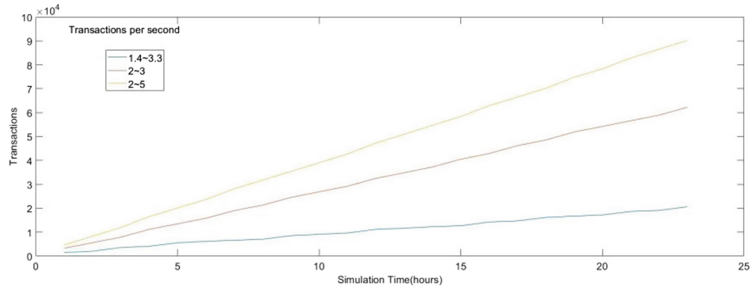
In Fig. 12, the number of nodes has little impact on the Blockchain performance. During our simulation, the node could be up to 20000. Actually, the increase of node number will give the system more computing power, which may cause a shorter mining time. Once the mining time is out of range, the system will improve the difficulty of calculation, which finally keeps the mining time limited. In our simulation, the mining time is randomized is in a range. So the increase of node number just cause some pressure on Memory.

In Figs. 10 and 11(a), we change the same settings. But it shows good performance in Fig. 11(a) which turns to be clients' view, bad performance in Fig. 10 which shows the miners' view. It turns out to be different.

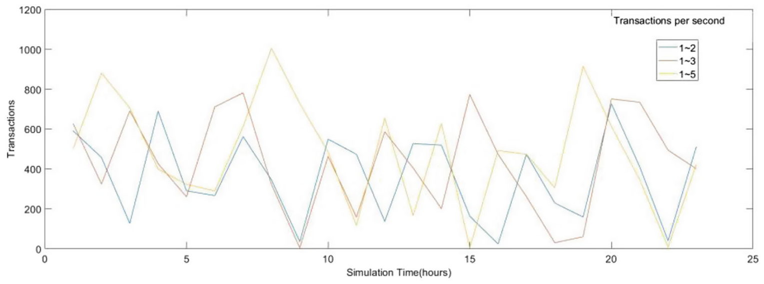
Our simulation can show that different parameters change the behavior of the whole network. In future work, we will learn more about how the parameters affect the behavior and how to find out a best setting which will show good performance in both clients' and miners' view.



(a)



(b)



(c)

Fig. 11. **(a):** Mempool size (change transactions size) **(b):** Mempool size (change transaction per second) **(c):** Mempool size (change transactions per second)

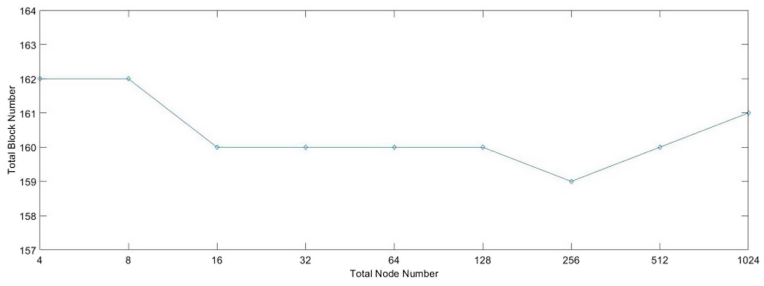


Fig. 12. Total block number (change total node number)

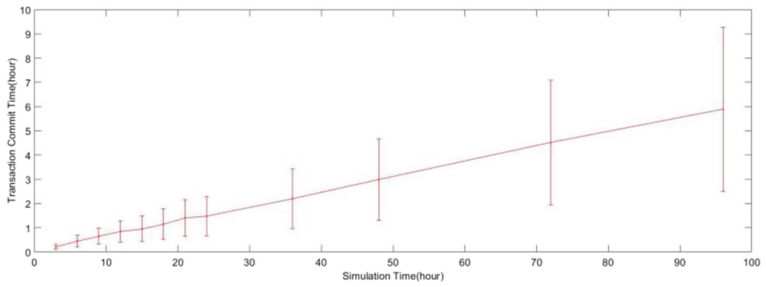


Fig. 13. transaction commit time

With these results, we can made the following observations:

- Our metrics are effective in QoS of Blockchain, we can get a clear view on the performance of the Blockchain.
- Once we jump the mining part and set the mining time in a range, the increase of node number won't influence our simulation till now. But it may cause higher possibility fir nodes to create an orphaned block in the future research.
- Changes in transaction size and transaction per second will show up in the Mempool size. With the rising of Mempool size, the users' experience will become worse because of the long transaction commit time.
- The transaction amount has peak and off-peak time in the real network, so we need a certain amount of Mempool size to keep the block filled and keep the system efficiently. Not too much, not rising all the time, but in a limited size.

5 Related Work

Rajitha et al. [2] used architectural performance modelling and the same incident management exemplar for this approach provided by Weber et al. [4] to measure the latency arising from the Blockchain-related factors, such as the configuration of the number of confirmation blocks and inter-block times. Their management system shows that predictions of median system level response time with a relative error mostly under 10%. Their approach could be used in the design of blockchain-based systems. Their model mostly regards as API, so they ignore the Blockchain mining network, node communication, or consensus algorithm. They modeled the resource and performance characteristics of a local node as a component. In our research, the consensus is one of the main points, which may highly influence the performance in no matter network view or user's view. But their modelling concepts are well-aligned with component-based development and support the re-use of constructed models and components.

Gobel et al. [5] developed two Blockchain simulators, based on the DESMO-J simulation framework [7]. They studied the effect of communication delay in Bitcoin Blockchain under a 'selfish-mine' strategy. First, they use a simplified Markov model that tracks the contrasting states which includes a small amount of dishonest (selfish) miners to establish that the use of block-hiding strategies, such as selfish-mine, which may cause the increase of orphan blocks. Then they use a spatial Poisson process model to study values of Eyal and Sirer's parameter γ, to find out the proportion an honest miner mine a block which previous block is mined by an honest miner. Finally, they use discrete-event simulation to study the behaviour of a network of Bitcoin miners, which includes selfish-mine action under a network with communication delay between miners. Their study found out that if dishonest miners are exist, the performance of both honest and dishonest miners will become worse, the system also can monitor the production of orphan blocks to find out the behavior of selfish-mining. We haven't mention selfish-miner yet, we could consider it in the further research.

Grevais et al. [6] provided a complete Bitcoin simulator written by NS2. They introduce a novel quantitative framework to analyze the security and performance implications of various consensus and network parameters of PoW blockchains. They find some method to fight against or limit double-spending and selfish mining under their framework, by changing the basic settings such as network propagation, different block sizes, block generation intervals, information propagation mechanism, and the impact of eclipse attacks. Under their framework, they can find a balance between performance and security in Blockchain Network. Compared with our simulator, their simulator is kind of complete but facing the problem on the number of nodes. All the simple node's location and its network delay should be defined individual. And it simulates the procedure of mining as well, which cost a lot of performance. When the node number rises, the simulator takes a bad respond.

Goswami [8] discuss the factors that make Block-chain largely non-scalable. They provide the simulator written by java. This research delves into the scalability issue of blockchains and provides a comparative analysis of several blockchain parameters with real time data. It delves into the factors that make block chains largely non-scalable. This is done by the simulation of blockchain. It then addresses the various mechanisms that can be employed to resolve this limitation through measuring the differences between the simulator and real time scenarios. Their simulator which simulated the PoW work without node communication, just finish the work in one client. It's effective but getting troubles in combination with real network.

6 Conclusion

In this paper, we collect and define a number of key performance metrics to quantify the Quality of Blockchain (QoB). We also use a simple simulation tool to simulate a simplified Blockchain Proof of Work (PoW) protocol within different arguments and our observations. The results shows its relation between the basic settings and the Quality of Blockchian. It shows that it is possible and practical to use a simulation approach to study Blockchain networks with different network sizes and protocols. The next step of our work is to make network delay which may cause orphan block into consideration, and then try some other Consensus such as DPoS, PBFT (Fabric) and Tangle (IOTA).

References

1. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system (2008)
2. Rajitha, Y., Staples, M., Weber, I.: Predicting latency of blockchain-based systems using architectural modelling and simulation. In: 2017 IEEE International Conference on Software Architecture (ICSA 2017), pp. 253–256 (2017)
3. Weingartner, E., Vom Lehn, H., Wehrle, K.: A performance comparison of recent network simulators. In: 2009 IEEE International Conference on Communications (ICC 2009), pp. 1–5 (2009)
4. Weber, I., Xu, X., Riveret, R., Governatori, G., Ponomarev, A., Mendling, J.: Untrusted business process monitoring and execution using blockchain. In: La Rosa, M., Loos, P., Pastor, O. (eds.) BPM 2016. LNCS, vol. 9850, pp. 329–347. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45348-4_19
5. Göbel, J., et al.: Bitcoin blockchain dynamics: the selfish-mine strategy in the presence of propagation delay. Perform. Eval. **04**, 23–41 (2016)
6. Gervais, A., et al.: On the security and performance of proof of work blockchains. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 3–16 (2016)
7. Page, B., Kreutzer, W.: Simulating discrete event systems with UML and JAVA. Environ. Sci. Pollut. Res. **13**(6), 441 (2006)
8. Goswami, S.: Scalability analysis of blockchains through blockchain simulation. University of Nevada, Las Vegas (2017)
9. Fairley, P.: Blockchain world-feeding the blockchain beast if Bitcoin ever does go mainstream, the electricity needed to sustain it will be enormous. IEEE Spectr. **54**(10), 36–59 (2017)
10. Augot, D., Chabanne, H., Chenevier, T., George, W., Lambert, L.: A user-centric system for verified identities on the Bitcoin blockchain. In: Garcia-Alfaro, J., Navarro-Arribas, G., Hartenstein, H., Herrera-Joancomartí, J. (eds.) ESORICS/DPM/CBT -2017. LNCS, vol. 10436, pp. 390–407. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-67816-0_22
11. Christidis, K., Devetsikiotis, M.: Blockchains and smart contracts for the internet of things. IEEE Access **4**, 2292–2303 (2016)
12. Decker, C., Wattenhofer, R.: Information propagation in the Bitcoin network. In: 2013 IEEE Thirteenth International Conference on Peer-to-Peer Computing, pp. 1–10 (2013)
13. Eyal, I., et al.: Bitcoin-NG: a scalable blockchain protocol. In: NSDI, pp. 45–59 (2016)
14. Gervais, A., et al.: Tampering with the delivery of blocks and transactions in Bitcoin. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, pp. 692–705 (2015)
15. Kogias, E.K., et al.: Enhancing Bitcoin security and performance with strong consistency via collective signing. In: 25th USENIX Security Symposium, pp. 279–296 (2016)
16. Kosba, A., et al.: Hawk: the blockchain model of cryptography and privacy-preserving smart contracts. In: 2016 IEEE Symposium on Security and Privacy, pp. 839–858 (2016)
17. Pazmiño, J.E., Rodrigues, C.K.S.: Simply dividing a Bitcoin network node may reduce transaction verification time. SIJ Trans. Comput. Netw. Commun. Eng. **3**(2), 17–21 (2015)
18. The Ethereum community. Ethereum White Paper, July 2015. https://github.com/ethereum/wiki/wiki/WhitePaper
19. Popov, S.: The Tangle. IOTA White Paper (2017). https://iota.org/IOTA_Whitepaper.pdf
20. https://en.wikipedia.org/wiki/SegWit

A Design of Digital Rights Management Mechanism Based on Blockchain Technology

Zehao Zhang and Li Zhao(✉)

Research Institute of Information Technology, Tsinghua University, Beijing, China
zhangzeh16@mails.tsinghua.edu.cn, zhaoli@tsinghua.edu.cn

Abstract. Digital rights management (DRM) has been widely used in digital content protection nowadays and has made a great contribution to the protection of digital content. However, the traditional DRM technology has several disadvantages such as centralization, non-transparency of copyright information and transaction information. Centralized servers are vulnerable to be attacked and opaque information is not user-friendly. The blockchain technology which has emerged in recent years has the advantages of decentralization, collective maintenance, security, and reliability. It can be a great solution to the above problems. In this paper, we propose a design of DRM Mechanism based on blockchain technology. We record the copyright transaction information and license information on the blockchain to make information transparent and safe. We use smart contact to ensure the reliability of copyright transaction and issue licenses automatically, which eliminates the need for centralized servers to verify identities and issue licenses. Our mechanism allows copyright owners to set prices for different content usage rules as they wish. Customers choose the usage rules they would like to purchase flexibly. We also design a blockchain based license structure, which is close to the current DRM standards and easy to be promoted.

Keywords: DRM · Blockchain technology · License · Transaction
Smart contract

1 Introduction

With the rapid development of internet technology, the global information process is promoted continuously. The application of information technology has been expended to all the fields, so that digital content can spread quickly and widely all the time. Due to its openness, digital features, content is facing a significant risk of being malicious disseminated, copied, tampered at any time, which has led to higher demand for digital rights management. Digital Rights Management (DRM) [9] refers to the realization of system solutions through the use of information security technology to ensure legitimate users' normal use of digital content. It also protect the legitimate income of producers and owners of

S. Chen et al. (Eds.): ICBC 2018, LNCS 10974, pp. 32–46, 2018.
https://doi.org/10.1007/978-3-319-94478-4_3

digital media content. After the copyright infringement problem occurs, it can identify the authenticity of the copyright information and make a correct determination of copyright ownership. Many international organizations and companies have formulated DRM standards and these DRM standards have made a great contribution to protecting digital rights.

However, traditional DRM technology have several disadvantages because of its centralization. First of all, the transaction of copyright is supported by centralized server, and the issuing of licenses needs centralized server too. Once the centralized server is attacked, the service is in a great risk of paralysis. Secondly, the specific information of copyright and transaction is not public to users so it is not transparent. Individual content producers must sell their content to large video site or media platform to seek for DRM protection. What's more, the identities in traditional DRM technology need to be verified through multiple interactions, which is cumbersome and inconvenient.

To solve the problems above, the blockchain technology is an excellent choice. Blockchain technology is the core supporting technology of digital cryptocurrency system represented by Bitcoin. It originated from the foundational paper published by Satoshi Nakamoto in 2008 [1]. Blockchain is a kind of decentralized shared ledger that assembles data blocks into specific data structures in a chain manner in chronological order and it guarantees irreversibility and unforgeability in a cryptographic way. Blockchain uses encrypted chain block structures to validate and store data, uses distributed node consensus algorithms to generate and update data, and uses automated script code to program and manipulate data. Blockchain has decentralized, time-sensitive data, collective maintenance, programmable, secure and trusted features. In terms of the application of blockchain technology, Bitcoin realized a payment method which does not require any third-party financial institution for the first time. Ethereum [2] added smart contracts based on electronic cash technology, making both financial and non-financial agreements intelligent. There are many applications and platforms [3–5] based on blockchain technology that devote to making blockchain technology step into our lives.

The purpose of this paper is using the decentralized, safe and credible characteristics of blockchain technology to improve the traditional DRM technology so as to make up for the technical defects mentioned in the previous article. The contribution of this paper is as follows: (1) We record copyright transactions on the blockchain to make the transaction information safe, reliable and transparent. We have designed a mechanism that allows copyright owners to set prices for different content usage rules flexible. Consumers are free to choose the purchase rules they use. At the same time we use smart contracts to ensure the reliability of the transaction. There is no need for interactive information between copyright owners and consumers. The blockchain technology based DRM mechanism we proposed provides a more credible means of protecting copyrights for individual content producers, and the authorization of copyrights is also extremely scalable. (2) Our license information is recorded on the blockchain, and the license information is safe and transparent as well. We use smart contracts to issue licenses automatically. In this way, we eliminate the need for centralized servers to issue licenses, and also eliminate the need for interactive verification between user and server identities. We simplify the process. (3) We designed a license structure suitable for blockchain, which can be used for automatic packaging and distribution of licenses by smart contracts. At the same time, the designed license structure is close to the current DRM standards, which is beneficial to the promotion of this blockchain-based license structure.

The rest of this paper is organized as follows. Section 2 presents the related works, mainly contains blockchain technology, DRM, and the application of blockchain technology in DRM so far. Section 3 describes the design of license, whereas Sect. 5 describes the design of DRM mechanism based on blockchain technology. The conclusions and future work are in Sect. 6.

2 Related Works

2.1 Blockchain Technology

Blockchain technology is considered to be the fifth type of disruptive innovation as a successor to mainframes, personal computers, the Internet, mobile/social networks [10]. Its features contains decentralization, time-series data, collective maintenance, programmability, security and reliability, etc. In the distributed system where nodes do not need to trust each other. Blockchain technology achieves point-to-point transactions, coordination, and operations. Thus it provides solutions to the problems of high costs, inefficiency, and insecure data storage that exist in centralized organizations. Blockchain technology is the embryonic form of the next-generation Internet. The existing Internet is just an information Internet. Without the centralization of banks and other institutions, the exchange of value cannot be achieved. The next-generation Internet, achieves not only the information communication, but also the value communication, and the cornerstone is blockchain technology. Blockchain technology is expected to achieve a transformation from the current information Internet to the next generation of value Internet, which will completely reshape the human social activities just like what the information Internet has made. The characteristics of blockchain technology make it widely used in digital cryptocurrencies, financial systems, and social systems. Innovative technologies have been used in the financial, medical, education, and Internet of Things fields to solve key issues in the industry. For example, filecoin [6] is a decentralized storage network that turns cloud storage into an algorithmic market which combines blockchain technology and IPFS [7] technology. It provides a good choice for file secure storage. MedRec [8] is a novel decentralized record management system to handle EMRs, using blockchain technology. Their system gives patients a comprehensive, immutable log and easy access to their medical information across providers and treatment sites.

A smart contract is an instance of a computer program that runs on a blockchain which has a unique address. Any user can create a contract by publishing a transaction to a blockchain. Once a smart contract's program code has been created, it cannot be changed and will be executed whenever a message from a user or other contract is received. The behavior of the contract is determined by the publisher while receiving a message. Smart contract can read and write stored files, send messages to other users or contracts. It can also deposit currency into the account balance or send it to other users or contracts. The terms of contracts cannot be changed. Therefore, contracts cannot be changed. The principle of smart contracts is shown in Fig. 1.

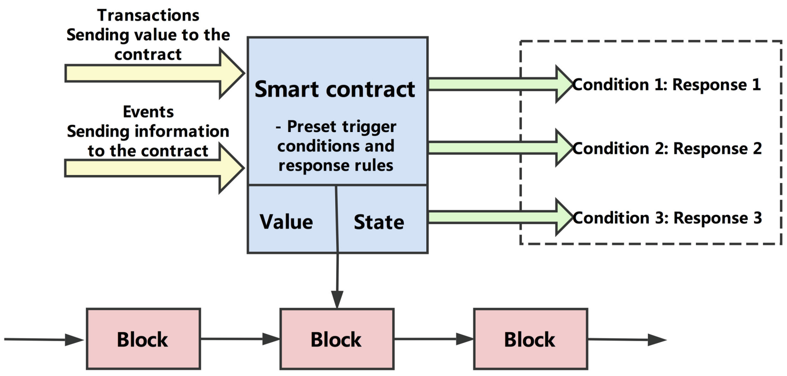


Fig. 1. Smart contract working principle

In this paper, we make use of the security and reliability features of the blockchain technology. We record copyright transaction information and license information on the blockchain, making the information safe and trustworthy. We use smart contracts to issue licenses automatically and also use smart contracts to ensure the reliability of copyright transactions.

2.2 Digital Rights Management

Digital rights management (DRM) [11,12] is a type of management system developed to enable secure distribution, and more importantly, to disable illegal distribution of paid content. DRM technologies are being developed as a means of protection against the online piracy of commercially marketed material. The high-level architecture and major components of a typical DRM system are shown in Fig. 2.

To ensure a great security of digital content, international organizations and companies have formulated many DRM standards. Open Mobile Alliance released the OMA DRM 1.0 standard officially in November 2002 [13,14]. OMA DRM is mainly designed for mobile data services and fixed data services, and it has been updated to version 2.0 now. Marlin DRM was introduced by the United States Intertrust company in 2005, co-founded with four other companies: Panasonic, Sony, Philips and Samsung [15,16]. The goal of Marlin DRM

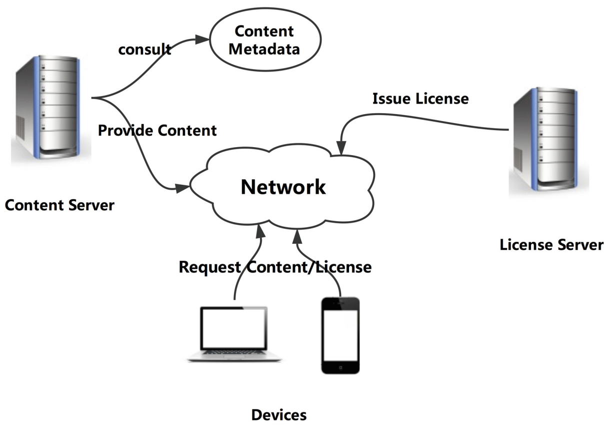


Fig. 2. High-level architecture and major components of a typical DRM system

is to implement a DRM system compatible with a wide range of consumer electronic devices. Google spent heavily to buy the video digital rights management software company Widevine in December 2010 [17]. Widevine DRM helps Google make up for the shortcomings in digital copyright protection and helps Android become the dominant mobile OS today. CCTV and Tsinghua University established China DRM Forum in November 2004 with the support of The State Administration of Radio Film and Television (SARFT). The forum aims to promote the development of DRM in China and protect the rights and interests of every participant in the digital content industry chain to build a sound environment for the development of digital content [18].

DRM has made great progress in the world. However, the traditional DRM technology has several disadvantages because of its centralization. Centralized servers are vulnerable to be attacked and opaque information is not user-friendly. The blockchain technology is thought to be a good solution for these disadvantages. Since the emergence of such a great technology as blockchain, there has not been a set of proper DRM mechanisms combined with blockchain technology. This paper has made an attempt. We hope to promote the development of DRM mechanisms based on blockchain technology. The license structure we proposed is close to current DRM standards, which is beneficial for the promotion of this blockchain-based license structure.

2.3 DRM Based on Blockchain Technology

There have been several papers make trys on combining DRM with blockchain technology. Xu et al. [19] proposed a network media's digital rights management scheme based on blockchain. They hope to provide an important support for the network media platform to build a sustainable development of benign ecological environment. But they only record the transaction information on the blockchain and ignore the copyright information, which makes the design incomplete. The blockchain they used is private clockchain which is not decentralized completely. Fujimura et al. [20] propose a concept for a new rights management system based on the blockchain technology, and clarified problems that occur when they applied the blockchain technology to the rights management system. They record the copyright information on the blockchain, but the transaction between copyright owners and consumers occur out of the blockchain, which means we need a central institution to ensure the copyright purchasing. This feature makes the decentralization not complete. The mechanism proposed in this paper avoids the disadvantages of the above papers. We record both copyright transaction information and license information on the public blockchain. On the one hand, we use the smart contracts to ensure the reliability of copyright purchasing, and record the transactions on the blockchain. We don't need any central institution. On the other hand, we make use of smart contracts issuing licenses automatically, and record the licenses on the blockchain. In this way, both the content providers and the consumers could query and verify the copyright information at any moment, and furthermore, no one could tamper the copyright information.

3 Design of License

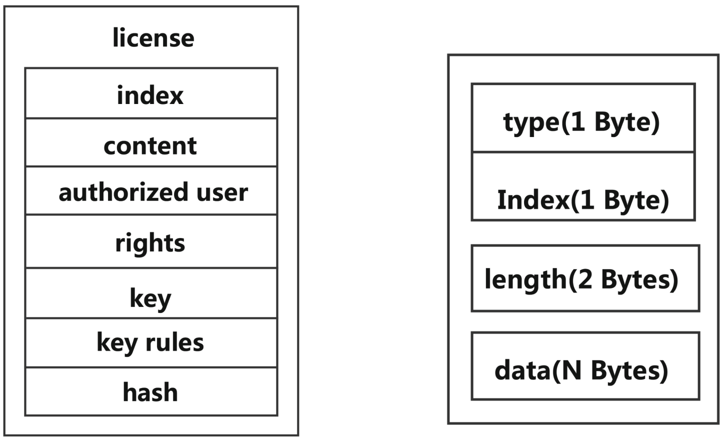
We make use of smart contracts issuing licenses automatically. After the content provider sets the price of the content according to the rules, the consumer who wants to purchase the content makes price estimation according to his own needs. Then the consumer sends information to the smart contract to purchase. The contract will package the license according to the needs of the purchase and send it to the consumer account.

3.1 License Structure

License consists of elements such as content, authorized accounts, rights, key, key usage rules, and hash values, which is shown in Fig. 3(a).

When the license is issued, it is composed of the license indexing unit and one or more subsequent basic units. All the units except the indexing unit are basic units. The elements described in the logical structure of the license are described as below:

(1) License Indexing Unit: The license indexing unit is the first unit of the license. The license indexing unit includes the version, license ID, and basic unit number. The license ID is the unique identity of a license. The basic unit number is the number of basic units.
(2) Content: Content is a digital thing, such as pictures, text, audio, video, etc. The license stores a unique identifier for the content, contentID.
(3) Authorized Account: The authorized account is the owner of the rights to the specified content. It is the account of the consumer who purchased the copyright in the blockchain.
(4) Rights: Rights refer to the rights of using content, such as playing, storing, etc.



(a) License structure (b) License unit encoding structure

Fig. 3. License

(5) Key: The key refers to the key information contained in the license, such as the encryption algorithm and key data. The key data is encrypted by the consumer's public key and can only be decrypted with the consumer's private key.
(6) Key usage rules: Key usage rules define how consumers can use keys, including the start time, the end time, time period, number of times and other relevant rules. The license is generated in the form of a purchase chosen by the consumer, and the consumer can only use the key according to the rules.
(7) Hash: The hash refers to the hash of all previous units and is used to verify the integrity of the license data. When consumers receive a license, they should check the hash first.

3.2 License Unit Encoding Structure

Both the license indexing unit and the basic unit in the license are composed of three parts: unit identification, length, and data. The unit encoding is as shown in the Fig. 3(b).

The unit identification consists of 2 bytes, including the type and index. The first byte is the type, and the second byte is the index of the unit in the license, which is used to support the segmented transmission of the license. The length section is the length of the actual data information of the unit and is represented by two bytes. The data section is the actual data of the unit and is represented by N bytes.

4 Copyright Transaction

The content providers and the customers can choose to be a peer of our blockchain or choose to link to one trusted peer our blockchain platform provides. The peers need to be listening to messages from the smart contract all the time. The copyright transaction is shown in the Fig. 4.

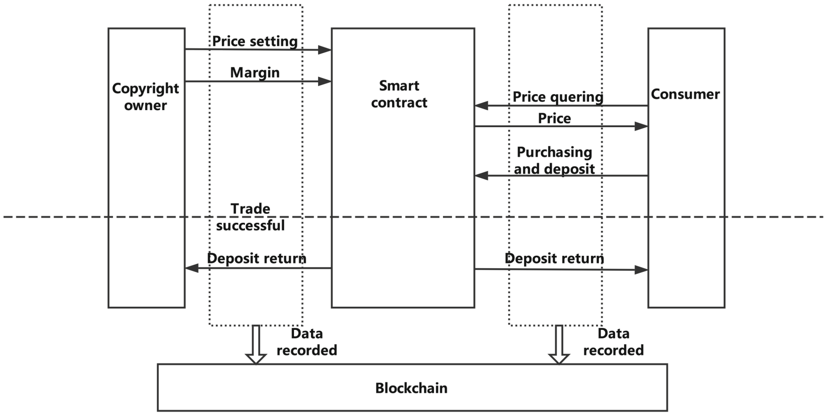


Fig. 4. Copyright transaction

4.1 Price Settings

We have designed a flexible license authorization system that allows copyright owners to set a price for their content. As the license structure mentioned in Sect. 3.1, the rights and key usage rules are designed for authorization. The copyright owners can set the price of the rights for their content, such as the price for playing the content, the price for storing the content, etc. They can also set the price corresponding to the key usage rules for their content. For example, the content can be used by time, number of times, or time period, etc. Corresponding to different key usage rules, copyright holders need to pass different parameters to the smart contract. If they want contents to be used by time, the copyright owners need to set the price required by the unit time. To be used by the number of times, the copyright owners need to set the price for each time. To be used by the time period, the copyright owners need to set the time period and the price corresponding to this time period. The price between the right and the key usage rules is in additive relation. The copyright holder's setting of the price will be written into the transaction information between the node and the smart contract, so that it will be written into a block and recorded in the blockchain.

When consumers choose to purchase a product, they search for the optional rights first and key usage rules corresponding to the content through the smart contract. Then they need to choose the rights and key usage rules according to their own needs. The smart contract will return the price to be paid. After the consumers have paid successfully, the smart contract will package and send the licenses to the consumers automatically according to the demand. Among them, consumer information such as content inquiries and purchases, as well as license information sent by smart contracts, will be written into the transaction information between the nodes and the smart contract, which will be written into a block and recorded in the blockchain.

In this way, buyers and sellers can trade flexibly according to their interests. At the same time, all trading information and copyright information are recorded in the blockchain, and both sides can check at any time. The use of smart contract also eliminates the direct interaction between copyright owners and consumers.

4.2 Trade Pledge

Copyright trading is guaranteed by smart contracts. When a copyright owner publishes his content to a smart contract, he will pay a certain amount of deposit. This deposit ensure that the trade can be proceed smoothly. The copyright trader can add deposit to his account at any time. Consumers are required to pay a 5% premium over the content price when choosing to purchase the copyright of a content. After the license is sent to the consumer and checked successfully, the consumer will feed back information that the content can be consumed successfully to the smart contract. At this time, the smart contract will return the 5% deposit of the content price to the buyers and sellers of the trade. In this way, we can ensure the reliability of the trade. Once one side destroys the trade, the deposit will not be returned to this side. With the support of blockchain technology, trades guaranteed by smart contracts are safe and credible.

5 Design of Blockchain-Based DRM Mechanism

5.1 Blockchain-Based DRM Mechanism Prototype

Figure 5 shows the blockchain-based DRM mechanism prototype. Content providers, consumers, and advertisers use blockchain clients to interact with nodes. Nodes are responsible for basic functions such as interaction with smart contracts and blocks production. The digital content can only be used by the client. The rules and keys used are obtained from the license. This avoids the fraudulent use of digital content.

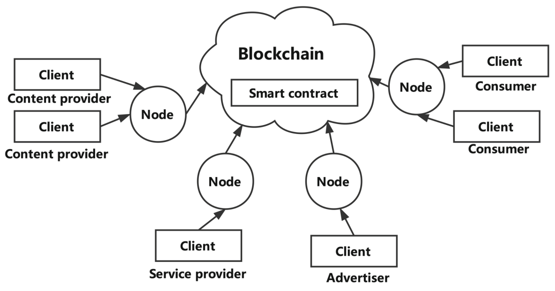


Fig. 5. Blockchain-based DRM mechanism prototype

5.2 DRM Mechanism Process

The business process of the blockchain-based DRM mechanism is shown in Fig. 6.

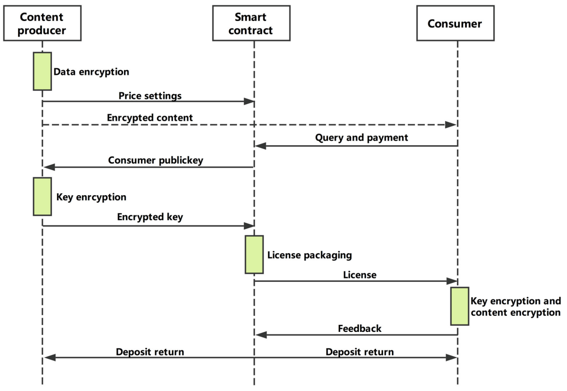


Fig. 6. The business process of the blockchain-based DRM mechanism

Step 1 A content producer encrypts the digital content and obtains the encrypted data and key data. Then the content producer keeps the key data for the follow using.

Step 2 The content producer, also called as copyright owners, interacts with smart contracts to set the price of the corresponding content rights and key usage rules.

Step 3 A consumer obtains the encrypted content he wants to use through an off-chain way. The consumer can obtain content through a centralized video site, or through a peer-to-peer network such as IPFS.

Step 4 The consumer interacts with the smart contract to find out how he/she can use the content and how much he/she have to pay for their needs. Then the consumer chooses the rights and the key usage rules he/she would like to pay.

Step 5 The smart contract sends the consumer's public key information to the content producer. The content producer uses the consumer's public key to encrypt the key data which was used to encrypt the content, and sends the encrypted key data to the smart contract.

Step 6 The smart contract packages the index, content, authorized accounts, rights, keys, key usage rules and hash together, according to the rights and key usage rules chosen by the consumer, as the way mentioned in Sect. 3.1. Then the smart contract sends the license to the consumer.

Step 7 After the consumer obtains the license, the consumer uses his private key to decrypt the encrypted key data according to the key encryption algorithm. Then the plaintext of key data is obtained. At this time, the consumer could use the key to decrypt the encrypted content to get the plaintext of content.

Step 8 The consumer feedbacks the message to the smart contract that the license can be used correctly. Then the copyright transaction is completed. The buyer and seller's deposit is returned to their respective accounts. The entire blockchain-based DRM process is completed.

5.3 Storage Management

Storage management mainly consists of three parts:

(1) Copyright Trading and License Information Storage: In essence, blockchain is a distributed database that can be used to store and transmit dispersed data. When we implement the DRM mechanism based on blockchain technology, we can store copyright transaction information and license information on the blockchain. In this way, users can check their transaction records and authorization information or authorized information at any time. Moreover, because this information is stored on the blockchain, it also has features that cannot be tampered with forgeries. The block structure is shown in Fig. 7. With using the hash chain and time signature technology, we can build a proof of these data. The license on the smart contract is temporarily stored in a part of the memory of the smart contract after being assembled. After the license's being sent to the consumer, the occupied memory is released.

(2) Content storage: After the content producer completes content creation, the content producer encrypts the content to obtain the encrypted content data, content ID and key. The encrypted content data is stored on a centralized video site or on a decentralized peer-to-peer network for consumers' downloads. The key is stored by the content producer himself, waiting to be encrypted with the consumer's public key later. Content IDs are published by content producers on smart contracts for consumers' inquiries.

(3) Account information storage: Both content producers and consumers need to register accounts on the blockchain platform, and need to store certain virtual currency in the account to complete subsequent copyright transactions. The smart contract stores the account information by mapping. The accounts are divided into two categories. The content producer's account

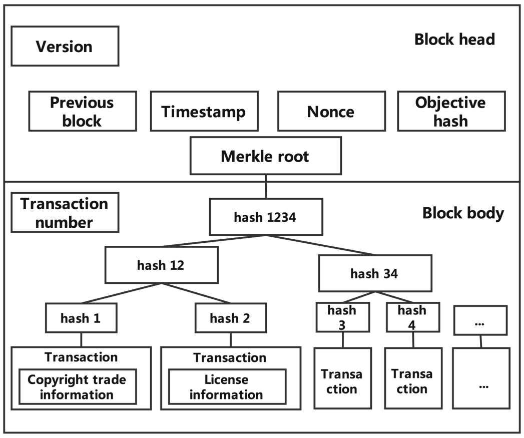


Fig. 7. Block structure

needs to store information such as address, virtual currency balance, content ID, price settings, and so on. Consumer accounts need to store virtual currency balances, public keys, addresses, and so on.

5.4 Copyright Information Management

Recording copyright information on the blockchain enables low-cost copyright confirmation of digital content. Individual content creators can publish content on the blockchain that implements our DRM mechanism easily. So that they don't have to sell content rights to a large media company or a media platform. Content creators can price their own content flexibly, set copyright rules for their content, and gain greater autonomy. At the same time, copyrights are recorded on the blockchain and cannot be tampered, which also solves the problem of copyright disputing fundamentally. Consumers can view published content and price information at any time, and purchase copyrights according to their interests for content using. Once the copyright is purchased, both buyers and sellers cannot make changes. Consumers don't need to worry about the fact that the purchased copyright will be tampered with or cancelled. The information in the blockchain is transparent.

Large online media platform can also add copyrighted digital products on the blockchain under our new DRM mechanism too. With blockchain technology, the platforms can track and monitor the use of copyright. At the same time they can record the consumer's user behavior and analyze the data to find out which products are more popular with consumers and introduce more high-quality digital products. The most important thing is that the decentralized blockchain systems are harder to be broken than the centralized servers.

We believe that the blockchain-based DRM mechanism can make the living environment of digital content more fair and more harmonious. Due to the convenience and permanency of copyright confirmation, we can also develop more application scenarios such as the copyright notarization, crowdfunding, certificate authentication and so on.

5.5 Transaction Management

We record all the information that users interact with smart contracts on the blockchain, including the content publisher's releasing information on the content, the price setting information on the content copyright use rules, the consumer's inquiry on the content price, the consumer's purchase of the digital content on the preference, and the smart contract feedback margin information. These transaction information is recorded in the blockchain to ensure that transactions can be tracked and queried at any time. Once the transactions are created, they will be broadcasted on the blockchain network. The nodes in the blockchain create blocks for these transactions and calculate the block's hash. The node that acquires the right to credit will link the new block to the blockchain, thus saving the information permanently. Using blockchain to record transaction information can greatly reduce transaction costs and improve transaction efficiency. Each node can view and check all transaction information at any time. The blockchain's consensus algorithm ensures that records cannot be forged. The blockchain's digital signature technology also reduces fraudulent transactions. This kind of management can achieve efficient and secure transactions and thus promote the healthy development of digital content protection environments.

5.6 Potential Business Benefits

Using the proposed blockchain-based DRM mechanism model, we can reasonably coordinate the relationship between content providers, service providers, consumers and advertisers. Firstly, content providers, both individual content creators and content production companies, have a clear understanding of how much consumers love their content. Secondly, smart contracts ensure the unforgeable nature of transactions and copyright information, and consumers will feel comfortable using them. For service providers, blockchain technology gives value to digital content. For example, if a digital content only sells 10,000 copies, the lifecycle of each content can be tracked using blockchain technology. The owner of the licenses can be queried on the blockchain, we could also provide transfer function of license, so that the digital content has a collection of meaning. For advertisers, using our model can clearly see the number of digital content using with advertisement, so as to know the effectiveness of advertising. They do not have to doubt the authenticity of the data, so that advertising costs the value for money they paid. We believe our model will provide a healthy development environment for the production, dissemination and consumption of digital content.

6 Conclusion and Future Work

In this paper, we designed a new DRM mechanism based on blockchain technology. We use the decentralized, safe and credible characteristics of blockchain technology to make up for the shortcoming of centralized traditional DRM technology. We record copyright transactions and license information on the blockchain, which makes them safe, reliable and transparent. At the same time we use smart contracts to ensure the reliability of the transaction and the issuing of licenses. There is no need for interactive information between the copyright owner and the consumer, and we don't need centralized license server to issue licenses either. In this way, we simplify the process and save the costs. The license structure based on blockchain technology we proposed is close to the current DRM standards, which is suitable for promotion.

One defect of our work may be the peers of our blockchain platform have to be high-powered to deal with high concurrent key acquisition. We will work to solve or weaken this problem in the future. The future direction of this work is the realization of this new blockchain based DRM mechanism. Considering the need for high concurrency, we plan to make a try on EOS blockchain for the first time. We will test the performance of this new mechanism and improve the performance. Then we aim to promote this new DRM mechanism for a better protection of digital content.

References

1. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system. Consulted (2008)
2. Ethereum White Paper. https://github.com/ethereum/wiki/wiki/White-Paper. Accessed 30 Mar 2018
3. Bitshares Forum. https://bitsharestalk.org/index.php. Accessed 30 Mar 2018
4. EOS.IO Homepage. https://eos.io. Accessed 30 Mar 2018
5. Hyperledger Fabric Homepage. https://www.hyperledger.org. Accessed 30 Mar 2018
6. Filecoin Homepage. https://filecoin.io. Accessed 30 Mar 2018
7. Benet, J.: IPFS - Content Addressed, Versioned, P2P File System. Eprint Arxiv (2014)
8. Azaria, A., Ekblaw, A., Vieira, T., et al.: MedRec: using blockchain for medical data access and permission management. In: International Conference on Open and Big Data, pp. 25–30. IEEE (2016)
9. Ru, Z., Yu, Y., Xiao, Z.: Information security professional series: digital rights management. Beijing University of Posts and Telecommunications Press, Beijing (2008)
10. Yong, Y., Feiyue, W.: Development status and prospect of blockchain technology. J. Autom. **42**(4), 481–494 (2016)
11. Van Tassel, J.: Digital Rights Management: Protecting and Monetizing Content. Focal Press, Waltham (2016)
12. Zhang, Z., Fan, K.: Digital rights management and security technology, pp. 96–106. National Defense Industry Press, Beijing (2013)
13. Irwin, J.: Digital rights management: the open mobile alliance DRM specifications. Inf. Secur. Tech. Rep. **9**(4), 22–31 (2004)

14. Choi, J., Aiken, W., Ryoo, J., et al.: Bypassing the integrity checking of rights objects in OMA DRM: a case study with the MelOn Music Service. In: Proceedings of the 10th International Conference on Ubiquitous Information Management and Communication, p. 62. ACM (2016)
15. Keoh, S.L.: Marlin: toward seamless content sharing and rights management. IEEE Commun. Mag. **49**(11), 174–180 (2011)
16. Srinivas, T.S., Narasimha, V.B., Puroshothammam, M.E.: Analysis of interoperability services of various DRM schemes and associations with Marlin scheme. Indian J. Sci. Technol. **10**(17) (2017)
17. Widevine home page. http://www.widevine.com. Accessed 30 Mar 2018
18. China DRM Forum. China Radio, Film and Television Digital Management (DRM) Technology White Paper. China Radio, Film and Television Technology Division (2011)
19. Xu, R., Zhang, L., Zhao, H., et al.: Design of network media's digital rights management scheme based on blockchain technology. In: IEEE, International Symposium on Autonomous Decentralized System, pp. 128–133 (2017)
20. Fujimura, S., Watanabe, H., Nakadaira, A., et al.: BRIGHT: a concept for a decentralized rights management system based on blockchain. In: IEEE, International Conference on Consumer Electronics, Berlin, pp. 345–346. IEEE (2016)

InfiniteChain: A Multi-chain Architecture with Distributed Auditing of Sidechains for Public Blockchains

Gwan-Hwan Hwang[1,2](✉), Po-Han Chen[1], Chun-Hao Lu[1], Chun Chiu[1], Hsuan-Cheng Lin[1], and An-Jie Jheng[1]

[1] TideTime Sun Limited, Kowloon, Hong Kong
[2] Department of Computer Science and Information Engineering, National Taiwan Normal University, Taipei, Taiwan
gwanhwan.hwang@gmail.com

Abstract. InfiniteChain proposes an all-new type of distributed auditing as well as a method for multi-chain operation that overcomes bottlenecks encountered thus far by state-of-the-art blockchain technologies and their implementation in commercial applications. Transactions are first processed outside of the main chain and index Merkle trees and distributed auditing are then employed to perform fraud proofs for these transactions in the sidechain. Its advantages include high bandwidth of transactions, protection of transaction privacy, and fusion with existing centralized commercial scenarios. The implementation and experimental results demonstrate the feasibility of the proposed system.

Keywords: Blockchain · Sidechain · Distributed auditing

1 Introduction

Bitcoin is a cryptocurrency with decentralized controls that was created in 2009. The blockchain technology it is based on is now widely accepted and used in many industries. Apart from becoming an internationally recognized currency, people are now hoping that this shared value system will enable the development of decentralized applications (Dapp) in each industry based on blockchain technology.

In addition to its use in cryptocurrency transactions, the advantages of blockchains, such as decentralized information verification and resistance to tampering, have been noted by various fields. Key applications include value registry, value web, and value ecosystem services. Industries with related applications include logistics, financial systems, medical records, the collection and verification of data in the Internet of Things (IoT), supply chain management, stocks or options trading, social networking software, electronic patient records, micropayment/mobile payment systems, asset transactions, and distribution of digital products. People are hoping that blockchains will be able to play the role of trusted machines in the operation of such systems. Keeping a detailed record of related information and solving information asymmetry problems will enable a trusted record to be established. In the use cases mentioned above, large amounts of information will need to be recorded on the blockchain.

S. Chen et al. (Eds.): ICBC 2018, LNCS 10974, pp. 47–60, 2018.
https://doi.org/10.1007/978-3-319-94478-4_4

Nevertheless, blockchain technology has encountered bottlenecks in the course of its development. If these cannot be overcome, it will be difficult for the blockchain to be fully implemented in the different application scenarios mentioned above. Each of these problems will be explained below.

The first problem is insufficient blockchain bandwidth. Blockchain's decentralized operation is dependent on Internet users worldwide for its maintenance and use. Any user can therefore use block transactions to exchange cryptocurrency, write smart contracts, or record information. Bitcoin and Ethereum are, however, limited to no more than 7 and 25 transactions per second (TPS) [1], respectively. If no technology available can place large numbers of transactions on the blockchain, it will simply not be practical to solve information asymmetry by storing transactions on the blockchain.

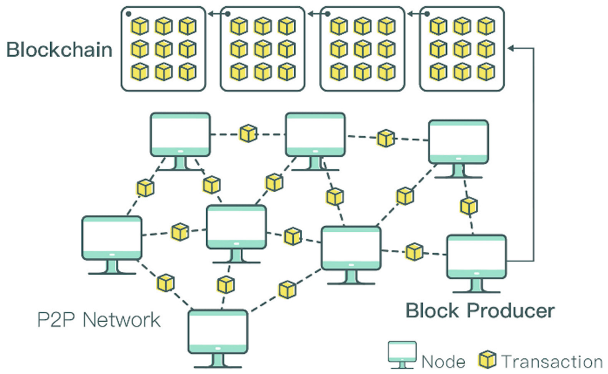


Fig. 1. A public blockchain

Public blockchains are generally considered to have a global consensus[1]. As shown in Fig. 1, the decentralized operation of the blockchain basically uses a consensus protocol such as Proof of Work (PoW) or Proof of Stake (PoS) to obtain or select a block producer from participating nodes. The block producer then collects transactions through a Peer-to-Peer (P2P) network and records these transactions in a single block within the blockchain using electronic signatures and a hash function. All nodes in the public blockchain participating in the consensus protocol must continuously update any changes to the data in the blockchain as well as obtain transactions that ordinary users want to place on the blockchain. Large amounts of information must therefore be exchanged over P2P networks, thus making it impossible to increase transaction bandwidth.

A "private blockchain" or "consortium blockchain" is a method that attempts to solve the problem of insufficient transaction bandwidth. The number of participating blockchain nodes is limited to facilitate rapid propagation and the use of special

[1] Bitcoin and Ethereum have between 8,000–10,000 participating nodes at any given time, many of which are also mining pool nodes with massive processing power.

consensus protocols (e.g., all types of PoS, BFT, and PoA), which contribute to speeding up the selection of block producers. There is obviously a big credibility gap between private blockchains and public blockchains. The core philosophy of a decentralized system is to reduce the access threshold and remove restrictions on participating nodes so that no monopoly on trust machines can be formed. In a private chain, the smaller number of nodes increases vulnerability to 51% attacks[2] and prevents global consensus.

Some public chain developments intend to increase speed by adopting an architecture with a smaller number of nodes, but that leaves them vulnerable to 51% attacks as well as Distributed Denial-of-Service Attack (DDoS) attacks that result in the stoppage of the entire blockchain.

The second problem is insufficient blockchain payload space. As described previously, in each type of system the public blockchain plays the role of the trust machine. As large amounts of transaction records are pushed onto the blockchain, the amount of data on the blockchain will rapidly increase within a short amount of time. Depending on the consensus model, full nodes of the blockchain must store every block on the blockchain and the transactions they contain. The consensus protocol of Bitcoin, for example, restricts the growth in blockchain capacity to around 70 GB per year. In the absence of such restrictions, the propagation and storage of blocks becomes a major problem. This situation is also known as "blockchain bloat." VISA reported that it generated a total of 92.064 million payment transactions in 2015. If translated into the data structure used for Bitcoin transactions, it would amount to around 2,900 transactions per second and 47 TB of storage space. This already far exceeds the hard drive space on an ordinary computer.

The third problem is lack of privacy protection. All the transactions stored in a public blockchain are replicated in nodes from all over the world. At the moment, privacy protection in blockchains consists mainly of using a mechanism similar to money-laundering to conceal information about cryptocurrency transactions. There are two main methods: (1) Cryptography accumulators [2], used by Zerocoin, and (2) CoinJoin [3], used by SharedCoins, Dark Wallet, CoinShuffle, the PrivateSend feature of Dash, and JoinMarket. Cryptocurrency transaction information recorded on the blockchain gives no indication of the sender.

The two methods above can only be used for cryptocurrency transactions and so cannot be used for other general transactions or contracts. The popularity of Ethereum's smart contract is due to its ability to handle general transactions or contracts, not just cryptocurrency transactions. These include asset transactions and patent licensing as well as contract, document, and information records. Smart contracts for non-cryptocurrency transactions cannot make use of cryptocurrency's privacy protection technology, thus limiting the system's scope of application.

The fourth problem is that the blockchain currently has limited application scenarios. The concept of decentralization has gained acceptance in some circles: the forging and trading of cryptocurrencies, for example, can now be completely

[2] A 51% attack is where control over more than 51% of the nodes gives the controllers the ability to modify blocks or control their production.

implemented using a decentralized model. However, human economic activities are influenced by law, habit, legacy systems, and interpersonal relationships. Dispensing completely with centralized operations is impossible. Industries with related applications, such as: logistics, financial systems, medical records, the collection and verification of data in the Internet of Things (IoT), supply chain management, share or rights transactions, social networking software, electronic patient records, micropayment/mobile payment systems, asset transactions, and distribution of digital products—nearly all of these applications still require a centralized agent or intermediary. If the public blockchain cannot be integrated with industries that have similar centralized operations, the use of the blockchain as a trust machine will be greatly limited.

The following example uses digital product distribution to explain why that is the case. For digital products such as e-books, music, movie rentals, and electronic tickets, the widespread use of the Internet and larger bandwidth has popularized sales over online platforms. To expand their sales channels, the rights-holder will usually commission agents to make sales over the agent's network platform. The agent collects payments from users and maintains a record of accounts. The accounts are then provided to the rights-holder at fixed intervals with details on downloads and corresponding royalties. However, since the accounts are recorded and maintained by the agent, the rights-holder is unable to verify their authenticity. For example, the agent's records may contain accidental omissions or other errors due to bugs in the system. Or the agent may deliberately forge or modify the records to reduce the amount of royalties payable to the rights-holder.

Based on the explanations above, we can now give a summary of the current problems to be solved:

Problem P1: Blockchain transaction speeds are too slow. How can the blockchain handle a large number of transactions in a short period of time?

Problem P2: How can blockchain bloat be avoided when storing large numbers of transaction records on the blockchain?

Problem P3: How can the privacy of involved parties be protected when records for transactions other than cryptocurrencies are written to the blockchain?

Problem P4: In the real world, the division of labor in the commercial environment means that intermediaries (or agents) cannot be easily replaced. They may also form a wall between digital asset publishers and consumers. So how can we retain a system with many intermediaries while providing transparent, reliable, and verifiable consumption records?

These four problems are tightly interrelated. For example, if Problem P1 is solved and blockchain's transaction bandwidth is greatly expanded, then Problem P2 seems inevitable. Large numbers of transaction records on the blockchain will lead to excessive data storage. If Problem P3 is solved, the privacy of the transaction parties will be protected and third parties will not be able to obtain transaction details, but then Problem P4, that of dishonest agents, may become even worse.

In this paper, we propose a multi-chain architecture to completely solve all of these problems. Transactions are first processed outside the main blockchain. A hash value which can be considered as a condensed fingerprint of these processed transactions is first recorded in a contact[3] of the main blockchain after the end of these transactions. Then, we apply a scheme called “distributed auditing” in which all participants examine their own transactions according to the uploaded fingerprint in the contact. In case there is anything wrong, such as incorrect or missing transactions, a participant can generate a small-sized cryptographic proof which can be sent to the contact to issue an objection (or even prove fraud). The pre-arranged function in the contact accepts the cryptographic proof and can then perform specific actions, such as paying bonds to the participant.

This paper is organized as follows. Section 2 shows the multi-chain architecture of InfiniteChain, including how we perform fraud proof in sidechains in index Merkle trees and distributed computing. In Sect. 3 we discuss how problems P1, P2, P3, and P4 are solved according to the proposed InfiniteChain technology. The implementation details and experimental results are presented in Sect. 4 and conclusions are drawn in Sect. 5.

2 A Multi-chain Architecture

The architecture of the proposed multi-chain blockchain is shown in Fig. 2. A multi-chain is a joint operating model consisting of the main blockchain and several sidechains. Generally speaking, transactions that do not need to be processed quickly, such as cryptocurrency transactions or individual contract records, are first sent directly into the P2P network and then linked to the main chain by nodes that have become block producers.

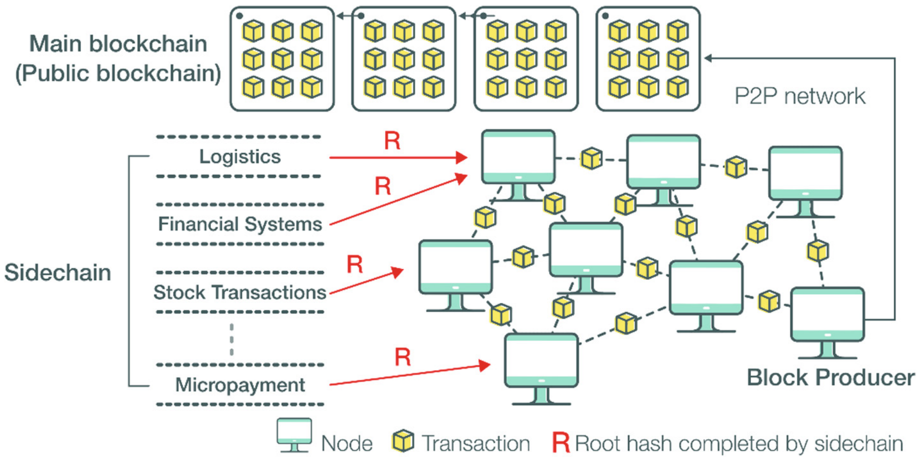


Fig. 2. The multi-chain architecture

[3] In the follows, we use contract to represent an entity, similar to Ethereum’s smart contracts.

High-volume transactions, or those that require centralized matchmaking, however, are first processed on a sidechain. A hash value for the transactions is then generated and sent to a node in the P2P network and linked to the main chain. The sidechain runs at a high speed and accumulates a large number of transactions after a certain amount of time. The auditing node responsible for the decentralized operation of the sidechain then generates a hash value and corresponding identification code, which is then sent to the main chain.

The maintenance and operation of the main chain is the same as the operation and management of ordinary public blockchains. Sidechain operations are initiated by industry applications (e.g., transaction agent platforms, professional brokers, investment banks, securities companies, auditors, appraisers, lawyers, toolkit developers). For market and business development, different sidechains are managed and operated for individual business types. Sidechains must regularly synchronize their information with the main chain to avoid counterfeiting or tampering of data.

2.1 Existing Sidechain Technologies

There are several techniques for running transactions outside of the main chain before adding them to the main chain. After explaining each of these techniques, we will look at how InfiniteChain's sidechains are different. The first technique uses "relay-based" sidechains. Assets are transferred between the main chain and sidechain before the transaction is conducted on the sidechain [4, 5]. The assets are transferred back to the main chain after a certain time. This reduces the number of transactions that take place on the main blockchain. Implementations of this system include BTC-Relay [6] and Rootstock [7]. The problem of the relay-based technique is implementing a 2-way peg protocol. For example, the blockchain of Bitcoin is impossible to establish such a relayer.

Another type is "channel-based" and usually referred to as "off-chain." Lightning-network [8] and Raiden [9], for example, use off-chain transactions to increase their TPS. In this method, there is no need to use nodes to obtain consensus about transactions produced in sidechains. A payment channel is created in advance on the main chain and all the participants in the transactions made over this channel exchange electronically signed information outside of the main chain to indicate that transactions have taken place. A summary of these transactions are then written back to the main chain. Both of the above methods only offer a solution for high-frequency transfer of cryptocurrency or tokens and cannot be applied directly to contract records and other non-cryptocurrency transactions. In addition, this approach requires a prepayment in the channel and the Internet connection in real time to avoid receiving transactions from others, conditions that can be difficult to apply.

InfiniteChain uses an agent type (Proxy-Based) system for its sidechains. In this type of scenario, the user will commission a platform or agent to assist them in chaining the transaction and submitting non-defective modifications into the consensus system. Thus, the effectiveness of the application is achieved by the underlying data structure. An unlimited number of lower sidechains can be generated at any time when needed, making InfiniteChain suitable for solving the problems of real-world scenarios interfacing with the blockchain. In InfiniteChain we employ *index Merkle trees* and invent a scheme called *distributed auditing* to have the main chain's block producers perform fraud proofs for sidechain transactions.

2.2 Sidechain Fraud Proof Performed by Nodes of Main Chain

In the multi-chain operation of InfiniteChain, the main chain is a public blockchain and so can obtain global consensus. It effectively validates the operating of agents which manifest the transactions in sidechains. Our idea is for block producers to perform fraud proof of transactions in sidechains. The multi-chain architecture of InfiniteChain can therefore solve all of the problems outlined in Sect. 1.

For fraud proof of transactions in sidechains, we propose a process of distributed auditing which is based on our index Merkle tree data structure.

Index Merkle Tree

The transactions conducted in a sidechain of InfiniteChain are stored in an index Merkle tree, which is not only a binary tree but also a hash tree. A Merkle tree is a tree of hashes in which the leaves are hash values and the top of the tree is occupied by a root hash. A root hash, which represents a fingerprint of transactions stored in the corresponding Merkle tree, and a slice of Merkle tree are sufficient to prove the existence of a transaction in a Merkle tree. The hash value of a transaction is stored in one of the leaf nodes in the Merkle tree. Further up the tree, an internal node stores a hash value that is the hash of the concatenation of all the hash values of its child nodes.

Figure 3 shows a Merkle tree with a height of four and eight leaf nodes and Fig. 4 shows a slice of a Merkle tree that is nine nodes high. If we would like to prove a transaction does not exist in a Merkle tree, we have to scan all the transactions in a Merkle tree. A Merkle tree is therefore not sufficient in the fraud proof of a sidechain, because block producers can only execute functions in contract with small-sized parameters[4]. We will employ the index Merkle tree proposed in [10][5] as a data structure to store transactions in a sidechain.

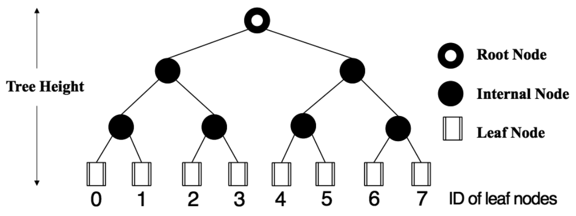


Fig. 3. A binary Merkle tree with a height of four

The hash value of a transaction is stored in one of the leaf nodes of the index Merkle tree according to the index function Γ. If a Merkle tree is a full binary tree, it

[4] Refer to Sect. 4 for details.

[5] The index Merkle tree is called an FBHTree in [10].

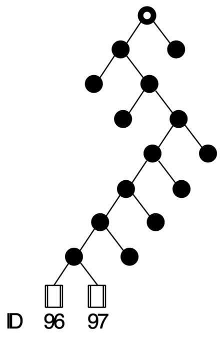


Fig. 4. A slice of an FBHTree that is nine nodes high

has 2^{N-1} leaf nodes if its height is *N*. The index function Γ returns the ID of a leaf node. In this paper, we propose the index function Γ to be:

Γ(Transaction ID) = SHA–256[6] (Transaction ID) mod 2^{N-1}

That is, Γ returns 0 to 2^{N-1} if the height of the index Merkle tree is N. Hash values of different transactions may be stored in the same leaf node due to a collision of the Γ function. In addition to proof of existence of a transaction in an index Merkle tree, proof of inexistence can also be easily applied with a small-sized cryptographic proof. Given a transaction with an ID of *t* and root hash *R* of an index Merkle tree, if $\Gamma(t) = x$, then we obtain the slice of the node whose ID is *x* and check if *t* exists according to this slice, because the protocol of operating an index Merkle tree has each transaction stored in a fixed slice. For the details of the index Merkle tree, refer to [10]. With index Merkle trees, we invent a form of distributed auditing that can do fraud proof for transactions in sidechains.

Distributed Auditing

The operation of a single sidechain is divided into multiple stages. Each stage contains the execution and auditing of some transactions. There is a corresponding contact of the sidechain in the main chain that keeps necessary information and functions for the sidechain's operation. An index Merkle tree will be generated in each stage. That agent is responsible for keeping and maintaining it. Its root hash will be put into the contact. Refer to Fig. 5 for steps involved in a stage.

Step (1): The agent responsible for initiating a stage starts by conducting a series of transactions with participants (or consumers).

Step (2): After a certain period of time, the agent sends processed transactions $\sum$ from Step (1) to the audit node.

Step (3): The audit node uses transactions $\sum$ to generate an indexed Merkle tree (IMT). The IMT is also used to generate a root hash value *R*. *R* and the corresponding identification tag, such as its stage number, are sent to its

[6] The SHA (Secure Hash Algorithm) is one of a number of cryptographic hash functions. SHA-256 algorithm generates an almost-unique, fixed size 256-bit (32-byte) hash.

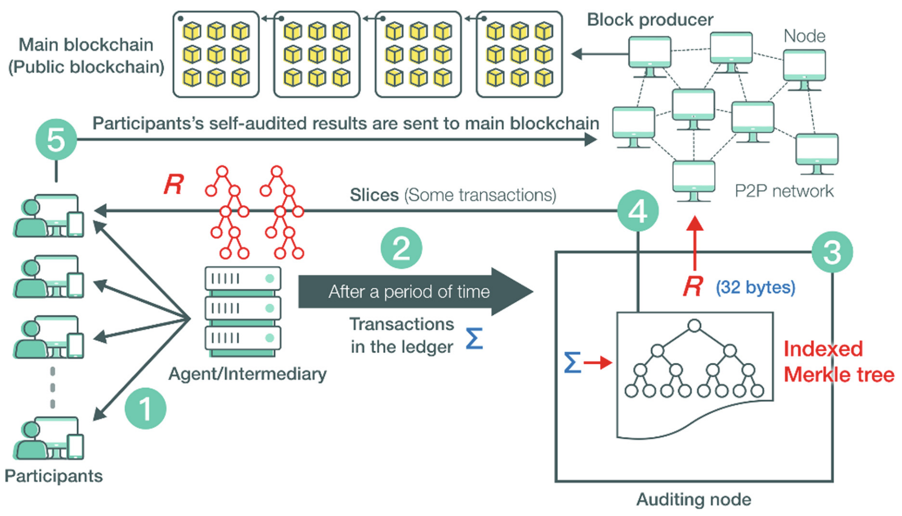


Fig. 5. A sidechain with distributed auditing

contact in the main chain for storing. All participants can use the identification tag from the contact in the main chain to obtain *R*.

Step (4): Participants are responsible for auditing their own transactions to see if they were correctly placed in the IMT:

- The given root hash value *R* is used to ask the auditing node to return the corresponding slices of the participant's own transactions, with each slice representing one such transaction. Since *R* is anchored to the main chain, an audit of the slice that does not turn up a particular transaction is cryptographic proof that the agent did not put their own transaction into the IMT.

Step (5): If the participant's audit finds that the agent provided missing or incorrect data, the associated information is signed and then sent to a node for arbitration by block producers by calling an arbitration function in the contact. If arbitration finds that the agent made an error, then the participant receives a share of the bond which was previously stored in the contact.

Step (6): The agent pays royalties to the rights-holder. A rights-holder can use *R* and the IMT to verify that a royalty payment[7] is free from error.

The integrity of transactions generated by sidechain operations is maintained by all participants. Participants and the agent both electronically sign their transaction information to realize mutual non-repudiation. In Step (4), multiple participants are involved in auditing the existence and integrity of transactions on the sidechain. Any

[7] The IMT can store a lot of different activities, such as electronic voting.

omissions or errors found in an agent's transactions are arbitrated by nodes on the main chain by calling a function in the contact. If arbitration is passed, then the bond is automatically shared among the participants who issued the arbitration; if not, it is refunded to the agent[8]. This boosts the incentive for participants to take part in an audit.

3 Why Are Problems P1, P2, P3, and P4 Solved?

An InfiniteChain will contain many sidechains. Transactions are conducted in different sidechains. A single transaction agent is usually responsible for all transactions from a matchmaking service. Transactions processed by a sidechain agent do not need to be immediately placed on the blockchain. The hash value for a ledger containing N transactions is eventually placed on the blockchain: the equivalent of placing N transactions on the blockchain all at once. In practice, placing the hash value of a transaction ledger on the blockchain only takes up the bandwidth of a single transaction. In this manner, the transaction bandwidth of the blockchain can therefore be expanded almost indefinitely. The number of transactions per second is no longer constrained by the limitations of the main chain. Not having a limit of transactions per second means that fast speeds can be attained.

Since public blockchains can currently handle dozens of transactions per second, the speed of an InfiniteChain blockchain can now be easily increased to tens of millions of transactions per second, as shown in the following formula:

$$\begin{aligned}\text{Transactions per second} = {} & (\text{Average number of blocks generated per second}) \\ & \times (\text{Average number of transactions per block}) \\ & \times (\text{Average number of transactions per InfiniteChain ledger})\end{aligned}$$

If we set the main chain's transactions per second to 10, then the entire blockchain's transactions per second =10 × 1,000,000 = 10,000,000 TPS. We can therefore achieve 10 million TPS with ease. This technique elevates the practicality of the blockchain to a whole new level. Problem P1 has now been solved.

InfiniteChain's approach is to let the agent conduct transactions. We refer to this as a sidechain operation. After a certain amount of time, the root hash of this sidechain is placed on the main chain. Since the transaction ledger formed by the sidechain does not need to be stored on the main chain, Problem P2, namely information bloat, does not exist on the main chain, nor does InfiniteChain's fix for Problem P1 create Problem P2. The details of all sidechain transactions are in the safekeeping of the responsible agent. Since its root hash and identification tag have already been placed on the main chain, they cannot be altered by the agent.

During sidechain operations, all transactions are stored securely in index Merkle trees. Transaction details are also encrypted with the public keys of the participant and digital asset provider. Only the participant and digital asset provider can use their

[8] Alternately, the mistake of agent or error can be recorded in the contact.

private keys to validate their own transactions. The privacies of the participant and digital asset provider are therefore protected. Problem P3 has now been solved.

Since only hash values of index Merkle trees are published, participants and digital asset providers can use the index function to immediately pinpoint the leaf node for a particular transaction in the index Merkle tree. When a participant wishes to audit their own transaction and see if it was correctly stored in the transaction ledger, the participant can submit a transaction audit request to the agent. As the participant already has the transaction serial number (the completed transaction is electronically signed by the agent and so cannot be repudiated by the agent), the agent must then present the slice for the transaction. The participants can then use the root hash of the ledger and the slice for the transaction to verify the integrity of the transaction and whether it exists in the transaction ledger. Problem P4 has now been solved.

4 Experimental Results

We have performed a series of experiments to demonstrate the feasibility of the proposed architecture. We installed a cloud server as our auditing node: a virtual machine of AWS EC2 r4.xlarge with 4 CPUs. The operating system was Ubuntu 16.04. Node.js 8.6.0 was employed to implement the auditing node. We divided the experiments into three major parts, including measuring the time required to generate an index Merkle tree, the time required to extract slices from an index Merkle tree, and the required gas consumed in the function execution of smart contract in the Ethereum blockchain.

The time and its standard deviation required to generate an index Merkle tree is shown in Table 1. For an index Merkle tree which contains less than 10,000 transactions, we only require about 4.6 s for generation.

Table 1. The time required to generate an index Merkle tree

# of transaction	Average time (ms)	Standard deviation (ms)
100	26.22	10.79
1,000	282.79	13.20
10,000	4685.59	175.94
100,000	62380.20	1015.32
1,000,000	1052168.24	44106.64

Table 2 lists the time required to extract slices from index Merkle trees, which is the operation of Step (4) in Fig. 5. We store different numbers of transactions with different heights. We only needed less than one millisecond to extract a slice even when an index Merkle tree contained one million transactions.

In the third part of our experiment, we implemented the InfiniteChain in the public blockchain, Ethereum. Ethereum provides a decentralized Turing-complete virtual machine, the Ethereum Virtual Machine (EVM), which can execute scripts using an international network of public nodes. Gas is the internal pricing for running a transaction or contract in Ethereum. Using more computation and storage in Ethereum means that more gas is used (See the yellow paper for a breakdown of operations and respective gas costs [11]). One fundamental reason for metering is that it provides an incentive for miners to operate the contract code [12].

Table 2. The time required to extract slices from index Merkle trees

# of transaction	Height of the index Merkle tree	Average time (ms)	Standard deviation (ms)
100	7	0.47	1.32
1,000	10	0.18	0.08
10,000	14	0.30	0.27
100,000	17	0.42	1.24
1,000,000	20	0.54	1.33

Since computation with global consensus is expensive, transactions have a gas limit field to specify the maximum amount of gas which the sender is willing to buy. If the gas used exceeds this limit during execution, processing is stopped, which protects full nodes from attackers who could make them execute infinity loops without a gas limit. If a transaction would take longer than a block to process, then it would not be included in the block.

A block also has a field called gas limit. It defines the maximum amount of gas all transactions in the whole block combined are allowed to consume. Its purpose is to keep block propagation and processing time low, thereby allowing for a sufficiently decentralized network [13]. Consensus protocol allows the miner of a block to adjust the block gas limit by a factor of 1/1024 (0.0976%) in either direction [14, 15]. Currently, the gas limit is around eight million. Refer to Table 3. In our experiment, the highest gas is 3.7 million, in which the height of the index Merkle tree is 20, for storing one million transactions, with 9 transactions in the leaf node. According to our previous experiments about index Merkle tree, the maximum number of transactions in a leaf node is 8 [10]. The size of cryptographic proof and required gas are therefore small enough to be performed as a function in Ethereum smart contract.

Table 3. Gas consumption in Ethereum blockchain platform. α: Height of the index Merkle tree, β: Number of transactions in the leaf nodes

α	β				
	1	3	5	7	9
3	315422	511245	810761	1216315	1730911
5	548841	744664	1044305	1450118	1964195
10	1138873	1334255	1633713	2039442	2553564
15	1736287	1931590	2230814	2636102	3150952
20	2340295	2526356	2835031	3240553	3755455

5 Conclusion

Decentralized application systems based on blockchain technologies are just now beginning to enter our everyday lives. However, experts have pointed out some bottlenecks in the basic technology. As noted in Sect. 1, these are Problem P1 (insufficient bandwidth), Problem P2 (blockchain bloat), Problem P3 (difficulty protecting privacy), and Problem P4 (difficulty integrating with centralized scenarios). If any one of these problems remains unsolved, then the dream of blockchains becoming trust machines will be just that, a dream. The blockchain will be limited to a platform for mining and trading cryptocurrency, or will only be used in a limited number of application scenarios.

In this paper we propose a multi-chain architecture which employs distributed auditing to perform fraud proof in the sidechain. An efficient and effective fraud proof solves problems P1, P2, P3, and P4. Experimental results shown in Sect. 4 demonstrate that the index Merkle tree is able to an appropriate data structure for supporting fraud proof. The cryptographic proof generated from an index Merkle tree is small enough to be validated in a contract of public blockchains.

In future work, we are extending distributed auditing to implement a real-time and low-cost cash flow sidechain. A contract published to the main chain is used to control the cash flow exchange of participants in the sidechain. General cash flow does not need to pass through the main chain, which speeds up the main chain. An agent is responsible for accepting and processing the remittance, deposit, and withdrawal requests from participants. The index Merkle tree is still the basis for the agent's generation of cryptographic proofs for fraud proofs.

References

1. Banjo, Y.: Ethereum won't scale like you've been told. https://medium.com/@yobanjo/ethereum-wont-scale-like-you-ve-been-told-cae445bef539. Accessed 10 Apr 2018
2. Benaloh, J., de Mare, M.: One-way accumulators: a decentralized alternative to digital signatures. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 274–285. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-48285-7_24
3. Maxwell, G.: CoinJoin: bitcoin privacy for the real world. http://bitcointalk.org. Accessed 10 Apr 2018
4. A simple explanation of bitcoin sidechains. https://gendal.me/2014/10/26/a-simple-explanation-of-bitcoin-sidechains/. Accessed 10 Apr 2018
5. How Two New Sidechains Proposals Could Change Bitcoin's DNA. https://www.coindesk.com/two-new-sidechains-proposals-change-bitcoins-dna/. Accessed 10 Apr 2018
6. BTC-Relay, A bridge between the Bitcoin blockchain & Ethereum smart contacts. http://btcrelay.org/. Accessed 10 Apr 2018
7. Rootstock Whitepaper. http://www.the-blockchain.com/docs/Rootstock-WhitePaper-Overview.pdf. Accessed 10 Apr 2018
8. Lightning Network: Scalable, Instant Bitcoin/Blockchain Transactions. https://lightning.network. Accessed 10 Apr 2018
9. Raiden Network - Fast, cheap, scalable token transfers for Ethereum. https://raiden.network/. Accessed 10 Apr 2018

10. Hwang, G.-H., Chen, H.-F.: Efficient real-time auditing and proof of violation for cloud storage systems. In: The 9th IEEE International Conference on Cloud Computing (IEEE Cloud 2016), 27 June–2 July 2016, San Francisco, USA (2016)
11. http://gavwood.com/Paper.pdf. Accessed 10 Apr 2018
12. https://media.consensys.net/ethereum-gas-fuel-and-fees-3333e17fe1dc. Accessed 10 Apr 2018
13. https://bitcoin.stackexchange.com/questions/39132/what-is-gas-limit-in-ethereum. Accessed 10 Apr 2018
14. https://www.reddit.com/r/ethereum/comments/6g6tww/there_are_hundreds_or_even_thousands_of_pending/dinzrgq/. Accessed 10 Apr 2018
15. https://www.etherchain.org/tools/gasLimitVoting. Accessed 26 Jan 2018

Research Track: Smart Contracts

A Method to Predict the Performance and Storage of Executing Contract for Ethereum Consortium-Blockchain

Huijuan Zhang(✉), Chengxin Jin(✉), and Hejie Cui(✉)

Tongji University, 1239 Siping Rd, Shanghai, People's Republic of China
mszhj@tongji.edu.cn, jinchengxin@outlook.com, cuihejie331771@gmail.com

Abstract. As the fundamental technology of Bitcoin, blockchain enables people to deal with trust problems in network. Ethereum, as a well-known public blockchain, is favored by large companies and organizations for its excellent account model and Turing-complete smart contracts, and is widely used to develop consortium-blockchain. However, the performance and storage of executing contract gradually degrade as the transaction volume increases. Meanwhile, compared with public blockchains, companies need a more accurate estimation of prospective performance and storage based on business scale for decisions making or early warnings. In this paper, a prediction model derived from the core structure of Ethereum's "World State" is proposed. The proposed model predicts the performance and storage of executing contract based on transaction volume. The comparison between the experimental and predicted data reveals that this model can perform a relative accurate prediction of the prospective system's performance and storage.

Keywords: Blockchain · Ethereum · Contract · Performance · Storage

1 Introduction

Ethereum [3], as the successor of Bitcoin [5], establishes a Turing-complete smart contract on blockchain to realize distributed application DApps [6]. Meanwhile, the account-based design of Ethereum provides convenience for the docking of existing business models (compared with UXTO model). Thus many companies choose Ethereum to build their consortium-blockchain system or develop on it (for example EEA [1]) based on two points. As a result, Compared with the public Ethereum blockchain, Ethereum consortium-blockchain mainly uses the transaction to execute the contract instead of making a transfer of ETH cryptocurrency. Therefore, this paper focuses on the performance and storage of executing the Ethereum contract.

However, the test results show that when the transaction volume reaches a certain scale, the execution performance of Ethereum will significantly reduce and large storage space will be occupied. (e.g. when the limit of block generation rate was modified to one block per second, the TPS of a contract, which is about 200 at start, would reduce to 100 as the transaction volume reaches one million). For companies, predicting the prospective performance and storage are important indicators in making technical decisions, and also allow companies to prepare for hardware, monitoring and plans in advance. Thus, it is necessary to predict the prospective performance and storage of the system in Ethereum consortium-blockchain based on the business scale.

In public Ethereum blockchain, it's impossible to predict the distribution and complexity of smart contract since anyone can deploy a contract easily. As a result, it's hard to accurately estimate the perspective performance and storage. Nevertheless, it is possible to predict the performance and storage in Ethereum consortium-blockchain resulting from that the participants are the authorized nodes; that the relatively-fixed smart contract with evaluable complexity is determined by business model; and that the transaction volume is determined by business scale.

The prediction method proposed in this paper speculates the performance and storage by analyzing the relationship between transaction volume and "World State" [7]. "World State" is the core part of Ethereum. The account system maps the state data as key/value form and stores them in LevelDB through this special structure [13]. "World State" is implemented using "the modified Merkle Patricia tree (trie)" [4] (hereinafter referred to as MPT). PATRICIA trie (Patricia tree) is a space-optimized version of the traditional trie data structure, in which every node with only one child is merged with its child. This data structure was firstly proposed by Morrison [16] in 1968, and then well analyzed in "The art of computer programming" by Knuth [17] in 1973. The "Merkle" part of the radix trie arises in the fact that a deterministic cryptographic hash of a node is used as the pointer to the node, leading to the fact that the Ethereum could trace the history state through root of "World State" in any block header. The contract in Ethereum is called transaction. Depending on the implementation of Ethereum [2], the time consumed by a transaction call for contract is mainly determined by the execution time of Ethereum Virtual Machine (EVM) [7] and by the modification of the "World State". Meanwhile, the increment of data generated by this transaction depends on the transaction scale and the increment amount of the "World State". With the transaction volume growing, the "World State" becomes larger, resulting in the rise of time consumption and data space. Consequently, the estimation of performance and storage can be obtained under the premise of figuring out the relationship between transactions volume and "World State".

This paper is focused on the relationship between the performance and storage increment of "World State" after the transaction volume reaches a certain scale, and a prediction formula is proposed for this relationship. Using this formula, companies could predict the prospective time consumption of executing a transaction and the storage occupancy based on the transaction volume. At the end of the paper, a suggestion is raised for the design of contract in Ethereum consortium-blockchain.

The rest of this paper is organized as follows. Section 2 reviews relatived development on blockchain. Section 3 gives the key technologies to introduce how to predict the performance and storage. Section 4 provide the experiments to prove the predict methods. Section 5 concludes this paper.

2 Relatived Work

Since blockchain is a distributed ledger maintained by all participants, many researches have been focused on PBFT [11] and other consensus algorithms to improve the efficiency. In this case, the performance of executing contract would become the bottleneck restricting the system, but this research area is rarely explored at present. For the contract of Ethereum, most researches are focused on the security [9] of smart contract or improving its smartness [10], while the effect of execution efficiency is scarcely investigated. High-Performance Computing is discussed for Ethereum Tokens [18], but not the performance bottleneck of Ethereum itself. The assessment [8] results show that the performance will decline and delay be high when the transaction volume grows (10,000 transactions), however the test volume in that paper is small and there are few analysis details. In terms of storage, EtherQL [12] provides highly-efficient query primitives for analyzing blockchain data, but not for predicting the size of data space. On the other hand, the properties of PATRICIA trie are very important due to the highly-correlated relationship between the performance/storage and "World State", as well as the close tie of "World State" MPT implementation with PATRICIA trie. Some existing papers point out that the height of the PATRICIA trie behaves quite differently across regions: it exhibits an exponential of a Gaussian distribution, which satisfies $log(n)$ [15]. There is another paper analyzing the relationship between the average height and random inserting in PATRICIA trie. To sum up, there is not a single paper concentrating on MPT and the relationship between performance/storage and MPT.

3 Key Technologies

3.1 Influencing Factors

In Ethereum, the time consumption of executing a transaction could be divided into two parts: the EVM execution time and the cost of modifying "World State", while the storage increment is composed of transaction volume and state data of changed "World State". Since the contracts in consortium-blockchain are usually fixed and predictable, the time consumption of EVM and the transaction volume tend to be stable. Therefore, the root cause for the change in performance and storage is the maintenance cost of "World State".

After every transaction is executed through a smart contract, it eventually modifies the tree node of "World State". The state of Ethereum is the result of mapping 160-bit address and account state in the tree, which is called `State Trie`. An account state corresponds to a leaf node while its address is depicted as the path from the root to the node. In Ethereum, the smart contract is also an account. Each contract account has its data storage, called `Storage Trie`. Data storage is also implemented using the MPT tree. Hence, the "World State" tree of Ethereum is actually composed of two parts: an upper account tree and a lower storage one. There are two ways to create an Ethereum contract: one is creating from a transaction, the other is from an old contract. The new contract will be stored in State Trie of "World State" and contract data in Storage Trie whose root points to the contract. For a specific business logic, depending on how the contract is implemented, the data entering the "World State" would be mainly distributed in State Tire (constantly create new contract to store more data) or Storage Trie (create a single contract to store a large amount of data) or between (a balanced distribution of State Tire and Storage Trie).

The nodes of an MPT tree are divided into `leaf`, `expansion`, and `branch` (`NULL` is not within the scope of our discussion). Leaf nodes and expansion nodes are similar in size, while branch nodes are much larger. The "World State" would be modified for multiple times according to the design of contract after the transaction calls a contract. Due to the features of tracing history, every modification would generate a new path from the new leaf to the root. In the implementation of Ethereum, when searching a node in the path or inserting a new node, the system would execute function $sha3()$ and read/write LevelDB which brings more time consumption, and the size of new node corresponds to the data increment in this modification. Therefore, if the height of current new leaf can be predicted, it is possible to infer the performance and data increment. The random hash of Ethereum address results in the random modification for MPT, thus the average height of the leaf nodes would increase as the times of modification increase. When a state change occurs for MPT, it is possible to speculate the execution time and storage space increment if the current tree height of new leaf nodes could be estimated.

As a result, predicting the performance and storage occupancy of a transaction can be achieved by analyzing the times of modification for "World State" or testing the business contract, and calculating the time consumption and storage based on the current height of "World State" in MPT.

3.2 Height of MPT

MPT is evolved from PATRICIA trie. In the implementation of MPT, a branch node could hold 16 branches. When the branches reach a certain scale, most nodes in MPT would become branch nodes, while the extension nodes are compressed prefix in PATRICIA trie. Therefore, MPT could be seen as a 16-ray PATRICIA trie.

Considering of the huge space of 160-bit address, the random selection of address could be approximated as an asymptotic distribution. Besides, the address would be executed by $sha3()$ to become the key of MPT, so the probability of choosing which alphabet be inserted into MPT is equal. As the result, the height of MPT can be regarded as the expected height of a 16-ray Patricia tree with the universal asymptotic distribution insertion of equal-probability alphabet.

According to Devroye [14], the expectated height of universal asymptotic PATRICIA trie is given by

$$\mathbf{E}\{H_n\} \sim c\ log\ n$$
$$where\ c = 2/log_2(1/\textstyle\sum_j\ p_j^2)$$

and the highest node is expressed by

$$\frac{H_n - log_2 n}{\sqrt{2log_2 n}} \rightarrow 1$$

Hence, the expectation and the max height of 16-ray MPT ($p_j = 1/16, 1 \leqslant j \leqslant 16$) can be obtained as:

$$\mathbf{E}\{H_n\} \sim\ log(n)/2$$
$$H_n = \tfrac{2\sqrt{2log_2 n}+log_2 n}{4}$$

3.3 The Prediction of Transaction Performance and Storage

A simple instance of Ethereum contract is used to describe the relationship between the transaction and performance/storage, as shown in Fig. 1.

```
pragma solidity 0.4.16;
constract Order {
  uint256 private order_no;
  func Order(uint256 _no) { _no = order_no; }
  func doOrder() { ... }
}

constract Business {
  uint256 private version;
  func createOrder(uint256 _no) returns(address) {
    Order o = new Order(_no);
    o.doOrder();
    return o;
  }
}
```

Fig. 1. Simple business model contract in Ethereum consortium-blockchain.

When a transaction calls the function `createOrder` in contract business, the Ethereum actually loads the contract business code from State Trie into EVM, and calls the function `createOrder` to create a new contract order (account), and then the function in new contract will be invoked. At the committing phase of the entire process, the data in cache would be inserted to "World State" to form a new world state and generate a series of new nodes. The tree root of the current world state is written to the new block so that the state can be extracted from this block. Like `block 3` in Fig. 2, the path from the leaf contract "`Order`" to the root is the new inserted node in `block 3` compared with `block 2`. These nodes are the potential influential factors that affect efficiency and storage. The contract "`Order`" and "`Business`" are stored in State Trie, and the attributes "`order_no`", and "`version`" are stored in the instance of contract "`Order`" and "`Business`". It's obvious that the State Trie and Storage Trie would insert data continually as long as the transaction calls function `createOrder`.

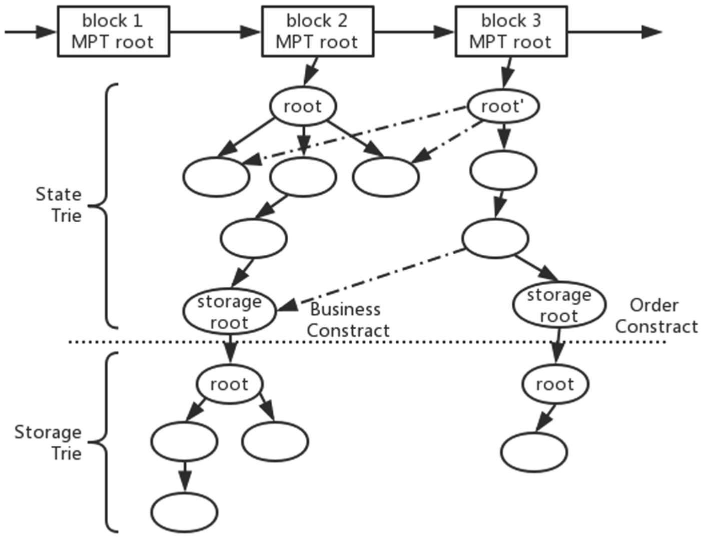


Fig. 2. "World State" in Ethereum.

For a specific business model, the State Trie and Storage Trie would be assigned different proportions of data depending on the design of contract in the consortium-blockchain. In general case, three types of models can be designed, as shown in Fig. 3:

1. The data are distributed on State Trie by the method of creating new contracts from old contract.
2. For the minority contracts with complex storage structure, the data are mainly stored in the Storage Trie of those contracts.
3. New contracts are created and appropriate contracts data storage is designed, and the data are allocated to State Trie and Storage Trie as needed.

Thus, after the prediction of the height of MPT, different designs of contracts would lead to different prediction methods, but the design ideas are the same.

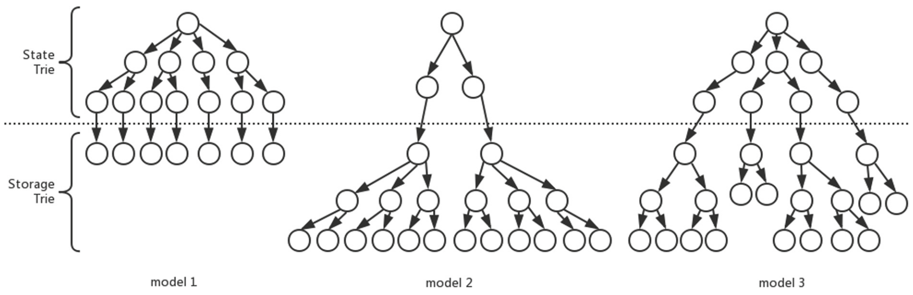


Fig. 3. Three type of business model.

It is supposed that the contract is designed as the first type, which mainly distributes the data on the State Trie. Considering a general business scene, a transaction would deploy a times contract (e.g. the `Order` contract in Fig. 1) for a business request, and the time consumption of executing or generating every node is t (including the calculation of $sha3()$ and database access time). For the situation depicted above, when the transaction volume reaches n and a new transaction is executed, the average time consumption of modifying MPT is given by

$$\mathbf{T}_{mpt}(n) = \frac{t}{2} log(an)$$

The max time consumption is represented by

$$\mathbf{T}_{mpt\_max}(n) = \frac{t}{4}(2\sqrt{2log_2(an)} + log_2(an))$$

Ethereum uses LevelDB as the database to store key/value. The key to accessing database is irregular on account of the discreteness of hash. The LevelDB has an excellent performance in reading/writing continuously, while bad for random key [13]. Therefore, the time t for accessing LevelDB would be longer as the amount of data storage increases. In fact, the test results show that if n is large enough, the value of t will increase and the efficiency will degrade largely for some data which not hit LevelDB cache at times.

Then, the T_{exec} is added to execute the contract in virtual machine. After n transactions are executed, the average execution time and the maximum time of a transaction are given by

$$\mathbf{T}_{avg}(n) = \mathbf{T}_{exec} + \mathbf{T}_{mpt}(n)$$
$$\mathbf{T}_{max}(n) = \mathbf{T}_{exec} + \mathbf{T}_{mpt\_max}(n)$$

Because of the discreteness of hash, the common prefixes between addresses are hardly identical. So, most nodes of MPT are on the branch. When n is large, it can be assumed that the branches are fully filled. At the same time, since the length of address is fixed, only the leaf nodes can store state, rather than the branch or extension nodes. It is assumed that the filled branch size is s_b, the size of the leaf node is s_l, the storage space occupied by the account state is s_a, and the sum of storage trees for each account is represented by s'. After performing n transactions, the storage increment resulting from modifying the MPT tree is calculated by

$$\mathbf{S}_{mpt}(n) = s_b \left(\frac{log(an)}{2} - 1 \right) + s_l + s_a + s^{'}$$

the maximum storage increment is given by

$$\mathbf{S}_{max}(n) = s_b \left(\frac{2\sqrt{2log_2(an)} + log_2(an)}{4} - 1 \right) + s_l + s_a + s^{'}$$

The size of the transaction itself is S_t. The total storage and maximum space occupation after n times of transaction executions are expressed as

$$\mathbf{S}_{sum\_avg}(n) = nS_t + \textstyle\sum_{i=0}^{n} S_{mpt}(n)$$
$$\mathbf{S}_{sum\_max}(n) = nS_t + \textstyle\sum_{i=0}^{n} S_{max}(n)$$

The same analysis also applies to designing the contracts of type 2 and 3.

4 Experiment

4.1 Experimental Method

Experiments are conducted based on the formula in Sect. 3.3 to verify the correctness of the proposed prediction method. The recorded experimental data are compared with the predicted data. Three dimensions of data are collected in the experiment, i.e. the tree height of the State Trie in the "World State", time consumption of execution, and the storage occupancy after transaction.

The testing conditions of the experiment are as follows: 1. Build a single Ethereum node, without networking. 2. Modify the logic of generating block so that one block will only package one transaction (e.g. each transaction would have one submit, to avoid the impact of the Ethereum cache on the experimental results). 3. Using the simple contract presented in Fig. 1, only one new `Order` account contract will be created each time when the `createOrder` method is called in this contract. 4. Execute randomly one million transactions in the system. We use a single server to establish the test environment which consists of two Intel Pentium CPU, 8 G RAM and SSD for storage.

The contract design presented in Sect. 3.3 meets the first-type contract model in Fig. 3, where the data is mainly distributed on State Trie. The design of the contract indicates that each transaction creates a new contract and only changes the State Trie once, that is $a = 1$(do nothing in `doOrder()`). According to the implementation of MPT, when the branch is full, its size is $s_b = 532 + 32 = 564\,B$, the leaf is smaller than $s_l = 96\,B$, and $s_a = 70{+}32 = 102\,B$. The storage s' is regarded as 0, and the average time consumption of each visit to the database is $t = 0.03$ ms (Since the performance of LevelDB random access will decrease drastically as the amount of data increases, t will continue to increase. For the sake of simplicity, the average value in a test is used here). Further, as LevelDB will compress the stored data, we use the sum of key and value which would be stored in LevelDB for space occupancy instead of the amount of space in disk.

Taking the above data into the equations in Sect. 3.3, the prediction formula can be expressed by

$$H_{max}(n) = \frac{2\sqrt{2log_2(n)}+log_2(n)}{4}$$
$$H_{avg}(n) = \frac{log(n)}{2}$$
$$T_{predict}(n) = \frac{0.06*log(n)}{2}$$
$$S_{predict}(n) = 564 * \left(\frac{log(n)}{2} - 1\right) + 96 + 102$$

4.2 Experiment Result

The experimental data are collected and analyzed. In the following figures, the sampling of the experimental data is represented by a red dot, and the blue curve represents the prediction curve.

Transactions and MPT Height. Figure 4 is the prediction for the height of Storage Trie. The fitting curve of $H_{fitting}(n) = 0.356594log(n) + 1.767782$ is obtained based on the actual sampling data. This curve shows the change of the actual tree height, which is represented by the green curve in the figure. The orange curve is the predicted maximum tree height. It can be found from Fig. 4 that our prediction curve is certainly close to the fitting curve, and the sampling points are mainly distributed on both sides of the curve. Besides, there are almost no sample points going over the orange curve, indicating that the prediction curve can predict the tree height of the State Trie better as the transaction volume increases. At the same time, the prediction of the maximum tree height also indicates the change in the upper limit of the tree height.

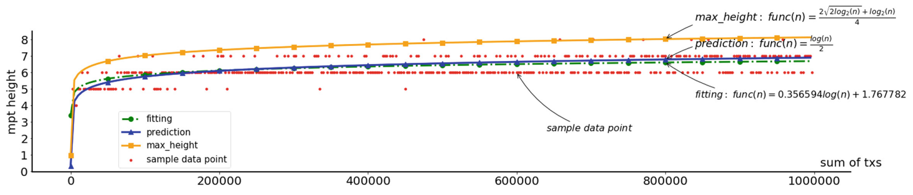


Fig. 4. Transactions and MPT height. (Color figure online)

Transactions and Time Cost. Figure 5 is the prediction for the number of transactions and time cost. As t changes in the production environment, and the change is mainly determined by LevelDB, which is beyond the scope of this paper, a fixed value t is adopted as replacement in the prediction curve. It can be found from Fig. 5 that for the transaction volumes less than 200,000, there are deviations between the predictive value and actual experimental value. This is because the actual t is much smaller than the adopted fixed value. As the transaction volume increases, the experimental data tends to be evenly distributed on both sides of the curve. Although a simple method is used here for the predicted value, the experimental results validate this prediction. An ideal prediction curve can be obtained if the fixed t is replaced by getting the relationship between the read time t and data volume of the LevelDB from an actual test or a theoretical calculation.

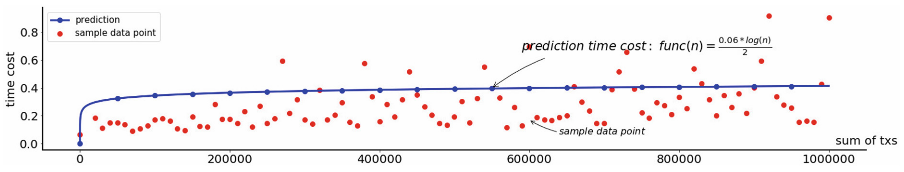


Fig. 5. Transactions and time cost.

Transactions and Storage. Figure 6 is the prediction for the number of transactions and storage occupation. On account of the multi-level cache in the implementation of Ethereum, a scheme of one-transaction-one-block is adopted in testing. Thus, the testing object here is the change in maximum space occupation. In production environment, if multiple transactions are packaged in one block, the data cache will cause the actual stored data to be smaller than the prediction (it is found in testing that if more than 100 transactions are stored in a block, the storage occupancy is roughly half of the prediction). Adding LevelDB compression, the space will be smaller. Thus, it can be concluded that the growth of the storage under the current test conditions is consistent with the prediction.

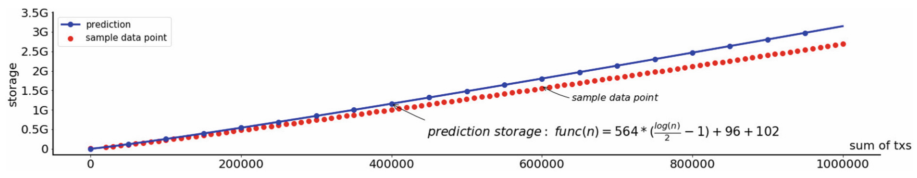


Fig. 6. Transactions and storage

From the results of these three experiments, we can see the prediction method presented in the third chapter is more consistent with the experimental expectation.

5 Conclusions

When companies use the Ethereum consortium-blockchain, they need to make predictions about the prospective performance and storage of the system if the transaction reaches a certain scale. This paper analyzes the core issues that affect the performance and storage of Ethereum is the "World State". According to the analysis result that "World State" is implemented by MPT, the relationship between MPT performance or storage increment and transaction volume n is obtained to be $log(n)$. On the other hand, the "World State" is made up of the upper layer of State Trie and the underlying layer of Storage Trie. The data distribution would be different based on the organization of contracts. The formulas are offered for a business model to predict the relationship between transaction volume and performance/storage based on State Trie. Other business models can be derived by the same method. In this way, the companies can deduce the prospective performance and storage of blockchain according to their own contracts under the premise that its business scale can be predicted (transaction volume).

At the same time, when transaction scale could be estimated, we can properly design the contracts in order to minimize the consumption of performance and storage, so that the data distribution between the State Trie and Storage Trie could reach an inflection point, which minimizes the costs of performance and storage. Therefore, it is suggested that the contract developer can estimate the future transaction volume in advance and properly allocate the data in the state tree and storage tree to achieve the optimal efficiency and storage when writing the contract.

References

1. Enterprise Ethereum Alliance. https://entethalliance.org/
2. GitHub. ethereum/cpp-ethereum. https://github.com/ethereum/cpp-ethereum
3. Etherchain.org. etherchain.org - The Ethereum Blockchain Explorer. https://etherchain.org/
4. GitHub. ethereum/wiki. Merkle Patricia Trie Specification (also Merkle Patricia Tree). https://github.com/ethereum/wiki/wiki/Patricia-Tree
5. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008). https://bitcoin.org/bitcoin.pdf
6. Buterin, V.: A Next-Generation Smart Contract and Decentralized Application Platform, Ethereum White Paper. https://github.com/ethereum/wiki/wiki/White-Paper
7. Wood, G.: Ethereum: a secure decentralised generalised transaction ledger (2014). http://gavwood.com/paper.pdf
8. Pongnumkul, S., Siripanpornchana, C., Thajchayapong, S.: Performance analysis of private blockchain platforms in varying workloads. In: 2017 26th International Conference on Computer Communication and Networks (ICCCN), Vancouver, BC, pp. 1–6 (2017)

9. Atzei, N., Bartoletti, M., Cimoli, T.: A survey of attacks on ethereum smart contracts (SoK). In: Maffei, M., Ryan, M. (eds.) POST 2017. LNCS, vol. 10204, pp. 164–186. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-662-54455-6_8
10. Loi, L., Chu, D.H., Olickel, H., Saxena, P., Hobor, A.: Making smart contracts smarter. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS 2016), pp. 254–269. ACM, New York (2016). https://doi.org/10.1145/2976749.2978309
11. Sankar, L.S., Sindhu, M., Sethumadhavan, M.: Survey of consensus protocols on blockchain applications. In: 2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, pp. 1–5 (2017). https://doi.org/10.1109/ICACCS.2017.8014672
12. Li, Y., Zheng, K., Yan, Y., Liu, Q., Zhou, X.: EtherQL: a query layer for blockchain system. In: Candan, S., Chen, L., Pedersen, T.B., Chang, L., Hua, W. (eds.) DASFAA 2017. LNCS, vol. 10178, pp. 556–567. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-55699-4_34
13. LevelDB github page. https://github.com/google/leveldb
14. Devroye, L.: Universal asymptotics for random tries and PATRICIA trees. Algorithmica **42**, 11 (2005). https://doi.org/10.1007/s00453-004-1137-7
15. Knessl, C., Szpankowski, W.: Heights in generalized tries and PATRICIA tries. In: Gonnet, G.H., Viola, A. (eds.) LATIN 2000. LNCS, vol. 1776, pp. 298–307. Springer, Heidelberg (2000). https://doi.org/10.1007/10719839_31
16. Morrison, D.R.: PATRICIA practical algorithm to retrieve information coded in alphanumeric. J. ACM **15**(4), 514–534 (1968)
17. Knuth, D.E.: The Art of Computer Programming. Sorting and Searching, vol. III. Addison-Wesley, Redwood City (1973)
18. Dhillon, V., Metcalf, D., Hooper, M.: Ethereum Tokens: High-Performance Computing. Blockchain Enabled Applications. Apress, Berkeley (2017)

Smart Contract Programming Languages on Blockchains: An Empirical Evaluation of Usability and Security

Reza M. Parizi[1](✉), Amritraj[1], and Ali Dehghantanha[2]

[1] Department of Software Engineering and Game Development, Kennesaw State University, Kennesaw, GA 30060, USA
rparizi1@kennesaw.edu, amritra@students.kennesaw.edu
[2] Department of Computer Science, University of Sheffield, Sheffield, UK
a.dehghantanha@sheffield.ac.uk

Abstract. Blockchain is a promising infrastructural technology that is finding its way into a growing number of domains like big data, finance, and medical. While blockchain has come to be thought of primarily as the foundation for Bitcoin, it has evolved far beyond underpinning the virtual currency. As it becomes progressively popular, the need for effective programming means would be more demanding. Blockchain programming as a core means provides accounts of the 'code is law' that specifies agreements between parties and allows its stakeholders to still trust the platform to execute the agreed-upon contract (known as smart contract) as expected. Although it seems straightforward in theory, it is hardly the case when it comes to real-life situations. There have been several instances that show smart contracts are riddled with issues and vulnerabilities in code, causing damages. What's for sure is lacking is that the existing languages are not living up to the point to be able to unleash the full potential of the blockchain, as often have resulted in buggy code with a steep learning curve for developers. This denotes that the current research on contract development is not sufficient and is still in a stage of infancy. In order to advance the state of the research in this area, an evaluation of the current state-of-the-art practices in a thorough and experimental manner is required. Thus, the objective of this paper is to give a comprehensive analysis of such domain-specific programming practices from critical points of usability and security to provide a working guideline for newcomers and researchers.

Keywords: Blockchain · Blockchain coding · Smart contract platforms
Smart contract programming · Decentralized computing and development

1 Introduction

Blockchain [1] is a new trend rising fast from the community and the enterprise world. A blockchain is theoretically an incremental list of records called blocks which are linked together and secured using cryptography, forming a chain in the process. Copies of this chain are stored across several peers on a network who can all see the chain and its contents. To add a new block, a peer must find a key to a random pattern generated using cryptography and verify the block itself. As soon as a peer adds a new block, it also broadcasts this addition to all the other peers on the network, so they can update their copies of the blockchain.

Blockchain has already disrupted a wide range of industries including Finance, Cloud computing, Privacy, Security etc. Also, in the recent years, an interesting new application of blockchain has surfaced, i.e. Smart contract [2]. Smart contracts are self-executing contracts where the terms of the agreement between multiple parties are directly written into lines of code. The code and the agreements contained therein exist across a blockchain network. Smart contracts allow trusted transactions and agreements to be carried out among disparate, anonymous parties without the need for a central authority, legal system, or an external enforcement mechanism. They render transactions traceable, transparent, and irreversible. Recognition of the unique challenges of smart contract programming has inspired developers to create domain-specific languages, such as Solidity [3] to ease development.

Although it is a promising domain, Smart contracts, in its first decade has been plagued by unfortunate incidents. In June 2016, vulnerabilities in the DAO code was exploited to empty out more than 2 Million (40 Million USD) ether [4]. The attack took advantage of the reentrancy problem in the 'splitDAO' function of the code. Since, the program was not designed carefully, a call to the function that behaved as a regular call was modified into a recursive call and used to make multiple withdrawals when only one was to be authorized.

Also, in November 2017, a developer [5] whilst fixing a bug that let attackers steal 32 million USD from a few multi-signature wallets accidentally left a second bug in the system that allowed one user to become the sole owner of every single multi-signature wallet. Realizing the mistake, the developers tried to fix damages by deleting the program instead of returning the funds to their original owners. This act of deletion of the program simply locked all the funds in those multi-signature wallets permanently. Unlike most cryptocurrency hacks, however, the money was not deliberately taken instead, it was permanently locked by accident and lack of understanding of the program.

The above incidents show that even the most experienced developers can leave behind security vulnerabilities and bugs that are exploitable and failure prone. Thus, there is still a steep learning curve for developers when it comes to contract programming. This steep learning curve makes it even more difficult for new developers to write correct and safe contracts. As of current date, the state of empirical studies in the domain of smart contract development is still in infancy. Hence, the objective of this paper is to take this initiative by providing an empirical evaluation of smart contract programming languages, in order to shed light on future directions of its development research, education and practices. To this end, we assessed the usability and security vulnerability aspects of three domain-specific languages namely Solidity, Pact and Liquidity (see Sect. 2). The results demonstrated that although Solidity is the most usable language for a new developer to program smart contracts, it is the least secure language to vulnerabilities. While, Liquidity and Pact demonstrated better security results, implying it is harder for new developers to leave behind bugs and security vulnerabilities when working with Pact and Liquidity, but they show less usability compared to Solidity.

The remainder of this paper is organized as follows. Section 2 gives an overview of the smart contract programming languages and state-of-the-art practices for building smart contracts; Sect. 3 presents and describes the details of the experiment and its evaluation results; Sect. 4 presents the related work; and lastly, Sect. 5 reports the conclusion and future work.

2 Smart Contract Programming Languages

In this section, we discuss widely used smart contract programming languages namely Solidity, Pact, Liquidity. We have also discussed sample contracts implemented with each of the mentioned programming languages to provide an insight into real-world contract development.

2.1 Solidity and the EVM

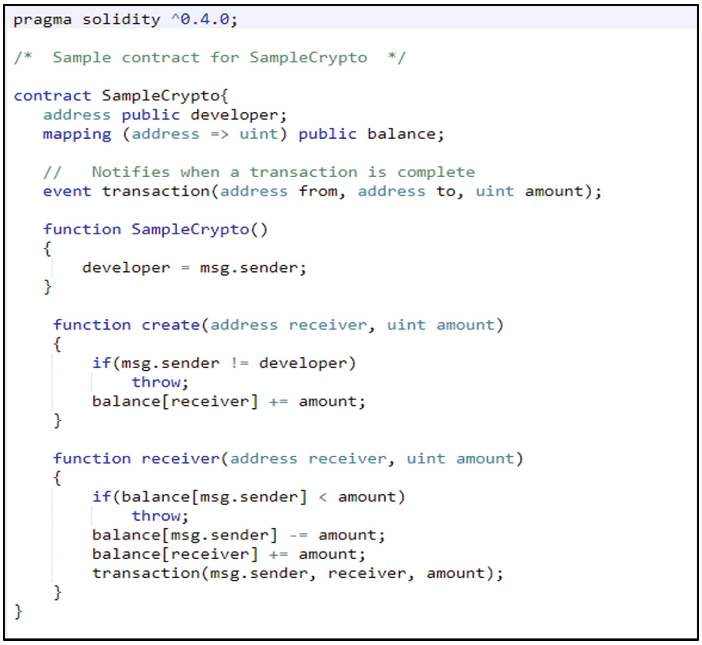
Solidity [3] is a statically-typed programming language with a similar syntax to ECMAScript (JavaScript) built for writing smart contracts. It is the primary choice language for implementing smart contracts on the Ethereum [6] platform.

Ethereum is an open source, decentralized platform for building smart contracts. It facilitates the development and execution of complex applications such as financial exchanges and insurance contracts on a distributed platform. The core of Ethereum is the Ethereum Virtual Machine (EVM), which executes code of random algorithmic complexity. Solidity is designed for developing smart contracts that run on the EVM. Solidity contracts are first compiled to bytecode which is ultimately executed on the EVM.

Like other blockchains, Ethereum includes a peer-to-peer network protocol. The Ethereum blockchain database is maintained and updated by several nodes connected to the network. Every node on the network runs the EVM and executes the same set of instructions.

The Ethereum platform itself is featureless or value-agnostic. It is up to organizations and developers to decide what it should be used for. However, certain application types benefit more than others from Ethereum's capabilities. Specifically, Ethereum is suited for applications that automate direct interaction between peers or facilitate coordinated group action across a network. For instance, applications for coordinating peer-to-peer marketplaces, or the automation of complex financial contracts. When it comes to programming on Ethereum, there are some key points to notice from the Ethereum Design Rationale document [7].

Sample Contract Implementation. In Fig. 1, we show a smart contract for an imaginary cryptocurrency, we named 'SampleCrypto'. SampleCrypto can only be issued by its developer and can be transferred to a receiver with his/her address.



```
pragma solidity ^0.4.0;

/*  Sample contract for SampleCrypto  */

contract SampleCrypto{
   address public developer;
   mapping (address => uint) public balance;

   //   Notifies when a transaction is complete
   event transaction(address from, address to, uint amount);

   function SampleCrypto()
   {
       developer = msg.sender;
   }

    function create(address receiver, uint amount)
    {
        if(msg.sender != developer)
            throw;
        balance[receiver] += amount;
    }

    function receiver(address receiver, uint amount)
    {
        if(balance[msg.sender] < amount)
            throw;
        balance[msg.sender] -= amount;
        balance[receiver] += amount;
        transaction(msg.sender, receiver, amount);
    }
}
```

Fig. 1. SampleCrypto implementation in Solidity

2.2 Pact

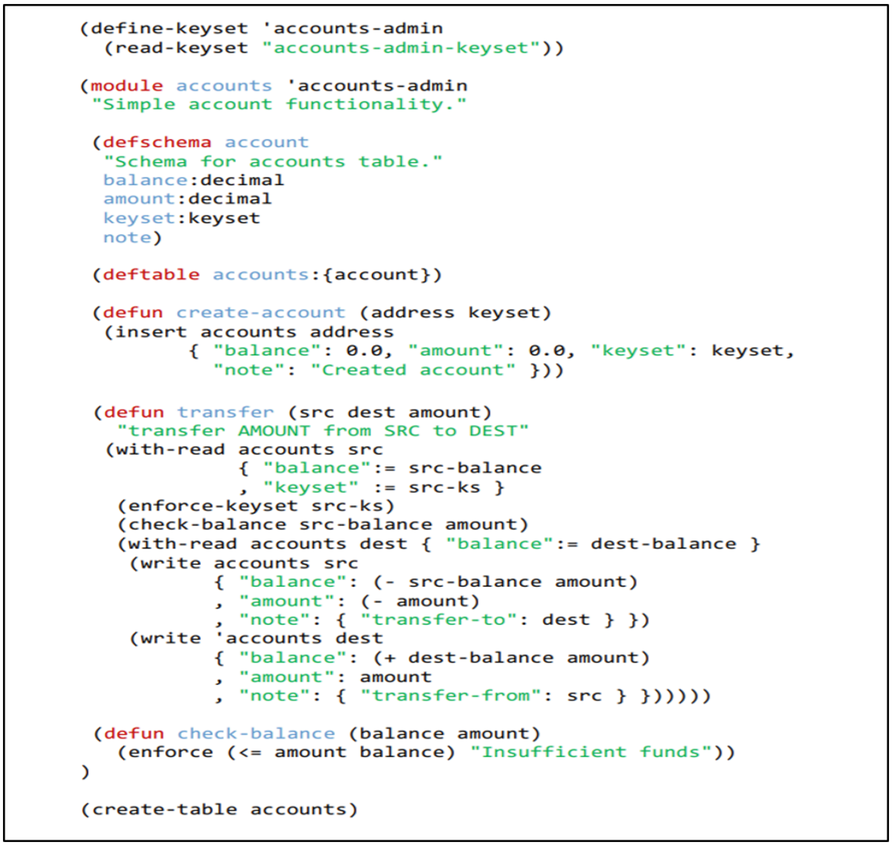
Pact [8] is a programming language for writing smart contracts to be executed by the Kadena [9] blockchain. Pact empowers developers to implement robust, performant transactional logic, executing mission-critical business operations quickly and safely.

Pact is immutable, Turing-incomplete and favors a declarative approach over complex control-flow. This makes bugs harder to write and easier to spot. Pact smart contracts are designed to enforce business rules guarding the update of a system-of-record: complex, speculative application logic simply does not belong in this critical layer.

Sample Contract Implementation. We present an example Pact code [8], implementing a simple "account balance" smart contract, with functions to create accounts and transfer funds, in Fig. 2. A detailed description of the above contract including information on Installing the module, Keyset Definition, Module Definition, Table Creation and finally, Invoking the 'accounts' module can be found at [8].

2.3 Liquidity

Liquidity [10] is a high-level typed smart-contract language for Tezos [11]. It is a fully typed functional language, it uses the syntax of OCaml [12] and strictly complies with Michelson [13] security restrictions. A formal verification framework for it is under development, to prove the correctness of smart-contracts written in Liquidity.



```
(define-keyset 'accounts-admin
  (read-keyset "accounts-admin-keyset"))

(module accounts 'accounts-admin
 "Simple account functionality."

 (defschema account
  "Schema for accounts table."
  balance:decimal
  amount:decimal
  keyset:keyset
  note)

 (deftable accounts:{account})

 (defun create-account (address keyset)
  (insert accounts address
         { "balance": 0.0, "amount": 0.0, "keyset": keyset,
           "note": "Created account" }))

 (defun transfer (src dest amount)
   "transfer AMOUNT from SRC to DEST"
  (with-read accounts src
             { "balance":= src-balance
             , "keyset" := src-ks }
   (enforce-keyset src-ks)
   (check-balance src-balance amount)
   (with-read accounts dest { "balance":= dest-balance }
    (write accounts src
           { "balance": (- src-balance amount)
           , "amount": (- amount)
           , "note": { "transfer-to": dest } })
    (write 'accounts dest
           { "balance": (+ dest-balance amount)
           , "amount": amount
           , "note": { "transfer-from": src } })))))

 (defun check-balance (balance amount)
   (enforce (<= amount balance) "Insufficient funds"))
)

(create-table accounts)
```

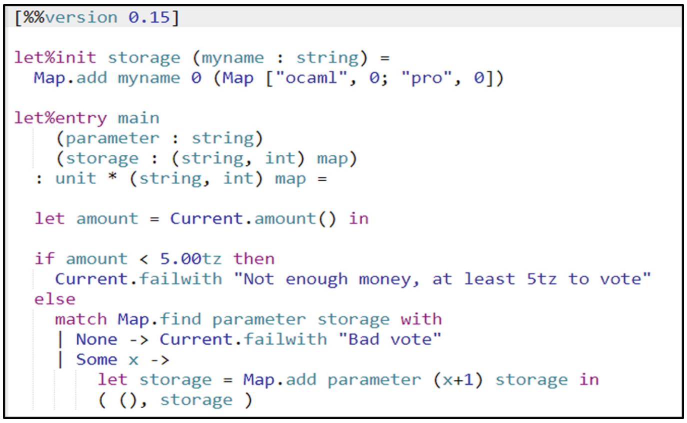
Fig. 2. Account balance smart contract in Pact

The Liquidity language provides three key features: (1) full coverage of the Michelson language: anything that can be written in Michelson can be written in Liquidity. (2) local variables instead of stack manipulations: values can be stored in local variables. The only restriction is that local variables do not survive to Contract.call, following the philosophy of Michelson to force explicit storage of values to limit reentrancy bugs. (3) high-level types: types like sum-types and record-types can be defined and used in Liquidity programs. Liquidity's contract format can be found in [10].

Sample Contract Implementation. The following contract [14], shown in Fig. 3, is a simple voting system that requires a user to have at least 5 tz to submit a vote.

The contract will display an error message "Not enough money, at least 5 tz to vote" if the user attempts to vote with a balance lower than 5 tz.

In Table 1, we have summarized the smart contract programming languages discussed in this section. The table lists the major platform that supports or plans to support these languages. We have also listed some of the key features of these languages in the designated column.



```
[%%version 0.15]

let%init storage (myname : string) =
  Map.add myname 0 (Map ["ocaml", 0; "pro", 0])

let%entry main
    (parameter : string)
    (storage : (string, int) map)
  : unit * (string, int) map =

  let amount = Current.amount() in

  if amount < 5.00tz then
    Current.failwith "Not enough money, at least 5tz to vote"
  else
    match Map.find parameter storage with
    | None -> Current.failwith "Bad vote"
    | Some x ->
        let storage = Map.add parameter (x+1) storage in
        ( (), storage )
```

Fig. 3. Smart contract with Liquidity

Table 1. Summary of smart contract programming languages

Programming languages	Major platforms	Key features
Solidity	Ethereum	• Statically typed • Supports inheritance, libraries and complex user-defined types
Pact	Kadena	• Turing-incomplete safety-oriented design • Human-readable, on-ledger code • Atomic execution (transactions) • Module definition and import • Unique “key-row” + columnar database metaphor • Expressive syntax and function definition • Single-signature and multi-signature public-key authorization • Type inference
Liquidity	Tezos	• High-level types: types like sum-types and record-types can be defined and used in Liquidity programs • Full coverage of the Michelson language: Anything that can be written in Michelson can be written in Liquidity, • Local variables instead of stack manipulations

3 Empirical Evaluation

The goal of our empirical study is to determine the usability and security vulnerabilities of the smart contract programming languages discussed in the previous section. Our study was designed around the scenario in which the formal descriptions of three smart contracts were provided to human test subjects to implement using an assigned smart contract programming language while assessing the usability and analyzing the types of bugs and security vulnerabilities that developers can leave behind in the contracts. We therefore, designed our experimental study based on the following research questions (referred to as RQ's):

RQ1. How do the languages under study stack up in terms of usability to new contract developers?

RQ2. What are the common security issues left behind in the contract by new developers?

3.1 Experimental Planning

Conducting an empirical study involving human subjects can lead to several challenges and pitfalls. Guidelines exist in the literature [15] to help researchers to carry out such type of studies. These guidelines helped us to design our experiment, especially because a frequent problem with controlled empirical studies is that, due to their cost and complexity, they are often limited in size.

We have divided our experimental planning into four parts, namely *test subjects*, *test contracts*, *measures*, and *experimental design*. We discuss the experimental setup in the sub-sections below:

Test Subjects. We sent email invitations to undergraduate students and graduate research assistants in the College of Computing and Software Engineering (CCSE) at Kennesaw State University (KSU), US to participate in our study. The email described the aim and objective of our experiment (which is to perform an empirical evaluation of smart contracts programming language based on usability and security), location, time, expected length of the experiment and an RSVP link. We received a response from a total of 15 undergraduate students and Graduate research assistants within the mentioned deadline. Each subject had prior experience with at least one general purpose programming language and object-oriented concepts.

Test Contracts. We prepared formal descriptions of three test contracts in the form of scenario paragraphs for the test subjects in our experiment. The three contracts were selected after a careful evaluation of several smart contracts from various online sources that are prone to security vulnerabilities when implemented by new developers. The prepared formal descriptions were carefully checked for completeness and ambiguity.

Table 2 gives a brief description of the three test contracts chosen for our experiment. Each test subject had to implement these three contracts in a randomly assigned smart contract programming language, as described in detail in the experimental design.

Measures. To quantify usability, we used a built-in timer (as part of the helper program in the experiment environment) to measure the average times of implementation of each contract in an assigned programming language for each test subject (in minutes). The longer the subjects took to implement a contract, the lower would be the usability of the language to a new developer. To arrive at more solid results, we additionally asked the subjects to answer a questionnaire regarding the usability of the assigned smart contract programming language at the end of the experiment in the exit

Table 2. Summary of the test contracts for our study

Test contract	Description	Reference
Contract 1: HoneyPot	A contract to keep a record of balances for each address that puts currency in it and allow these addresses to get them later	[16]
Contract 2: Bank Account	A contract to deposit/withdraw money into a user's bank account	[17]
Contract 3: King of the currency	A simple contract in which the highest bidder becomes the leader of a group	[18]

questionnaire. We asked the subjects to rate the usability of the language on a scale of 1.0–10.0, 1.0 being extremely difficult to use and 10.0 being extremely usable. We also asked them for their comments on the language, such as what did they find easy? What did they find difficult?

We used a two-facet method to capture and analyze security vulnerabilities in the implemented smart contracts. Firstly, we ran the implemented contracts against six known security vulnerabilities including, *Callstack Depth Attack Vulnerability*, *Reentrancy Vulnerability*, *Assertion Failure*, *Timestamp Dependency*, *Parity Multigeniture Bug 2* and *Transaction-Ordering Dependence (TOD)* with the help of 'OYENTE' [19, 20] tool. OYENTE is an automated security analysis tool for revealing the above-mentioned security vulnerabilities in smart contracts. Secondly, we analyzed the implemented contracts manually to check for further vulnerabilities that were not covered by the tool including DoS (Denial of Service) with (Unexpected) revert and DoS with Block Gas Limit in Solidity [21]. The more these vulnerabilities surface in a contract (from both automated and manual parts), the less secure the underlying contract programming language would be.

Experimental Design. We prepared 15 envelopes each of which contained the formal description of the three smart contracts and the smart contract programming language that these contracts need to be implemented with. Each envelope was also assigned an Envelope ID number which helped us to keep track of the language the contracts need to be implemented in. The envelopes were prepared such that only 5 envelopes would contain the language L_i (where $L_i \in$ {Solidity, Pact, Liquidity}). Hence, out of our 15 test subjects, only 5 random chosen subjects would implement the test contracts in a language L_i. We made sure that we assigned test subjects computers such that no two subjects who had to implement the contracts in the same language sit alongside each other. Each language was to be implemented on an online compiler, i.e. we used Remix [22] for Solidity, Try-Pact [23] editor/compiler for Pact, and Liquidity online editor/compiler [14] for Liquidity.

Table 3 summarizes the organization of our experiment. As shown in the table, there were 5 test subjects who worked on implementing the given three smart contracts in the assigned smart contract programming language.

Table 3. Organization and assignment of envelopes and languages in the experiment

Subject ID	Envelope ID	Language
01, 04, 07, 10 and 13	03, 06, 09, 12 and 15 respectively	Solidity
02, 05, 08, 11 and 14	01, 04, 07, 10 and 13 respectively	Pact
03, 06, 09, 12 and 15	02, 05, 08, 11 and 14 respectively	Liquidity

3.2 Experimental Execution

To avoid fatigue, we decided to conduct our experiment in two sessions. The first session was a background and demo session. During this session, each subject received a starter pack consisting of their subject ID, a statement of consent, a background questionnaire, instructions regarding the experiment, a printout of the demo slides and an exit questionnaire. Before commencing with the demo, each subject was required to fill in the questionnaire based on their background and programming experience and sign a statement of consent to participate in our experimental study.

After the statement of consent and background questionnaire were signed, completed and collected, we proceeded with a small presentation on the basics of smart contracts programming to familiarize the subjects with the same. Next, we conducted a Q&A session with the subjects to answer any of their questions and concerns. When all the questions were answered and any confusions cleared, we asked the test subjects to take a 45-min break to refresh themselves. We also asked them to keep their starter packs with them in case they needed to review the slides during the break.

After the break, we commenced the second session of our experiment. This session was for the practical implementation of test contracts. Each test subject was handed a sealed envelope with an envelope ID number on it (this helped us to keep track of the smart contract language that the envelope's contracts need to be implemented in). Each envelope had a formal description of the three test contracts and the language in which the subjects were assigned to implement these contracts. After all the envelopes were handed, we matched the subject ID's with envelope ID's to keep a track on our experiment. We then asked the subjects to open their envelopes and read all the problem statements thoroughly, we then conducted a second Q&A session to remove all doubts and confusions regarding the problem statements and the programming languages that they were assigned. When this was over, we asked them to implement a simple warm-up "Hello world" exercise in the language they were assigned. Finally, when all the subjects were done with the warm-up exercise, we asked them to begin working on their problems and started a timer for each subject.

The subjects were given two hours to implement all the contracts, and we asked them to remain seated even if they finished their task before the time limit. To be considered "finished", we required them to be certain that their test contracts compiled successfully on the online compilers mentioned earlier.

Including presentation and break, the duration of the experiment was three hours and forty-five minutes. The experiment was completed under "exam conditions", i.e., subjects were not allowed to communicate with others, or consult with other sources to avoid introducing biases into the experimental findings. Finally, after the end of the experiment, each subject was asked to fill in the exit questionnaire before leaving.

3.3 Experimental Results and Analysis

Following the experimental execution process described in the preceding section, we collected the required experimental data and carefully analyzed all the data collected to arrive at conclusions. We now present these results in response to our research questions.

RQ1: How do the languages under study stack up in terms of usability to new contract developers? The data collected for measuring the usability of smart contracts programming languages is shown in Tables 4 and 5. Table 4 summarizes the average implementation times of each test smart contract with a given smart contract programming language (i.e. Solidity, Pact and Liquidity). We found that the average implementation time of each contract was significantly lower in case of Solidity as compared to Pact and Liquidity.

Table 4. Summary of the implementation times of the test contracts

Language	Average implementation time (Contract 1)	Average implementation time (Contract 2)	Average implementation time (Contract 3)	Total average implementation time
Solidity	13 min 26 s	20 min 07 s	19 min 14 s	52 min 47 s
Pact	17 min 31 s	31 min 16 s	33 min 32 s	82 min 19 s
Liquidity	14 min 21 s	23 min 21 s	27 min 51 s	65 min 33 s

We observed the data extracted from the experiment to be consistent across all three test contracts and languages for all test subjects, i.e. each test subject who implemented the three test contracts with Solidity did so faster than every test subject who implemented the contracts with Pact or Liquidity. Similar observation was made in case of Liquidity and Pact, i.e. each subject who implemented test contracts with Liquidity did so faster than every subject who implemented the contracts with Pact.

In Fig. 4, we represent the average implementation times of each test contract (shown in Table 4) with each smart contract programming language in our experiment. We made an interesting observation for Test Contract 1, i.e. the average implementation times of test contract 1 with Solidity and Liquidity were almost similar, i.e. $\Delta t_{LS1} < 1$ min (where Δt_{LS1} = Average implementation time of test contract 1 with Liquidity - Average implementation time of test contract 1 with Solidity = 55 s). But, this time difference increased significantly (i.e. Δt_{LS2} = 3 min 14 s and Δt_{LS3} = 8 min 37 s) with the increased complexity of test contracts 2 and 3 as compared to test contract 1. We made another anomalous observation for implementation times of test contract 2 and 3, i.e. In case of Solidity, the average implementation time of test contract 3 is lower than average implementation time of test contract 2. On the other hand, this is not the case for Pact and Liquidity, as the average implementation time of test contract 3 is higher with these languages when compared to the average implementation time of test contract 2. But, since, the average implementation time of test contract 2 and 3 is lower with Solidity when compared to average implementation times with Pact and Liquidity, this observation has negligible value.

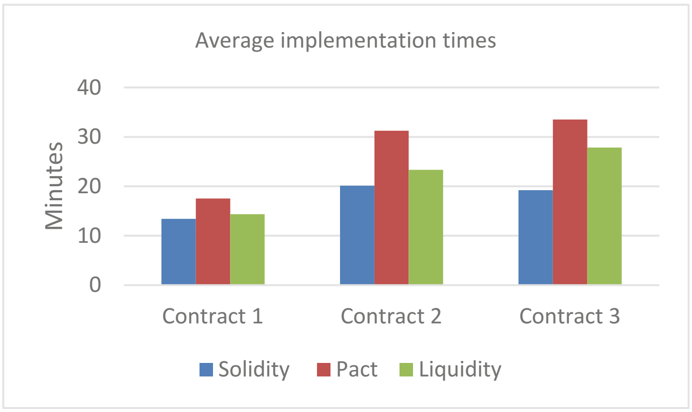


Fig. 4. Average implementation times of all contracts with Solidity, Pact and Liquidity

Additionally, Fig. 5 represents the total average implementation times of all three test contracts implemented with each smart contract programming language in our experiment, i.e. Solidity, Pact and Liquidity. The figure shows that the total average implementation time of all test contract with Solidity is 52 min and 47 s. The total implementation time increased by 24.2% to 65 min and 33 s with Liquidity and almost by 56% to 82 min and 19 s with Pact as compared to total average implementation time with Solidity.

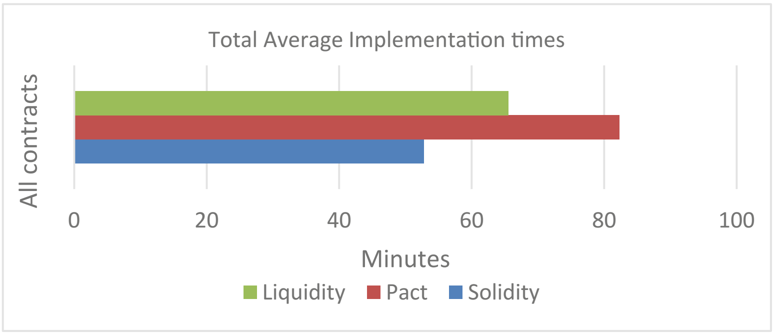


Fig. 5. Total average implementation times of all contracts with Solidity, Pact and Liquidity

Finally, Table 5 summarizes the results from the exit questionnaire which was completed by the test subjects at the end of the experiment. The table represents the average usability score as graded by the test subjects to their respective assigned language for the experiment. The higher the average usability score (see Sect. 3.1 - Measures) the more usable the language is to a new developer.

Table 5. Average usability score from the exit questionnaire of the test subjects

Language	Average usability score by test subjects
Solidity	8
Pact	4.5
Liquidity	5.5

Summarizing the overall usability results of our experiment, we see that the average implementation times of each contract are such that - $t_{SC1} < t_{LC1} < t_{PC1}$ for test contract 1, $t_{SC2} < t_{LC2} < t_{PC2}$ for test contract 2 and $t_{SC3} < t_{LC3} < t_{PC3}$ for test contract 3 respectively (where, t_{SCi}, t_{LCi} and t_{PCi} are the average implementation times of test contract ‘i’ with Solidity, Liquidity and Pact respectively and $i \in \{1,2,3\}$). This leads to the subsequent result regarding the total implementation times of all 3 test contracts, i.e. $T_S < T_L < T_P$ (where, T_S, T_L and T_P are the total average implementation times of all 3 test contracts). Additionally, from the results of the exit questionnaire in Table 5 we see – $U_S > U_L > U_P$ (where, U_S, U_L and U_P are the average usability scores of Solidity, Liquidity and Pact respectively). For a language to have higher usability, we require it to have faster implementation times and high usability scores in the exit questionnaire. Hence, it is clear from the results presented in this section that the usability of Solidity > Liquidity > Pact for a new developer.

RQ2: What are the common security issues left behind in the contract by new developers? We analyzed the implemented test contracts for security vulnerabilities using the ‘OYENTE’ tool and methods described in Sect. 3.1. While we couldn’t find any security vulnerabilities for Liquidity and Pact implemented contracts similar could not be said for Solidity implemented contracts. The results of our security analysis are summarized in Table 6.

Table 6. Security issues found in the implemented test contracts

Contract	Solidity	Pact	Liquidity
Contract 1	Reentrancy vulnerability - (5/5 contracts)	None	None
Contract 2	Reentrancy vulnerability - (1/5 contracts)	None	None
Contract 3	DoS with (Unexpected) revert - (5/5 contracts)	None	None

For test contract 1, we found that all implemented contracts by the test subjects were vulnerable to Reentrancy attacks. Similar results were found for test contract 3, where all the implemented contract were prone to DoS with (unexpected) revert vulnerability [21]. Meanwhile, for test contract 2, only one of the implemented contract was vulnerable to Reentrancy vulnerability.

A Reentrancy attack occurs when a function ‘x’ calls a function ‘y’ in an external contract, which makes a reentrant call to ‘x’. If x’s call to ‘y’ occurs while the contract is in an inconsistent state, then the reentrant call may make invalid assumptions about the initial state of the contract. This reentrancy vulnerability was recently exploited to steal over $40 million [24]. This problem is difficult and error-prone to avoid in

Solidity, this is because one shall reason about reentrant call anytime an external call is made from a function. Since, sending money always results in an external call when using Solidity, this becomes a frequent vulnerability. In general, external calls from function ‘x’ in contract C may invoke additional calls on C to any function, not just ‘x’, via an intermediate external invocation. This is more problematic than internal-only calls because the external contract is likely to assume C is in a consistent state.

Figure 6 visually represents the security vulnerabilities in all the implemented contracts with the Solidity, Pact and Liquidity languages. No security vulnerabilities were found in the Pact and Liquidity implemented contracts but 11 (approx. 73%) Solidity contracts were found to be vulnerable out of a total of 15 implemented. Out of these 11 vulnerable contracts, 6 had Reentrancy vulnerabilities and 5 were vulnerable to DoS with (unexpected) revert. Only 4 out of 15, i.e. about 27% of Solidity implemented contracts were found to be secure.

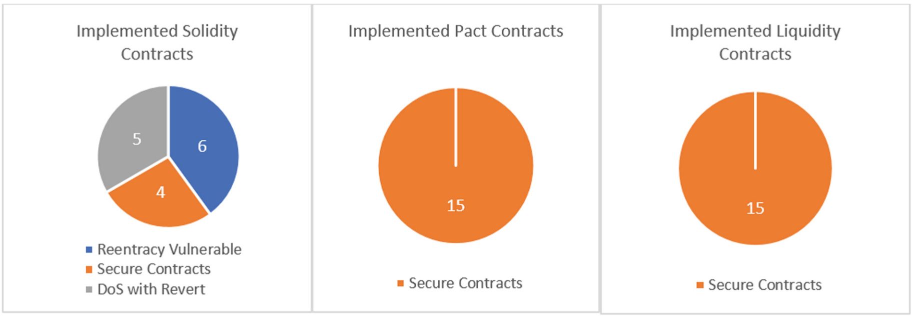


Fig. 6. Security vulnerabilities in all implemented contracts

While using Solidity it is often difficult for even experienced developers to avoid certain pitfalls [5], it is no surprise that the implemented Solidity contracts were prone to security vulnerabilities as the subjects for our experiment can be considered as new and inexperienced contract developers. Hence, from the results of our experiment and in response to our second research question: RQ2, we found that contracts implemented by new developers with Solidity are more prone to security vulnerabilities as compared to smart contracts implemented with Liquidity and Pact. Even though, no vulnerabilities were found in Pact and Liquidity implemented contracts, this by no means necessarily suggests that Pact and Liquidity contracts are 100% immune to security vulnerabilities. It just implies that it is harder for new developers to leave behind bugs and security vulnerabilities when working with Pact and Liquidity as compared to Solidity.

3.4 Threats to Validity

This section discusses the threats to validity for this experiment. The threats to external validity primarily answer the question of how representative the human subjects, the test contracts, and used tools are. Some of our subjects were essentially students and did not have professional smart contract development experience. However, analysis of the results indicated that subjects had similar programming experience and managed to implement their contracts in similar times as others in their group. The test contracts used in our study were not developed by the subjects and may have been unfamiliar to them. To mitigate this, we had organized Q&A and warm-up sessions. We made sure to provide sufficient time to our test subjects for the experiment and this was confirmed in the exit questionnaire, where we asked the subjects if they felt they had been given enough time for the experiment. All test subjects stated to have had enough time to complete the experiment. Additionally, the test contracts selected for the experiment can be considered relatively simple for experienced developers. Hence, it is possible more complex contracts may yield different results. As no previous human studies have been done in this area, we believe beginning with reasonable-scale studies and the lessons learned is prudent to pave the way for larger studies.

The threats to internal validity are implementation effects that could have possibly biased our test results. We designed the formal descriptions of the chosen test contracts carefully for intuition. To check the completeness of our description we conducted pilot research where we provided the prepared formal description to experienced developers and professors for constructive criticism, fault detection and completeness checking. Only when all errors were rectified in our formal descriptions, we decided to go ahead and commence our experiment.

3.5 Discussion

Based on the presented results and comparisons from our experiment, we found that Solidity is the most usable language to a new developer when it comes to programming smart contracts. We found from our exit questionnaire, that this was because of Solidity's intimacy to general purpose programming languages such as Java or C#, which are often used by developers and students in professional and academic environments respectively.

But unfortunately, when it comes to security vulnerabilities in smart contract implemented by new developers, Solidity lacks behind as we found it to be most prone to security vulnerabilities. Although being usable is a huge plus, on the other hand being prone to security vulnerabilities is a huge downside as these security vulnerabilities can be exploited by malicious users to cause financial damages as seen from the recent attacks on the Ethereum platform [25]. Meanwhile, Liquidity and Pact are still new languages which lack high usability at this time but seem more secure than Solidity for now. Ultimately, all the three languages have their pros and cons. The exciting thing to note here is all the three languages are changing and evolving with time and research. Hence, our work aims to help contribute towards forming the body of knowledge for the continuous growth and evolution of this infant domain.

4 Related Work

Our work is related to assessing and comparing the programming languages for smart contract development. There are in fact very few numbers of related studies known for evaluation of smart contract programming languages or platforms. In this section, we present an overview of all the related work, which has been proposed in the literature in recent years.

The work in [26], compares Ethereum, IBM Open Blockchain (Hyperledger project) [27], Intel Sawtooth lake [28], BlockStream Sidechain Elements [29] and Eris [30] platforms. This study suggests Ethereum to be the primary choice of platform in terms of scalability, development, documentation and support. As Intel Sawtooth Lake wasn't fully implemented at the time, the authors conclude that Ethereum is the better solution for developers as it had no security issues known at the time.

In another work [31], the authors analyze the usage of smart contracts platforms from various perspectives. The study examines a sample of 6 platforms, namely Bitcoin [32], Ethereum [33], Counterparty [34], Stellar [35], Monax [36] and Lisk [37] for smart contracts by highlighting some of the key differences in terms of type of blockchain, contract language and volume of daily currency transfers. A sample of 834 contracts was studied for the 2 platforms — Bitcoin and Ethereum, categorizing each of them by application domain, and measuring the relevance of each of these categories. They concluded that about 80% of the Ethereum contracts use at least one of the nine design patterns presented in the paper.

There have also been assessment kind of works that study different blockchain technologies. Anderson et al. [38] compare three blockchains - Ethereum, Namecoin, and Peercoin. For Ethereum, the authors briefly analyze the issues that are introduced by the negligent design of smart contracts. In the case of Namecoin, the focus was, how the name registration is used and had developed over time. For Peercoin, the interest was in the use of proof-of-stake. Similarly, Seijas et al. [39] compare a variety of smart contract platforms. Their work also provides an overview of the scripting languages used in cryptocurrencies, particularly scripting languages of Bitcoin, Nxt and Ethereum. Their work covers technologies that might be used to underpin extensions and innovations in scripting and contracts, including technologies for verification (e.g., zero-knowledge proofs, proof-carrying code and static analysis), as well as approaches to making systems more efficient, e.g. Merkelized Abstract Syntax Trees.

Studies that analyze the security of Ethereum smart contracts have been growing recently. For instance, the work proposed in [25] surveys vulnerabilities and attacks on the Ethereum contracts, while, works [20, 40] propose analysis techniques to detect these vulnerabilities.

Majority of the work listed in this section mainly compares and analyzes the various smart contract platforms. There has been a lack of empirical data on how domain-specific smart contract programming languages might work in tandem with developers and their comparative quality measures such as usability and security. Hence, our work took the first step by providing an experimental analysis of three current domain-specific programming languages. We hope that our proposed work can be useful in the process of building a body of empirical knowledge helping developers and organizations write safer and more secure smart contracts.

5 Conclusion and Future Work

Research on smart contract programming's evaluation is quite young and there is still a long road ahead to reach its maturity. This paper realizes an evaluation of the current languages as a fundamental step towards reaching this maturity and obtaining useful advances. The given evaluation included an experiment that was performed to compare the usability and security vulnerability of the three domain-specific languages, namely Solidity, Pact and Liquidity. The experiment results demonstrated that although Solidity is the most usable language for a new developer to program smart contracts, it is the least secure language to vulnerabilities. On the other hand, Liquidity and Pact show lower usability but seem secure for now. Consequently, our results contribute to the body of experimental evidence about the usability and security of the smart contract programming languages, which is currently scarce.

In future, we intend to conduct more experiments in order to improve the generalizability of our results in this paper. There are several points that can be suggested to be reinforced towards obtaining more solid conclusions for the comparison of smart contract programming languages. These might include the conduction of experiments with (i) extra object programs (test contracts) that will comprise larger systems with varied context parameters such as application domain or size; (ii) more diversified-background of human subjects; (iii) new upcoming smart contract programming languages that are being developed.

References

1. Peck, M.E.: Blockchains: how they work and why they'll change the world. IEEE spectrum (2017)
2. Cuccuru, P.: Beyond bitcoin: An early overview on smart contracts. Int. J. Law Inf. Technol. **25**, 179–195 (2017)
3. Solidity. https://solidity.readthedocs.io/en/develop/
4. Sirer, E.G.: Thoughts on The DAO Hack. http://hackingdistributed.com/2016/06/17/thoughts-on-the-dao-hack/
5. Hern, A.: "$300 M in Cryptocurrency" Accidentally Lost Forever Due To Bug. https://www.theguardian.com/technology/2017/nov/08/cryptocurrency-300m-dollars-stolen-bug-ether
6. Ethereum Project. https://www.ethereum.org/
7. Design Rationale. https://github.com/ethereum/wiki/wiki/Design-Rationale
8. Popejoy, S.: The pact smart-contract language (v1.5), pp. 1–15 (2017)
9. Kadena. http://kadena.io/#/
10. Liquidity, a simple language over Michelson. https://github.com/OCamlPro/liquidity/blob/master/docs/liquidity.md
11. Tezos. https://www.tezos.com/
12. OCaml Documentation. https://ocaml.org/docs/

13. Ii, S.: Michelson : the language of Smart Contracts in I - Semantics
14. Liquidity Online Editor. http://www.liquidity-lang.org/edit/
15. Kitchenham, B.A., Pfleeger, S.L., Pickard, L.M., Jones, P.W., Hoaglin, D.C., El Emam, K., Rosenberg, J.: Preliminary guidelines for empirical research in software engineering. IEEE Trans. Softw. Eng. **28**, 721–734 (2002)
16. Guimaraes, G.: Reentrancy attack on smart contracts: how to identify the exploitable and an example of an attack contract. https://medium.com/@gus_tavo_guim/reentrancy-attack-on-smart-contracts-how-to-identify-the-exploitable-and-an-example-of-an-attack-4470a2d8dfe4
17. Martinsson, F.: Smart contract programming on Ethereum - solidity beginners tutorial part 2. https://www.youtube.com/watch?v=F4XQFEievJI
18. Konstantopoulos, G.: How to secure your smart contracts: 6 solidity vulnerabilities and how to avoid them (Part 2). https://medium.com/loom-network/how-to-secure-your-smart-contracts-6-solidity-vulnerabilities-and-how-to-avoid-them-part-2-730db0aa4834
19. Oyente. https://oyente.melon.fund/#version=soljson-v0.4.21+commit.dfe3193c.js
20. Luu, L., Chu, D.-H., Olickel, H., Saxena, P., Hobor, A.: Making smart contracts smarter. In: Proceedings of 2016 ACM SIGSAC Conference on Computer and Communications Security – CCS 2016, pp. 254–269 (2016)
21. Smart Contracts - Best practices (Known attacks). https://github.com/ConsenSys/smart-contract-best-practices/blob/master/docs/known_attacks.md
22. Remix. http://remix.ethereum.org/#optimize=false&version=soljson-v0.4.21+commit.dfe3193c.js
23. Try Pact. http://kadena.io/try-pact/
24. Omohundro, S.: Cryptocurrencies, smart contracts, and artificial intelligence. AI Matters **1**, 19–21 (2014)
25. Atzei, N., Bartoletti, M., Cimoli, T.: A survey of attacks on Ethereum smart contracts (SoK), pp. 1–24 (2017)
26. Macdonald, M., Liu-Thorrold, L., Julien, R.: The blockchain: a comparison of platforms and their uses beyond bitcoin. Work. Pap., pp. 1–18 (2017)
27. Hyperledger. https://www.hyperledger.org/
28. Intel: Intel: Sawtooth Lake. https://intelledger.github.io/
29. BlockStream Sidechain Elements. https://blockstream.com/technology/
30. Documentation for Eris. https://abal.moe/Eris/docs
31. Bartoletti, M., Pompianu, L.: An empirical analysis of smart contracts: platforms, applications, and design patterns (2017)
32. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system, p. 9 (2008). www.Bitcoin.Org
33. Buterin, V.: A next-generation smart contract and decentralized application platform. http://buyxpr.com/build/pdfs/EthereumWhitePaper.pdf
34. Counterparty: Protocol Specification. https://counterparty.io/docs/protocol_specification/
35. Stellar. https://www.stellar.org/
36. Monax. https://monax.io/
37. Lisk. https://lisk.io/
38. Anderson, L., Holz, R., Ponomarev, A., Rimba, P., Weber, I.: New kids on the block: an analysis of modern blockchains (2016)
39. Seijas, P.L., Thompson, S., McAdams, D.: Scripting smart contracts for distributed ledger technology. Cryptology ePrint Archive, Report 2016/1156 (2016). http://eprint.iacr.org/2016/1156
40. Bhargavan, K., Swamy, N., Zanella-Béguelin, S., Delignat-Lavaud, A., Fournet, C., Gollamudi, A., Gonthier, G., Kobeissi, N., Kulatova, N., Rastogi, A., Sibut-Pinote, T.: Formal verification of smart contracts. In: Proceedings of 2016 ACM Workshop on Programming Languages and Analysis for Security – PLAS 2016, pp. 91–96 (2016)

Applying Design Patterns in Smart Contracts

A Case Study on a Blockchain-Based Traceability Application

Yue Liu[1], Qinghua Lu[1,2,3](✉), Xiwei Xu[2,3], Liming Zhu[2,3], and Haonan Yao[1]

[1] College of Computer and Communication Engineering, China University of Petroleum (East China), Qingdao, China
qinghua.lu@data61.csiro.au
[2] Data61, CSIRO, Sydney, Australia
[3] School of Computer Science and Engineering, UNSW, Sydney, Australia

Abstract. Blockchain, the technology Bitcoin lives on, is an emerging research field due to its nature of decentralisation, and properties of data immutability and transparency. Smart contracts are the programs executed on programmable infrastructure provided by blockchain, which can manage complex business logic, extending the field significantly. As blockchain technology is still at an early stage, there are little works on applying software architectural methods to the design of blockchain-based applications. In this paper, we summarise eight smart contract design patterns based on existing smart contracts and our experience, and classify them into four categories: *Creational Patterns*, *Structural Patterns*, *Inter-Behavioral Patterns*, and *Intra-Behavioral Patterns*. We share some experiences of applying the presented design patterns of smart contract on a real-world blockchain-based traceability application, and also discuss how patterns can improve the quality attributes of blockchain-based application.

Keywords: Blockchain · Smart contract · Interoperability
Adaptability

1 Introduction

Blockchain, the technology behind Bitcoin [5], is a decentralised append-only data store, where all participants in the network can reach agreements on the states of transactional data, without relying on a centralised system. Data transparency and immutability are the key characteristics of blockchain technology, which can help prevent tempering or revising the submitted transactions on blockchain.

Besides the distributed ledger as a data storage, blockchain provides a general-purpose programmable infrastructure. *Smart contracts* [6] are programs deployed and running on blockchain, which can express triggers, conditions and business logic [9] to enable more complex programmable transactions. Many startups, enterprises, and governments [7] are currently exploring blockchain applications in areas as diverse as supply chain, electronic health records, voting, energy supply, ownership management, identity management, and protecting critical civil infrastructure. However, since blockchain technology is still at an early stage, there are little works on applying software architectural methods to the design of blockchain-based applications, particularly design of smart contracts.

In software architecture community, a blockchain taxonomy has been proposed to compare different blockchain platforms and assist in the design and evaluation of software architectures using blockchain technology [11]. Other than taxonomy, architectural design patterns is also a mechanism to classify and organise the existing solutions.

A design pattern is a reusable solution to a problem that commonly occurs within a given context during software design [3]. We investigate some existing patterns for distributed system, peer-to-peer system and software design patterns in general, and assess the applicability of the existing patterns to the design of smart contracts. The study results in the experiences that there are some reusable solutions that can be applied to the design of smart contract in a blockchain-based system.

In this paper, we first summarise and classify eight smart contract design patterns. The patterns are divided into four categories: *Creational Patterns*, *Structural Patterns*, *Inter-Behavioral Patterns* and *Intra-Behavioral Patterns*. By using the patterns, blockchain can not only be used for storing or exchanging data, but also handle with more complicated programs with complex logic, which can benefit developers on building blockchain-based applications. Besides, we use a real-world blockchain-based traceability system, originChain, as a case study to show how to apply design patterns to smart contracts. The architecture design of originChain is briefly discussed in our previous work [4]. This paper focuses more on the structural design of smart contracts, gives more details of several design patterns, and we also share some experiences of applying the patterns to improve the quality attributes of originChain, such as adaptability and interoperability.

The remainder of this paper is organised as follows. Section 2 discusses background information and introduces the related work. Section 3 summarises design patterns for smart contracts and classifies them into four categories. Section 4 presents how to apply those patterns on a blockchain-based traceability application. Section 5 discusses the lessons learned from this case study. Section 6 concludes the paper and outlines future work.

2 Background and Related Work

2.1 Blockchain and Smart Contracts

When Bitcoin was released to public, its capability was limited, providing merely a public ledger to record the transactions related to a specific digital cryptocurrency [8]. Since a programmable infrastructure called *smart contract* has been deployed, the blockchain technology is considered enhanced, as it is enable to deal with more complex transactions, such as triggers, conditions and business logic [9], what users need to do is to authorise their operations via cryptographic signature.

Solidity, a Turing-complete programming language for writing smart contract, is supported by several blockchain platforms that implement Ethereum Virtual Machine[1] such as Ethereum and Parity. *Solidity* is similar to the object-oriented programming languages, a contract in *Solidity* can be considered as a "class" in Java. In addition, *Solidity* also has the mechanisms such as interface, inheritance, and exception, etc. Due to such properties of *Solidity*, it is possible to apply existing design patterns to the *Solidity*-based smart contracts.

2.2 Designing Blockchain-Based Applications

Although blockchain is young, there are a lot of enterprises, institutions, and governments all over the world who are interested in this technology and investigating the applications based on it. Big companies, like Microsoft[2], IBM[3], Amazon[4] provide convenient and instant service of building up a private blockchain. Such works are considered as the combination of cloud service and blockchain technology.

In academia, from the perspective of software architecture, blockchain technology has been considered as a connector to store data in software architecture [10], and the trade-off analysis of choosing blockchain instead of traditional centralised database was discussed in [11]. P. Zhang and his colleagues [12], shared the experience of designing a blockchain-based healthcare platform, to which they applied several software patterns to improve the application scalability.

There are some works on design patterns of smart contract for blockchain-based application. In [2], J. Eberhardt and S. Tai proposed four patterns, including challenge response pattern, off-chain signatures pattern, content-addressable storage pattern, delegated computation pattern, and low contract footprint pattern, which mainly focus on the separation of on-chain and off-chain for data and computation. Bartoletti and Pompianu [1] demonstrated an empirical analysis of smart contracts, in which they collected hundreds of smart contracts and divided them into several categories: token, authorisation, oracle, randomness, poll, time constraint, termination, math and fork check. However, each kind of the smart contracts presented in this analysis has a specifically functional feature, there is a lack of a systematic analysis on smart contracts against architectural properties, which is the focus of this paper. This study focuses more on the architecture design of smart contracts, we emphasise the interoperation among contracts, and present the pseudocode of some patterns.

1 https://solidity.readthedocs.io/en/develop/.
2 https://azure.microsoft.com/en-us/solutions/blockchain/.
3 https://www.ibm.com/blockchain/.
4 https://amazonaws-china.com/cn/partners/blockchain/.

3 Design Patterns of Smart Contract

In this section, we summarise several design patterns [3], and discuss the technical details of each pattern. The patterns are categorised as following: (1) *Creational Pattern* abstracts the process of instantiation, helping developers create smart contracts independently, (2) *Structural Pattern* focuses on the relationship between contracts and their instances, (3) *Inter-Behavioral Pattern* can enhance the flexibility when contract instances need to operate with each other, (4) *Intra-Behavioral pattern* is aimed to improve a particular property, like interoperability or adaptability of the whole system.

3.1 Interaction Among Design Patterns

Figure 1 illustrates a high-level design of applying eight software patterns in a blockchain-based application. Developers can instantiate a contract instance through *Contract Composer*, if the instance contains several objects, and *Contract Factory*, if the instance has a relatively simple structure, and the address of instance is stored in a *Contract Facade* instance for optimal management. Before a contract instance in *Contract Facade* is called, the authenticity of the caller needs to be examined by *Hash Secret* or *Multi-Signature*, depending upon the situation. The caller's identification result is checked by server, if it is valid, the operations will be passed to *Contract Mediator*, which carries out the implementation. When there are new requirements according to the issue of regulation in a certain industry area, *Contract Decorator* helps update a particular contract instance or the whole *Contract Composer* into a new one, by encapsulating the old-version via contract address and appending the new requirements. The new-version contract instances have different addresses, thus *Contract Observer* is needed to update the corresponding contract information in *Contract Facade*.

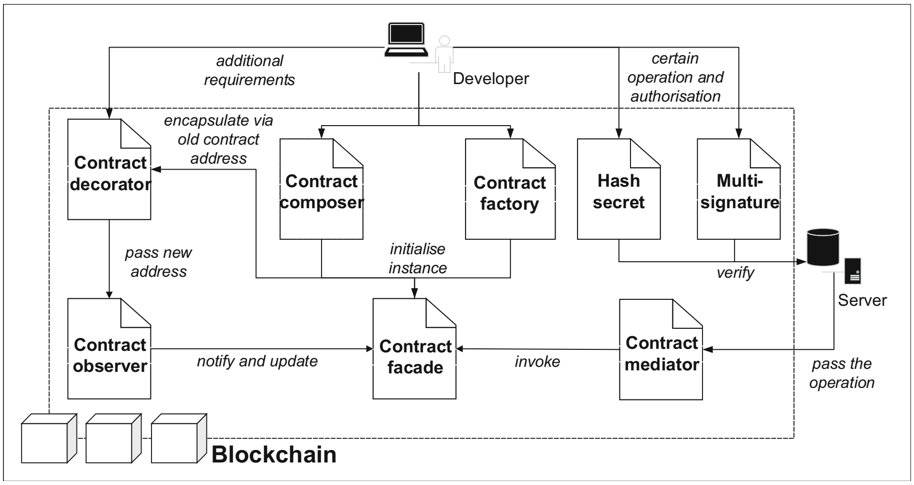


Fig. 1. Structural design of smart contracts.

3.2 Creational Pattern

Contract Factory. As the compiled code of a smart contract deployed on blockchain is not readable, it is tedious to deploy and manage smart contracts that have same properties but aim to diverse clients. With the help of this pattern, developers do not need to deploy the smart contracts one after another, but deploy a contract factory once, through which the required multiple instances can be instantiated.

Contract Composer. In a blockchain-based application, the combination of services or objects is inevitable. Consequently, how to effectively control such a combination becomes a challenge to developers, especially under the condition that each service or object is represented in the form of smart contract. Compared with *Contract Factory*, *Contract Composer* focuses on the complex structure of a contract instance, as it can construct a complicated target through multiple small pieces.

3.3 Structural Pattern

Contract Decorator. Once a smart contract is deployed on blockchain, it is not allowed to modify or update the source code of that contract. *Contract Decorator* pattern can avoid rewriting the whole contract when there are new requirements, developers just need to encapsulate the old contracts and append the required features into a new version of the contract through this pattern, to achieve updatability and modifiability.

Contract Facade. Managing smart contracts may be a burdensome work as there are massive contracts having similar features in a blockchain-based system. *Contract Facade* pattern can relieve such pressure via providing a simple interface by coping with contract addresses. Such an interface is also in the form of smart contract, for developers to call the functions of similar contracts.

3.4 Inter-behavioral Pattern

Contract Mediator. In a business process, smart contracts need to interact with each other to finish a certain activity, which may result in tight coupling of the contracts. *Contract Mediator* pattern aims to reduce the communication complexity of smart contracts, an instance of this pattern is in the form of smart contract, which collects and encapsulates the interactions and invocations from one contract to the others, to decoupling the smart contracts.

Contract Observer. When a smart contract is modified due to the changing requirements in industry, all the related contracts need to be informed and updated automatically. *Contract Observer* pattern can deal with such problem to achieve interoperability and updatability among the contracts via an observer instance. An instance of *Contract Observer* needs to define the objects and information involved, once there are any changes, it should notify all the objects to update information.

3.5 Intra-behavioral Pattern

Intra-Behavioral Patterns do not contribute to the contract architecture as much as the three categories mentioned above, but each one has the ability to work both independently and collaboratively. In this study, *Hash Secret* and *Multi-Signature* are proposed, they have at least one specific property to advance the non-functional requirements of a blockchain-based application respectively.

Hash Secret. This pattern can help a user to achieve authorisation of a particular activity to unknown authorities, by generating a digital secret key known as the hash secret. When the authority is decided, it will then receive the hash secret and thus have the ability to finish further task.

Multi-signature. As there are multiple authorities in a blockchain network, this pattern can provide a flexible way to achieve better cooperation. A transaction is valid only when there are enough signatures from the authorities. In addition, this pattern can also be considered as an individual safeguard mechanism as the current blockchain technology does not provide a way to recover the lost private key.

4 Applying Design Patterns in the Traceability System

In this section, we apply five of the above patterns to a real-world blockchain-based traceability system proposed in [4]: *Contract Composer*, *Contract Facade*, *Contract Observer*, *Hash Secret*, and *Multi-Signature*. *Contract Factory* is commonly used in many blockchain-based applications, thus in this paper we decide not to demonstrate this pattern. *Contract Mediator* and *Contract Decorator* need further refinement, consequently these two patterns will be included in our future work.

4.1 Traceability Systems

Tracking products during production and distribution for the relevant product information (e.g., originality, transportation route and location, and quality certificate) is the main goal of a traceability system. Figure 2 demonstrates the business process of product traceability. A government-certified traceability company can provide traceability services to product suppliers and retailers, by sending staffs to inspect major operation flow, namely to examine factory and freight yard, and contact third-party labs to do sample testing, and if the requirements

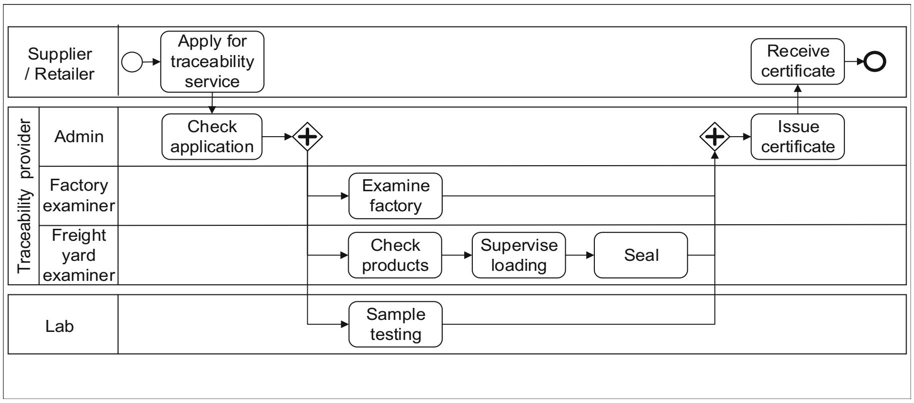


Fig. 2. Process of traceability services.

are met, the traceability company issues inspection certificates which represent the verification of quality and originality of products.

A real-world blockchain-based product traceability system called originChain was proposed in [4], in which the conventional centralised database is combined with decentralised blockchain technology. The supply chain industry requires data transparency and immutability to ensure the reliability of product information, which makes blockchain technology suitable for traceability system.

In originChain, we decided to store sensitive and small data on-chain, such as the hashes of certificates, the on-site freight yard photos and other traceability information like the result of sample testing, for data integrity, while the raw data are stored off-chain in the database. Applying some design patterns of smart contract to originChain can improve the quality of the whole system.

4.2 Contract Composer for Flexibility on Contract Instantiation

A *Contract Composer* instance is demonstrated in Listing 1.1. When a product supplier signs a legal agreement with traceability company, an on-chain version of the agreement should be created. An employee of the traceability company should input the basic information, including the names of the trading parties and the hash of agreement, and some required transaction information, such as price and valid period of each service selected according to the agreement. After inspection, the employee should call *confirm()* function, which can prevent revision to the data stored in this on-chain agreement. In addition, the address of the on-chain agreement should be appended into the actual legal agreement for data integrity.

For a contract that needs to contain many other objects, *Contract Composer* helps to improve flexibility when creating a contract instance. In a traceability process, factory examination, freight yard examination and sample testing are the optional services, and there is a *Service* interface in which a function is defined but not implemented. For each service, there is a contract that implements the *Service* interface, overrides the abstract function, and stores the transaction information. In *LegalAgreement* contract, a service is initialised when the corresponding function is called, and after confirmation, the variable "confirmed" ensures that the data stored in the contract cannot be revised.

4.3 Contract Facade for Connectivity Between Contracts

Every batch of products needs contracts to store the hash of those sensitive traceability service data such as freight yard photos, sample test results, and inspection certificates, which should relate to the legal agreement of the corresponding companies, either. In Listing 1.2, a *FreightYardServiceFacade* contract is generated to connect the on-chain legal agreements generated by *Contract Composer* and the data contracts which store the traceability information. An originChain employee needs to provide batch number and address of on-chain legal agreement when calling function *link()*, in which a new instance of *FreightYardData* contract is created, the on-chain legal agreement address and the *FreightYardData* contract address are connected via the batch number. Similarly, other traceability services have their data contract too. There are accesses for the corresponding staffs of each traceability service process to upload the sensitive data. A freight yard examiner invokes *setFreightYardPic()* function to upload the hash of freight yard photos, while the smart contract records the address of that examiner automatically.

As mentioned in Sect. 3, here *Contract Facade* pattern helps connect two separate smart contracts. There can be more contracts linked by this pattern, nevertheless, connection of multiple contracts may result in a complicated facade contract, thus, developers should reach a balance in a facade contract according to their proximity.

4.4 Contract Observer for Updatability and Interoperability

A legal agreement needs to be revised when there are new regulations released by government. For example, the new Food Safety Law of China regulated new requirements on the formulation of national food safety standards and food safety traceability systems. Furthermore, it can be revised when the corresponding companies reach a new agreement on the traceability services. In either of the two cases, the on-chain legal agreement should be replaced by the new version. A direct way is to issue a new legal agreement contract and replace the old version address with the new one in all related contracts, however, it is tedious and unsafe to do the revision manually. *Contract Observer* pattern can deal with such problem.

Listing 1.3 demonstrates such a situation that the address of an old legal agreement needs to be updated to the new version. *Facade* is an interface, in which there is an abstract function *update()*. *FreightYardServiceFacade*, *FactoryServiceFacade* and *LabServiceFacade* are the contracts that connect legal agreement and the data contracts, they all implement the interface *Facade*.

```
interface Service{
  function setupContract(string sDate, string eDate, uint
      temPrice);
}
contract FactoryExamination is Service{
  string startingDate;
  string endingDate;
  uint price;
  function setupContract(string sDate, string eDate, uint
      temPrice){
    startingDate = sDate;
    endingDate = eDate;
    price = temPrice;}
  function getInfo constant returns (string, string, uint){
    return startingDate, endingDate, price;}
}
contract FreightYardExamination is Service{...}
contract SampleTesting is Service{...}
contract LegalAgreement{
  address firstParty;
  address secondParty;
  bytes32 contractHash;
  FactoryExamination FactoryService;
  FreightYardExamination FreightService;
  SampleTesting LabService;
  bool confirmed;
  function LegalContract(address firstP, address secondP,
      bytes32 cHash){
    if(!confirmed){
      firstParty = firstP;
      secondParty = secondP;
      contractHash = cHash;}
  }
  function setFactoryService(string temStart, string temEnd,
       uint temPrice){
    if(!confirmed){
      FactoryService = new FactoryExamination();
      FactoryService.setupContract(temStart, temEnd,
          temPrice);}
  }
  function setFreightService(string temStart, string temEnd,
       uint temPrice){...}
  function setLabService(string temStart, string temEnd,
      uint temPrice){...}
  function confirm(){
    confirmed = true;
  }
  ...
}
```

Listing 1.1. Contract composer.

LegalAgreementObserver is similar to the reception in a company, receiving information and notifying the corresponding departments. An originChain employee invokes *subscribe()* function to add the three *Facade Contracts* into "subscriber" in advance, and when the new on-chain legal agreement is created, the function *notify()* is called, to inform all the "subscriber" contract to update the legal agreement address.

```
contract LegalAgreement{...}
contract FreightYardData{
  bytes32 [] freightYardPic;
  address [] freightYardExaminer;
  function setFreightYardPic(bytes32 pic, address uploader){
      freightYardPic.push(pic);
      freightYardExaminer.push(uploader);
  }
  function getFreightYardPic(uint i) constant returns (
     bytes32, address){
      return (freightYardPic[i], freightYardExaminer[i]);
  }
  ...
}
interface Facade{...}
contract FreightYardServiceFacade is Facade{
  mapping (string => address) legalAgreement;
  mapping (string => address) traceData;
  mapping (address => string []) batchNo;
  function link(string batchID, address LegalAgreementAddr){
      legalAgreement[batchID] = LegalAgreementAddr;
      traceData[batchID] = new FreightYardData();
      batchNo[LegalContractAddr].push(batchID);
  }
  function setFreightYardPic(string batchID, bytes32 pic){
      FreightYardData(traceData[batchID]).setFreightYardPic(
         pic, msg.sender);
  }
  function getFreightYardPic(string batchID, uint i)
     constant returns (bytes32, address){
      return FreightYardData(traceData[batchID]).
         getFreightYardPic(i);
  }
  ...
}
```

Listing 1.2. Contract facade.

4.5 Hash Secret for Adaptability

A *Hash Secret* contract is shown in Listing 1.4. Every hash secret needs a struct to store hash key, while the boolean variable "init" can prevent the situation that a caller calls function *initial()* twice and then the previous hash key will be revised without permission. The ciphertext of each hash secret should be generated off-chain but verified on-chain, which leads to the difference between the two parameters in function *changeKey()* and *verify()*.

Hash Secret can be used as both on-chain and off-chain component for permission control in a blockchain-based application system. For instance, when a batch of products needs sample testing service, *Hash Secret* can help with the dynamic binding of labs. An originChain employee initiates a hash secret and links it with his/her own address by invoking *initial()* function, then releases the address and plain-text of hash key to the available labs off-chain. Only the authorised labs can acquire "true" from the *verify()* function and upload the result of sample testing. As an on-line component, *Hash Secret* contract should be inherited or contained by another contract, and if it is used as an off-chain component, *Hash Secret* can interact with database or other components by *web3* library[5].

```
contract LegalAgreement{...}
...
interface Facade{
   function update(address oldAddr, address newAddr){};
}
contract FreightYardServiceFacade is Facade{
  mapping (string => address) legalAgreement;
  mapping (address => string []) batchNo;
  function update(address oldAddr, address newAddr){
    for(uint i = 0; i < batchNo[oldAddr].length; i++){
      legalAgreement[batchNo[oldAddr][i]] = newAddr;
      batchNo[newAddr].push(batchNo[oldAddr][i]);
    }
    delete batchNo[oldAddr];
  }
  ...
}
contract FactoryServiceFacade is Facade{...}
contract LabServiceFacade is Facade{...}
contract LegalAgreementObserver{
  address [] subscriber;
  function subscribe(address facadeAddr){
    subscriber.push(facadeAddr);
  }
  function notify(address oldAddr, address newAddr){
    for(uint i = 0; i < subscriber.length; i++)
      Facade(subscriber[i]).update(oldAddr, newAddr);
  }
}
```

Listing 1.3. Contract observer.

[5] https://github.com/ethereum/web3.js.

4.6 Multi-signature for Adaptability

Multi-Signature, similar to *Hash Secret* pattern, can act as both on-chain and off-chain component, Listing 1.5 mimics the multi-signature mechanism in Ethereum. In originChain, a client sends a request of issuing quality certificate, which requires the approval from traceability company, labs, and other related departments, then the result should be decided by these authorities.

The authorisation of *Hash Secret* is dynamic, while the authorities in *Multi-Signature* should be defined in advance. A request can be "agree request", aiming to the external business as issuing certificates, or "update request", aiming to the internal business of the authorities, such as changing authority address, and threshold of accepting a request. Different requests use different thresholds, but share the same pre-defined authority addresses. Each authority invokes the corresponding *Signature()* function to accept the request, and the request result is true if there are enough valid signatures. If so, further operations can be implemented. A requester can call *cancelAgreeRequest()* or *cancelUpdateRequest()* function to withdraw invalid request.

```
contract HashSecret{
  struct hashSecret{
    bytes32 hashKey;
    bool init;
  }
  mapping (address => hashSecret) secret;
  function initial(bytes32 key){
    if(secret[msg.sender].init != true){
      secret[msg.sender].hashKey = key;
      secret[msg.sender].init = true;}
  }
  function changeKey(string oldKey, bytes32 newKey){
    if(secret[msg.sender].init == true)
      if(secret[msg.sender].hashKey == sha256(oldKey))
        secret[msg.sender].hashKey = newKey;
}
  function verify(address initiator, string inputKey)
      constant returns (bool){
    if(secret[initiator].hashKey == sha256(inputKey))
      return true;
    else
      return false;
    }
}
```

Listing 1.4. Hash secret.

5 Discussion

The patterns are divided into four categories in this study to improve the non-functional requirements of blockchain-based applications, and in this section we share some experiences that learn from applying several proposed patterns to a blockchain-based application.

```
contract multiSignature{
  uint total;
  address[] authorities;
  uint agreeThreshold;
  uint updateThreshold;
  address agreeRequester;
  address updateRequester;
  mapping(address => bool) agreeState;
  mapping(address => bool) updateState;
  function multiSignature(uint totalAut, uint aThres, uint
     uThres, address[] aut){...}
  function agreeSignature(){
    agreeState[msg.sender] = true;
  }
  function agreeResult() returns (bool){
    uint k = 0;
    for(uint i = 0; i < total; i++){
      if(agreeState[authorities[i]] == true)
          k++;
    }
    if(k >= agreeThreshold)
      return true;
    else
      return false;
  }
  function initialAgree() internal{...}
  function cancelAgreeRequest(){
    if(msg.sender == agreeRequester)
      initialAgree();
  }
  function updateSignature(){...}
  function updateResult() returns (bool){...}
  function initialUpdate() internal{...}
  function updateAuthorityList(uint newTotal, address[] par)
      {...}
  function updateAgreeThreshold(uint newThres){...}
  function cancelUpdateRequest(){...}
  ...
}
```

Listing 1.5. Multi-signature.

Interoperation Among Smart Contracts. *Creational Pattern*, *Structural Pattern* and *Inter-Behavioral Pattern* focus on the structural design of smart contracts. Contracts are interactive when applying these patterns, for example, in Sect. 4.4, interface *Facade* is implemented by three facade contracts, all of which contain the addresses and invocations of *LegalAgreement*.

LegalAgreementObserver contains the addresses of the three facade contracts, and invokes them via interface *Facade.*

Smart contracts inherit, contain, or invoke each other, which enriches the architectural design, when applied these three kinds of patterns. *Creational Pattern* concentrates on the instantiation of smart contracts, *Structural Pattern* helps manage the relationship among contracts, *Inter-Behavioral Pattern* enhances the flexibility when contract instances operating with each other.

However, the issue of a structured contract includes more complex permission control and cost. Applying such patterns brings the consequence that the architecture of a smart contract may be complicated if it clutters an excess of functions. Moreover, it is hard to manage permission control as a complicated contract may involve multiple roles. Developers need to make reasonable separation when designing smart contracts to balance the coupling degree.

Independence of Intra-behavioral Patterns. Every *Intra-Behavioral Pattern* can work both individually or with other contracts as they do not need to rely on other smart contracts, namely, they do not need to inherit, contain or invoke other contracts but the contrary. Taking *Hash Secret* as an example, it provides a flexible way to implement permission control, either off-chain or on-chain. In Fig. 1, *Hash Secret* verifies a network participant's identity and sends the result to server, only the authorised ones can continue to further operations. Apart from that, smart contracts can inherit or embody this contract to obtain the ability of on-chain permission control.

Reusable Logic Saves Storage. Storage is one of the limitations of blockchain technology. According to blockchain's properties, every participant has a local replica of the whole transaction history, moreover, any revision or deletion to the existing transactions is not allowed, thus joining a blockchain network can be a heavy burden to individual's storage. Design patterns provide smart contracts reusable logic, which helps relieve such burden. For example, developers can change the secret key in *Hash Secret*, and revoke the request or update the authority information in *Multi-Signature* on demand to achieve reusability. Besides, *Contract Decorator* aims to encapsulate the old contracts into the new ones, which can also be considered as a method of saving storage.

6 Conclusion and Future Work

In this paper, we propose a taxonomy of the design patterns for smart contracts, and share the experiences of applying several patterns of *Solidity*-based smart contract to a real-world blockchain-based traceability system called originChain. Blockchain's unique properties provide new thoughts to application architecture, in which blockchain technology acts as a special component. The design patterns affect some specific aspects of the blockchain-based application, such as updatability, adaptability and interoperability.

We divide the design patterns into four categories: *Creational Pattern*, *Structural Pattern*, *Inter-Behavioral Pattern* and *Intra-Behavioral Pattern*, according to their contribution to the architecture design of smart contracts. Specifically, we apply *Contract Composer*, *Contract Facade*, *Contract Observer*, *Hash Secret*, and *Multi-Signature* to a real-world blockchain-based application to improve the non-functional requirements.

Smart contracts used to be standalone and aimed at a specific function, but now in a blockchain-based application, smart contracts can cope with some complex business process instead of barely store the data. Applying some software design patterns to smart contracts helps developers to have a better design on the architecture of smart contracts.

The future work includes summarising more design patterns of smart contract, refining the taxonomy, and giving more detailed discussion about the patterns. In addition, we will optimise the blockchain-based traceability system and extend blockchain technology to other potential industries.

References

1. Bartoletti, M., Pompianu, L.: An empirical analysis of smart contracts: platforms, applications, and design patterns. ArXiv e-prints, March 2017
2. Eberhardt, J., Tai, S.: On or off the blockchain? Insights on off-chaining computation and data. In: De Paoli, F., Schulte, S., Broch Johnsen, E. (eds.) ESOCC 2017. LNCS, vol. 10465, pp. 3–15. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-67262-5_1
3. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design patterns: Elements of Reusable Object-oriented Software. Pearson Education, London (1995)
4. Lu, Q., Xu, X.: Adaptable blockchain-based systems: a case study for product traceability. IEEE Softw. **34**(6), 21–27 (2017)
5. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system (2008)
6. Omohundro, S.: Cryptocurrencies, smart contracts, and artificial intelligence. AI Matters **1**(2), 19–21 (2014). https://doi.org/10.1145/2685328.2685334
7. Staples, M., Chen, S., Falamaki, S., Ponomarev, A., Rimba, P., Weber, A.B.T.I., Xu, X., Zhu, J.: Risks and opportunities for systems using blockchain and smart contracts. Technical report, Sydney (2017). Data61(CSIRO)
8. Swan, M.: Blockchain: Blueprint for a New Economy. O'Reilly, Sebastopol (2015)
9. Weber, I., Xu, X., Riveret, R., Governatori, G., Ponomarev, A., Mendling, J.: Untrusted business process monitoring and execution using blockchain. In: La Rosa, M., Loos, P., Pastor, O. (eds.) BPM 2016. LNCS, vol. 9850, pp. 329–347. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45348-4_19
10. Xu, X., Pautasso, C., Zhu, L., Gramoli, V., Ponomarev, A., Tran, A.B., Chen, S.: The blockchain as a software connector. In: The 13th Working IEEE/IFIP Conference on Software Architecture (WICSA), Venice, Italy (2016)
11. Xu, X., Webber, I., Staples, M., et al.: A taxonomy of blockchain-based systems for architecture design. In: IEEE International Conference on Software Architecture (ICSA), Gothenburg, Sweden (2017)
12. Zhang, P., White, J., Schmidt, D.C., Lenz, G.: Applying Software Patterns to Address Interoperability in Blockchain-based Healthcare Apps. ArXiv e-prints, June 2017

AODV–Based Routing for Payment Channel Networks

Philipp Hoenisch and Ingo Weber(✉)

Data61, CSIRO, Sydney, Australia
philipp@hoenisch.at, ingo.weber@data61.csiro.au

Abstract. Payment Channel Networks such as the Lightning Network (LN), Raiden or COMIT were created to tackle the scalability problems of their underlying blockchains, by moving from expensive and slow on-chain transactions to inexpensive and fast off-chain ones. However, those networks are unregulated and decentralised, comprise point-to-point channels that may be opened or closed without coordination or warning, and fees may change at any time – making routing over these networks a hard problem. In addition, by connecting different blockchains using such off-chain networks, an immense network of channels will evolve which is under continues change. Routing needs to take into account the current network status, availability and distributions of channels' funding, fees for each node, and exchange rates between different currencies. In this work, we identify requirements for such a routing protocol and adapt the Ad-hoc On-Demand Distance Vector Routing (AODV) protocol to this end by enhancing the messages with information on fees and exchanges rates. This approach allows finding suitable routes through the network, while intermediate nodes can maintain their economic incentives. We simulate different network topologies and evaluate the adapted AODV protocol on 3 different networks of 500, 1,000 and 5,000 nodes.

1 Introduction

Ever since the first appearance of Bitcoin in 2008 [16] a multitude of blockchain derivatives and other implementations have emerged. The general purpose is to decentralise the management of a particular asset, such as a cryptocurrency, by removing the need of a trusted central entity and to create a network of untrusted nodes. However, common problems of blockchains include slow confirmation and commit times and high transaction fees. For Bitcoin, a transaction is often regarded as committed (irreversible) after 6 confirmation blocks, taking on average 60 min. Due to this commit delay and high transaction fees (recently between USD 0.50 and 50 on Bitcoin), small payments and particularly micropayments (i.e. payments of a few cents or even a fraction of a cent) are not very economical. In addition, many blockchains have a throughput (7 to 20 transactions per second, tps) that is many orders of magnitude below that of payment networks like VISA (up to 47,000 tps).

In order to tackle this problem, researchers proposed to not settle every transaction on the chain but rather move some transactions off-chain, allowing two parties to interact with each other directly. By tracking their payments between each other on their own, the two parties are able to avoid expensive and time-consuming interactions with the blockchain. Should there be a dispute regarding the balance or should one party become unresponsive, the most recent balance sheet provided by either of the two parties can be settled on the blockchain (on-chain). The Lightning Network (LN) is the most prominent of these off-chain solutions [20]. It proposes to create an overlay network of off-chain payment channels – i.e. Payment Channel Network (PCN) – where transaction between two parties are not recorded on the Bitcoin blockchain. For that, two parties create a pair of transactions first: a *funding transaction* and a *spending transaction*. The former one specifies the total amount held in the channel, i.e. either one or both of the participants pay an arbitrary amount into this channel. The other transaction specifies the output, i.e. it defines which party receives how much from the total amount. Only the funding transaction is settled directly on the blockchain whereas the output transaction can be delayed to a point of time in the future. If one of the party wants to pay the other party, it updates the output transaction mirroring the actual state. Note, the actual output transaction involves a complex reconciliation between the two parties, its process is described in detail in [20]. Decker et al. in [4] presented an alternative. While the LN was specifically designed for Bitcoin, PCNs can be realised on other blockchains as well if they provide a minimal scripting language that allows realising so called Hash-time Lock Contract (HTLC). Alternatives include the Raiden Network (Ethereum) [22], Sprites (Bitcoin) [12] or COMIT (cross-chain) [8].

Opening a channel only makes sense for recurring payments to (or through) the respective other party. To transact with parties where no direct channel exists, multiple payment channels can be chained together. The payment is then routed through the network with one or more intermediaries. This however gives rise to a major challenge: how to find an optimal (or acceptable) route through the network, i.e., from the sender of a payment to the recipient. The route needs to be acceptable (and ideally optimized) in terms of to specific criteria such as routing fees, exchange rates, and reliability. The routing within the LN applies a proactive routing protocol: each node broadcasts the information about its neighbours (e.g. the nodes which itself is connected to via a channel) through the network. Hence, the downside of this is that a lot of information needs to be send around prior a route can be established. Consequently, each node has complete knowledge about the network topology. So far, only information about the channel funding is included, but not the funding distribution, i.e. how much of the funding is currently on which side and the solution is limited to the Bitcoin blockchain. Obviously broadcasting global topology information is costly and introduces its own scalability limits, which are particularly severe when considering a cross-chain network.

We argue that, without a fully automated solution for payment routing, with localized routing and the ability to adapt to the ever-changing environment of such a network, PCNs cannot realize their true potential or achieve meaningful coverage on a global scale. Hence, we make the following contributions:

– We present an adaptation of an Ad-hoc On-demand Distance Vector (AODV)-based routing algorithm for a network of off-chain payment channels.
– Our approach can cater for different currencies, hence allowing to route payments across multiple blockchains.
– We evaluate the applicability of our routing protocol experimentally and discuss advantages and disadvantages.

In the next section, we discuss related work in the field of (payment) routing. Afterwards we formulate the requirements of the routing protocol in Sect. 3 followed by the protocol in Sect. 4 and its evaluation in Sect. 5. We discuss the results in Sect. 6 and conclude in Sect. 7.

2 Related Work

Routing can be described as the technique of "[...] sending a unit of information from point A to point B by determining a path through the network, and by doing so efficiently and quickly [...]" [11, Chap. 1, p. 3]. This topic has received a lot of attention in research and industry resulting in various different routing algorithms as routing is fundamental to power almost any small to large network efficiently. Examples range from Circular Switched Telephone Networks (PSTN) over Mobile Ad Hoc Network (MANET) to packet routing on the Internet. However, routing on payment channels has hardly been examined. Hence, we expand our search into the other fields. Especial the area of MANET networks is relevant, as these show similar characteristics as PCNs: Nodes may appear and disappear irregularly, be offline for a while (e.g. as do connections do in MANET) or the channel balances might change frequently. Routing protocols can be classified into five major types: reactive, proactive, hybrid (i.e. a combination out of both: reactive and proactive), hierarchical and coordinate-based.

Reactive protocols perform route discovery on-demand. They do so by *flooding* the network with route discovery requests. Two famous examples are Ad-hoc On-demand Distance Vector (AODV), Dynamic Source Routing (DSR) [9,19]. They work best in a highly dynamic environment (in cases where the network topology changes quickly). In contrast to that proactive routing protocols work well in static scenarios (whenever an update occurred, information is spread). In most cases, each single node maintains a routing table and can decide on the route on its own. Examples of such routing protocols are Destination-Sequenced Distance Vector routing (DSDV) and Wireless Routing Protocol (WRP) [15]. The offsite of re- and pro-active protocols is that excessive flooding can lead to network clogging. Hence, a combination of both achieves a better performance across a wide range of scenarios. Hybrid routing protocols such as Zone Routing Protocol (ZRP) or Enhanced Interior Gateway Routing Protocol (EIGRP) in which each node maintains a routing table on the routes inside its zone, for destinations outside the zones, a route discovery procedure is employed [1,7]. In contrast to these kind of algorithms, hierarchical and coordinated-based protocols rather use location-based algorithms than flooding the network with messages. Two examples of hierarchical routing protocols are LANMAR [18], L+[14] and two coordination-based protocols are GPSR [10] and BVR [5]. These kinds of algorithms use location-dependent addresses to route information.

These routing protocols were mostly designed for MANET, however, PCN may differ, e.g., additional hops may increase the overall expanses, a *route response* changes the networks state as funds will need to be locked or a found route might be only usable up to a certain amount of times as the involved channels might run out of funding. Nevertheless, routing in off-chain channel networks can benefit from ideas of MANET routing protocols.

An example for hybrid routing is the protocol Flare which is meant to replace the current DSR-based routing protocol employed in the LN [21]. Nodes proactively gather information about the network topology from neighbour nodes (as in DSR) and from beacons which are close (in the sense of Bitcoin addresses). Hence, a sender can decide on the route and issue the payment. Each node broadcasts in a regular interval (or when a change occurred) its local information to their neighbours. If a node receives an update message it first updates its local routing table and enhances the information in the received message with its local information and forwards it to its neighbours. Messages can be encrypted using onion routing [23]. The Ripple Network integrates a path-finding algorithm called *ripple paths* [24]. Payments can be *rippled* through several nodes. This involves moving debt around. Cross currency payments are possible through so called *order books*. However, how exactly a path is found is not clearly defined.

Flare is closest to our work. However, the fundamental difference to our assumptions is that while Flare focuses primarily on security and censorship resistance for the sending node, we focus on the autonomy of each single intermediate node. Each node should not be forced to forward a payment into a certain direction. This decision is driven from the economical point of view, as we assume, nodes primarily focus on profit maximisation and hence are more likely to select routes which ensure high profit. Notably, we covered mostly abstract algorithm of each category within this section. There are various adaptations focusing on more concrete problems, e.g. an improved AODV protocol against 'black hole' attacks [13], reducing the message overhead of AODV using availability prediction [2], adaptive multipath source routing for DSR [28] or an anonymous DSR protocol or AODV routing [26,30]. We argue that if the most abstract protocol is suitable, the improved version might lead to even better results.

3 Requirements and Algorithm Selection

In order to find a suitable routing algorithm for transferring values (in the form of cryptocurrencies) from a sender to a receiver via one or more intermediate nodes in an inexpensive and reliable way we define the requirements for the routing protocol similar to the ones in Flare in [21]. These requirements were formed from community beliefs [17]. Hence, we derived the following requirements which are imposed directly in the LN:

1. **Autonomy and self-reliance:** In order to provide high availability and a failure resistant network, the nodes need to be self-configurable: each node should be able to act autonomously and independently. Hence, a node should be able to act as a sender or recipient at the same time and should be able to route payment requests in any direction. In addition, the functionality of the network must be preserved despite of random changes in the network's topology or due Byzantine behaviour of some nodes.
2. **Cost guaranties:** Each node in this network may charge a certain fee to forward a payment. In addition, when crossing different blockchains, the bridging node will ask for a specific exchange rate between two currencies. Hence, it is essential that the overall cost to issue a payment across the network from a sender to the final recipient is known prior to its execution. This is required, as the sender wants to ensure that the desired amount arrives at the final recipient and is not eaten up by fees or exchange rates.
3. **Time-lock guaranties:** It is required that each payment (channel update) is assigned with a time-lock (compare HTLC [20]). This serves two purposes, first, the receiver needs enough time to redeem the payment, and second, the sender needs to have enough time to rollback in case a failure occurred.
4. **Flexibility:** The routing protocol needs to be flexible enough to take frequent changes into account. Changes in a huge network are likely to happen in various aspects: channels may appear or disappear, the channel's funding distribution may change, nodes may update their fees or nodes between blockchains may change the rates in order to not lose money.
5. **Prevent network partitioning:** A single (or several) failing nodes should not forestall payment routing or split the network in sub-networks. A node *should* always be able to find a route to a desired opponent, i.e. no other node should be able to prevent a payment. In other words: if a route exists between two nodes, the routing protocol should be able to find it.
6. **Real-time:** A major goal of PCNs is to enable instant micropayments. Hence, it is natural that the routing protocol needs to be very fast. Hence, network traffic delays are the only timely constraints which are allowed, i.e. the routing should take less than a few seconds.
7. **Up-to-dateness:** Having up to date information available is crucial for finding the best route through the network. Hence, a requirement is that a found route contains up to date information about fees and exchange rates.
8. **Lightweight and scalable:** It is expected that the off-chain channel network will grow over time. Hence, routing should be able to adapt and scale with it. In addition, routing should only use a moderate amount of resources.
9. **Trustlessness:** Routing should withstand when nodes show Byzantine behaviour (i.e. are lying about fees or routes).
10. **Optimal solution for each node:** The routing protocol should allow each node to act within its own economic incentives. These will differ from node to node, e.g. a node issuing a payment may want to send the payment along the cheapest, fastest or the path with the highest success rate. Contrary, intermediate or forwarding nodes are driven by different incentives. For example, some may provide the cheapest route or aim for high reliability.

Hence, intermediate nodes may not forward routes if this compromises their reputation.

Algorithm Selection – AODV. We have chosen the above mentioned 10 requirements in order to have a reliable and usable network. It should be noted that especially the last point is from highest priority as we want to guarantee each participant's economic incentives. The point lead to the routing protocol selection of AODV. AODV is a routing protocol originally designed for MANET and other wireless ad hoc networks. Key features are the ability to quickly adapt to dynamic condition changes, a low processing and memory overhead and loop freedom at all times [19]. As the name implies, a route is established only on demand on a hop-by-hop basis. Each node should only have one option to send the message through the network (notable, a sending node which is connected to multiple neighbours might end up with multiple routes) as each intermediate node can decide how to forward the message to the final receiver.

One of our main assumption about the network is that it is under continue change. Even more, we assume that network changes are more likely to happen than nodes are sending payments around. Examples for frequent changes are (I) Channel balances change frequently, i.e. a route which was valid before a payment must not necessarily be valid after that payment. (II) Fees may change frequently. Each intermediate node can charge an arbitrary amount of fee when forwarding a payment. This node may regularly update that fee in order to keep its channels balanced. For example, if one channel is at risk of running out of liquidity, the node could charge a higher fee when forwarding payments through that channel. (III) Exchange rates may change quickly. Nodes between two blockchains will need to adapt the exchange rate frequently. Otherwise this node may be at risk of losing money. (IV) Nodes may be offline (or not reachable) for some time. While it is fundamental that nodes are online at all time during a payment as this node needs to accept and forward payment request, if an intermediate node is offline (even for a short amount of time), the network should adapt accordingly.

An alternative would have been a proactive routing algorithm similar to Flare which is based on source routing [21]. In contrast to AODV, source routing can be easily combined with Onion Routing and brings a higher level of security to the network participants. It allows to encrypt the payment message in different layers so that no intermediate node in between is aware of what the message's final destination will be or who the original sender was. However, the downside of this is that a sending node can indirectly attack an arbitrary node in the middle, e.g. by draining its liquidity (locking it up) for some time so that it cannot forward any more payments. Even worse, if the message is encrypted, the attacked node will not even know who it gets attacked from in the first place. A more detailed discussion of this attack can be found in Sect. 6. The other main difference between a reactive and a proactive approach is that reactive routing eliminates the need of periodically flooding the network with table update messages. However, the disadvantage of this approach is that a route request messages may flood the network, i.e. a large amount of messages may be sent around until a suitable route has been found. We have decided for a reactive routing algorithm as it has the abilities of obtaining an up to date route, being loop free and a quick adaptation to the ever-changing network conditions. In addition, as each node can decide on its own how and where to route a payment to, it is easier to maintain its economic incentives.

4 AODV-Based Routing in PCN

System Model. Before defining to the routing protocol in pseudo code in Sects. 4.1 and 4.2 we define our PCN as an undirected graph $G = (N, C)$ whereas N is the set of all nodes and C is the set of channels between the nodes $C \subseteq \{(n_1, n_2) \mid n_1, n_2 \in N\}$. Each node n represents an independent party who wants to participate in the network either by playing an active role as payer or payee, or passively by earning money as an intermediate node which forwards payments. Each node is assigned with a globally unique id. Further, for $\forall c \in C$ we define $bal(n, c)$ as a the current balance of node n in channel c as a binary function $Bal : N \times C \to [0, +\infty)$. If a node n is not part of the channel the balance function is not defined, i.e. $\forall n \in N,\ c = (n_1, n_2) \in C,\ n \notin \{n_1, n_2\} \iff \uparrow bal(n, c)$ (read as $bal(n, c)$ is not defined). When forwarding a payment over a channel c an intermediate node n may charge a fee $f \in (-\infty, +\infty)$ and provide a rate $r \in [0, +\infty)$. This means, the cost to send an amount x via a node n_i is $p_i = r_i \times x + f_i$, i.e. the sending node has to pay p_i so that x arrives at the next node while the intermediate node will deduct f_i for itself. Hence, the total cost p (for payment) to send a payment from a starting node n_1 over the intermediate nodes n_2 and n_3 to node n_4 can be expressed as $p = p_3(p_2(x))$ or $p = r_3 \times (r_2 \times x + f_2) + f_3$. Notably, if an intermediate node is in between two blockchains, the arriving amount x is in a different currency. Note that nodes are deliberately enabled to offer a negative fee for forwarding a request. By providing this incentive, i.e. making routes through them cheaper compared to potential other available routes, the forwarding node is able to rebalance its channels in a specific way.

The payment process consists of two phases: first, the *route discovery* phase (Sect. 4.1) and second, the *route selection* phase (Sect. 4.2).

4.1 Route Discovery

As designed in the RFC of AODV a route discovery is only issued when a node decides to send a payment over the network: a route request is broadcast to each connected node, i.e. nodes which are connected via a payment channel: $REQ = <n_o, n_d, n_l, nr_d, nr_o, d_{req}, d_{route}, \#_{max}, \#_{rate}, \#_{fee}, \#_{hops}>$. The route request is defined as REQ whereas n_o defines the route request originator. This is needed so that each node can update its local routing table with a path towards n_o. n_d is the desired recipient and n_l is the last hop, i.e. the node this request was send from. nr_d and nr_o define a sequence number. The former one is the latest sequence number received in the past by the originator n_o for any route towards the destination n_d. The latter one is the sequence number to be used in the route towards the originator n_o of the route request. d_{req} defines the lifetime of the request, i.e. how long this request is valid. These fields can be found in the original AODV as well. $\#_{hops}$ defines the amount of intermediate nodes (or hops) to the request originator n_o. However, in order to have a more sophisticated algorithm we enhanced the REQ with additional fields. Whenever a node sends out a route request, we use this chance to establish a route backwards (i.e. towards the originator) on each node which receives the request. Hence, we added 3 more fields: d_{route} defines how long the route towards n_o is valid, $\#_{rate}$ is the rate to reach this node and $\#_{fee}$ represents the cost to reach this node.

As by definition of the channel network, each channel has a certain capacity, hence each node has a certain maximum it can pay over a specific channel ($bal(n, c)$). While this is the overall maximum, a specific node may only be willing to forward a fraction of it, i.e. $\#_{max}$. Notably, $\#_{max} \leq bal(n_i, c)$ where i is the current node forwarding this request over channel c. This value varies depending on the request. Further, if a node receives a REQ and is not the destination node, it first updates the REQ before forwarding it. The fields n_o and n_d remain unchanged, n_l is set to the current node. The rate $\#_{rate}$ is updated with its local rate times the former rate and so is the new fee the sum of the local fee and the former fee. In order to compute the new $\#_{max}$ the current node checks how much it can send over the channel to n_l ($\#_{maxOld}$) and takes the minimum of those two values, i.e. $\#_{maxNew} = Min(\#_{maxOld}, \#_{maxCur})$. The same applies for the expiry time d_{route}, i.e. the current node updates this field with the minimum between the last d_{route} and the time it is willing to offer this route to n_o. Hence, using REQ each receiving node receives the information of how much it can send towards the originator n_o ($\#_{maxNew}$), for how long the route is valid d_{route}. In terms of units, the expiry date is expressed using the unix timestamp format. The rate and the fee are floating-point numbers in order to be able to represent the smallest unit of an arbitrary currency, e.g. 1 Satoshi.

Algorithm 1 shows a pseudo code of how a REQ is handled if received by a node. Lines 1 and 2 perform validation steps. First, the local node checks the *req* is still valid, i.e. if *req*.d_{req} has not been expired and the max amount of hops has not been reached ($\#_{hops} \leq MAX\_HOPS$), i.e. if the REQ is dropped if it has traversed more than MAX_HOPS intermediate nodes. Afterwards, it checks if a valid channel exists to the last hop and if this channel has enough balance. Thereupon, the local node checks in its routing table if a route towards n_o is already present (line 3). If this is the case, the node checks whether the current *req* is *better* than the routing table entry. Within this check, the local node can follow its own economic incentives, e.g. take a lower rate and fee, a longer time expiry date, etc. The new routing table entry is created in line 5. In the next line 7, it is checked if this *req* has already been handled before, i.e. for this case the *req*.nr_d is taken. If so, no further actions are performed. In line 8 it is checked whether the local node is already the final recipient. If this is the case, a REP is returned to the last hop n_l (see Sect. 4.2). Similar, in line 11 the local node checks if a route is already known to the destination node n_d.

Algorithm 1. handleRouteReq(*req*)

```
Input: req: Route Request data
1  if !isValid(req) then return;
2  if !channelToExists(req.$n_l$) OR bal(req.$n_l$,c) $\leq$ 0 then return;
3  $t_o$ = getRoutingTableTo(req.$n_o$);
4  if $t_o$ != null AND req is better than routing table entry then
5      $t_o$ = updateTable(req.$n_o$, req);
6  end
7  if reqAlreadyHandled(req.$nr_d$) then return;
8  if this == req.$n_d$ then
9      sendRouteRep(req.$n_l$); //node is destination, send rep to requester;
10 end
11 $t_d$ = getRoutingTableTo(req.$n_d$);
12 if $t_d$ != null then
13     sendRouteRep($t_o$.$n_n$); //take from routing table and send response;
14 end
15 req.$\#_{rate}$ *= this.$\#_{rate}$; req.$\#_{fee}$ += this.$\#_{fee}$; req.$\#_{max}$ = Min(req.$\#_{max}$,
     this.$\#_{max}$); req.$n_l$ = this; req.$\#_{hops}$++;
16 lockFunding(req.$n_l$, req.$\#_{max}$, req.$d_{route}$);
17 for c in outgoingChannels do
18     sendReq(req, c.counterNode);
19 end
```

If so, a *REP* is returned to the next hop in the routing table entry. If the local node is an intermediate node the *REQ* will be forwarded to all nodes to which the local node has a channel to. Hence, starting from line 15, the *REQ* will be updated, i.e. the new rate and fee to the next node is calculated and the last hop n_l is set to the current node. Also, the hop count ($req.\#_{hops}$) is updated and incremented by 1. Since within the *REQ* the node also promises a route towards n_o, the local node has to lock up some funds for some time (line 16). This is needed, in order to ensure enough funding is available for a payment over this route. Notably, as it is required to lock funds up, a node can decide on its own whether it will forward the *REQ* or not. So, at any time, a local node can decide not to forward a *REQ* at all, this is however not depicted in the algorithm.

4.2 Route Selection

As soon the *REQ* has reached its destination node n_d, or if a route towards the destination was already known by an intermediate node, the route responds message (*REP*) is returned towards the originator. *REP* has a similar format as *REQ*: $REP = <n_o, n_d, n_l, nr_d, nr_o, d_{route}, \#_{max}, \#_{rate}, \#_{fee}>$.

The field n_o defines the originator of the message, i.e. it is the destination node n_d of the *REQ* message. Similar to that, n_d is the destination of *REP* (or the originator n_o of *REQ*). n_l follows the same principle, i.e. it is always set to the last hop the message was send from. nr_d and nr_o define sequence numbers.

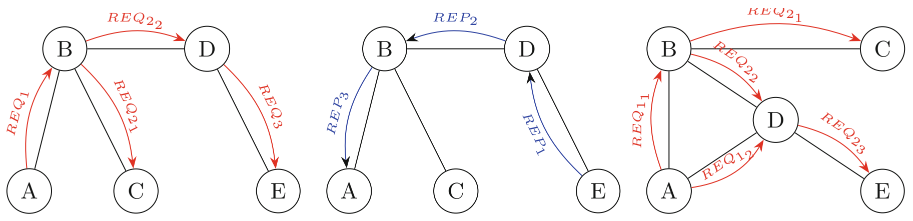
Algorithm 2. handleResponse(*rep*)

```
   Input: $rep$: Route Response
 1 if $!isValid(rep)$ then return;
 2 if $!channelToExists(rep.n_l)$ || $!channelIsFunded(rep.n_l)$ then return;
 3 $t_o$ = getRoutingTableTo($rep.n_o$);
 4 if $t_o == null$ || $rep$ is better than $t_o$ then
 5     updateTable($rep.n_o$, $rep$); //new rep is better, update table;
 6 else
 7     return; //known route is better;
 8 end
 9 if this == $rep.n_d$ then
10     return; //local node was original requester
11 end
12 $t_d$ = getRoutingTableTo($rep.n_d$);
13 if $t_d == null$ then
14     return;//error, no route to origin found
15 end
16 $rep.\#_{rate}$ *= $this.\#_{rate}$; $rep.\#_{fee}$ += $this.\#_{fee}$; $rep.\#_{max}$ = Min($rep.\#_{max}$,
     $this.\#_{max}$); $rep.\#_{hops}$++; $rep.n_l$ = this;
17 lockFunding($t_o.nextHop$, $rep.\#_{max}$, $rep.d_{route}$);
18 sendRep($rep$, $t_d.nextHop$);
```

The former one is the latest sequence number of the route's destination node and the latter one is the sequence number to be used in the route towards the originator n_o of the route request. d_{route} defines how long the route towards n_o is valid, $\#_{rate}$ is the rate to reach this node and $\#_{fee}$ represents the fee.

Algorithm 2 shows the pseudo code of how a *REP* message is handled: similar to handling the route request message, in line 1 and 2 the *REP* is verified to be valid. In this case, the field d_{route} is checked, i.e. if the route promised in *REP* has not yet expired. If this is the case it is checked if enough funding is available in the channel towards the last hop. In line 3 the local routing table is checked whether a route is already present to the node $rep.n_o$ and if the known route is *better* than the new route in *REP*. Again, the local node can follow its own economic incentives and accept only routes which are suitable. If the new route is better, or no routing table entry exists, the routing table is updated with a new entry towards the destination node. On the contrary, if a route already exists, the process ends here (line 7). In line 9 it is verified if the local node is the destination, i.e. the *REP* has reached the original requester of this route request. If this is the case, the process ends here and the local node can issue a payment along this route. Alternatively, the local node is an intermediate node and is meant to forward the *REP*. For that, it checks in the routing table if a route is known towards n_d (see line 12). If no route is found, the process ends here as it is not possible to forward the *REP*. Consequently, if the route towards n_d is known, the *REP* message is updated. Starting from line 16 to 16 the rate ($rep.\#_{rate}$), fees ($rep.\#_{fee}$), the max amount for this route ($rep.\#_{max}$), the last hop ($rep.n_l$) and the hop count ($rep.\#_{hops}$) is updated. Afterwards, the node locks the max amount of the promised route ($rep.\#_{max}$) in line 17 and forwards the route to the next node according to the routing table entry ($t_d.nextHop$) in line 18. Similar to the route discovery phase, the most important part is the locking of the funds in a specific channel for the time the route is valid ($rep.d_{route}$) as can be seen in line 17. Since this information is only kept offline in a local node, it only represent a *promise* that the funds are available but it does not give a 100% guarantee that the route is valid until $rep.d_{route}$ expired.

Figure 1a show the process of how a route from node A to node E is established. For that, node A sends a route request (REQ_1) to its connected neighbours, i.e. B. In turn, B forwards this request to C and D (REQ_{2_1}, REQ_{2_2}) and so on. Eventually, the REQ arrives at node E which returns a REP message towards the originator A. Notable, since each REQ contains the information of how to reach the originator node, every node in this example has now the information of how to route towards A. The path the REP message follows is comparably simpler. As it can be seen in Fig. 1b, node E issues the first REP_1 message along the path towards A. Hence, the message first passes node D (REP_1), then node B (REP_2) and eventually arrives back at node A (REP_3). If node C now wants to pay A it can do so immediately as it has already all the information. If C wants to pay E, it will only need to issue one REQ towards B. B knows already the required information and returns REP immediately. Hence, the more active the overall network is, the less messages will be needed to find a route. After having explained the algorithm, we will evaluate it in Sect. 5.



(a) REQ phase (b) REP phase (c) Ig. redundant messages

Fig. 1. REQ and REP phase, and ignoring redundant messages

5 Evaluation

In order to understand if the adapted AODV routing protocol (i.e. by enhancing the messages with information on fees and exchanges rates) is applicable for payment routing in PCNs we evaluated it in a simulated environment. As at the time of writing the LN just went live on the Bitcoin main net[1], no real data about how the network will look like in big scale was available. Hence, we have to come up with some assumptions in regard to the network topology in Sect. 5. The results of the evaluation are discussed in Sect. 6.

[1] https://lnmainnet.gaben.win/.

Setup. We evaluate our routing protocol on 3 different network topologies, with 500 nodes, 1,000 nodes, 5,000 nodes. In regards of the network, we divide the nodes among 3 different blockchains, e.g. BTC, ETH and LTC. Each node is placed at random in one of these 3 blockchains with a uniform probability of $p = 0.3$. In the next step we take the smallest blockchain (i.e. the one with the least amount of nodes $n_{\#}$) and select randomly between 1 and $|n_{\#}|$ nodes. These nodes are Liquidity Providers between two blockchains, i.e. they have a wallet on both blockchains enabling routing between them. We repeat the same procedure and connect the smallest chain with the second chain. The same is repeated between the remaining two blockchains. In the next step we randomly connect the nodes within each chain according the *dynamics of small-world* networks as presented by Watts and Strogatz [29]. We chose $k = 4$ as the average connection between each node within each chain and a probability of rewiring of $p = 0.3$. To generate the graph, we used the WattsStrogatzGenerator (http://graphstream-project.org). We decided to have the funding on each channel side randomly generated with a value $f = [1, 100]$. We ignored the different currencies, meaning that the overall maximum a channel can have is $MAX(bal(c)) = 200$ BTC (100 BTC on each side of the channel) and the minimum is $MIN(bal(c)) = 2$ LTC (1 LTC on each side). As exchange rates between BTC:LTC, LTC:ETH and BTC:ETH we took fix values as monitored from http://coinmarketcap.com (2017/09/25, 3:25 pm UTC+10): 1 ETH = 0.07508560 BTC, 1 LTC = 0.01262120 BTC and 1 LTC = 0.16506884 ETH. Each Liquidity Provider offers the same exchange rate; however, each node randomly charges a fee when forwarding payments ($rep.\#_{fee}$ or $req.\#_{fee}$). This fee is randomly generated only once per node and is between $*.\#_{fee} = [0, 1] * 10^{-9}$. The details of generated topologies can be found in Table 1. Notably, since channels are bidirectional a connection between two nodes counts as one channel.

Table 1. Evaluation settings

Nodes	500			1000			5000		
Channels	1098			2367			10689		
	BTC	ETH	LTC	BTC	ETH	LTC	BTC	ETH	LTC
Nodes	149	147	204	285	306	409	1519	1496	1985
Channels	298	294	408	570	612	818	3038	2992	397
BTC-ETH channels	13			257			408		
BTC-LTC channels	76			103			193		
ETH-LTC channels	9			7			88		

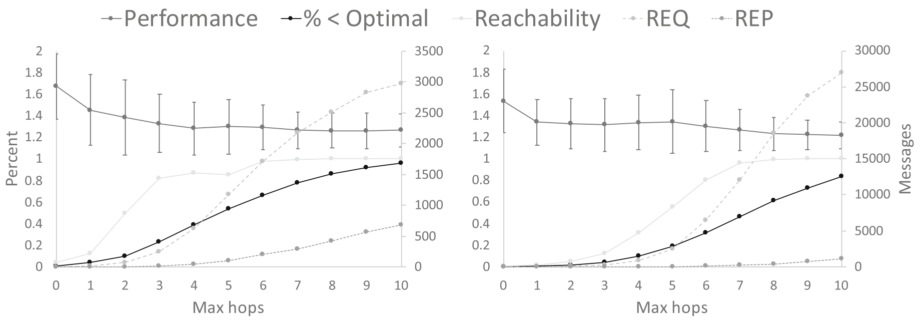
Scenario & Evaluation Criteria. In order to verify the quality of our approach we run 1,000 randomly generated transactions on each network topology, i.e. we randomly select one node which pays another random node. The payment amount is randomly generated with $\# = [0, 1] * 10^{-9}$. Notable, while in theory it is possible to have even smaller values as payments are not recorded on the blockchain immediately and hence are not limited to it, this would require a different setup. This means, if a payment is 1/10 of a satoshi, we would need 10 of these payments in order to have a noticeable change, hence, we ignored this factor. The limiting factor of the AODV routing protocol is the hop count (MAX_HOPS), i.e. if the REQ has traversed more than MAX_HOPS intermediate nodes, the request is dropped. Hence, we run each transaction with different MAX_HOP, i.e. with an $MAX\_HOP = [0, N]$ where N is the cheapest possible path but capped with 10. Differently expressed, if two nodes are connected directly the hop count is 0 and there is one channel between them. If there is an additional node in between, the hop count 1 and there are two channels between them. Hence, we have a maximum of 10 hops or a maximum of 11 channels. The cheapest possible path was computed manually using the Floyd-Warshall all pair shortest path algorithm [3, pp. 558–565]. As a single route request may change the networks topology we reset the nodes and the channel states before each new request. We compare the found routes with the optimal path, i.e. the cheapest overall path for each hop count. This means that a lower hop count could lead to a more expensive route than the optimal. In addition, we count the overall messages which were send around and measure the reachability, i.e. how many transactions where successfully depending on the hop count.

6 Discussion

The results of our evaluation can be found in Fig. 2 (not showing the 1,000 nodes scenario due to space constraints). As mentioned above in Sect. 5 we have 3 different network topologies with 500, 1,000 and 5,000 nodes. On each topology we ran 1,000 randomly selected transactions, i.e. the payer and payee were selected randomly. Each transaction was run 0 to N times where N is capped with 10 or with the shortest optimal route. Hence, the higher the HOPS get, the less transactions are executed. Two nodes are separated by 1 hop if there is a single intermediate node in between or by 0 hops if they are connected directly. $\overline{P}$ is the arithmetic mean of the performance P. It indicates how much more expensive the found paths were on average compared to the optimal shortest path (The optimal shortest path was calculated using Floyd-Warshall algorithm.), e.g. an $\overline{P}$ of 1.53 means that on average the found path was 1.53 times more expensive than the optimal path. In addition, $\sigma_{\overline{P}}$ states its standard deviation. For simplicity reasons we normalised the number of send REQ and REP messages by dividing it through the amount of transaction ($\#_{TX}$).

As the figures clearly show, the performance $\overline{P}$ goes towards 1 slowly the higher the max allowed of hops get (MAX_HOP). The reason why we do not hit 1 at 10 already is twofold. First, we allowed only a max amount of 10 hops although there were a few transactions with an optimal route with more than 10 hops, i.e. for 500 nodes 3.9%, for 1,000 nodes 1.6% and for 5,000 nodes 16.3% needed more than 10 hops to find an optimal solution. Besides, the standard deviation of the $\overline{P}$ decreases with the number of hops. Hence, we can say that our routing finds more results closer to the actual optimal route. The chance of being able to find the optimal route depending on the hops is expressed by $\% <$ *Optimal*. As it can be seen, the higher the hop count is the higher the chance to finding the optimal solution the closer we get to it. The second reason why this number approximates slowly to 1, is that there is a chance that the optimal route is never found, although the maximum allowed hops would allow for it. This can be reduced to the fact that we try to keep the amount of unnecessary messages low. For example, each node handles each *REQ* and *REP* only once unless its information is fresher or *better*. Figure 1c shows a route request (transaction) from A to E. In the first phase, A sends a request to B and D (REQ_{1_1} and REQ_{1_2}). As soon as received, these nodes forward their messages REQ_{2_1} and REQ_{2_2} from B to C and D and REQ_{2_3} from D to E. As D has received a *REQ* for a payment from A to E prior from A, it ignores the message from B. This could be a limiting factor if the route from A to D via B is cheaper than A to D. However, we made this decision knowingly in order to reduce the amount of *REQ* which are send around, as otherwise the network would get flooded completely. In addition, in order to have a higher probability of finding a better route, we allowed for the following. If the *REQ* from B arrives at D when D has already found a route to E it returns indeed a *REP* message. Hence, in this example, A would receive two *REP* messages from which it can chose the more suitable one. This fact explains why our results show a relatively high amount of *REP*. The figures show the total amount of *REQ* and *REP* messages in the network normalised by the amount of sent transactions. It is obvious that the amount increases almost exponential with the amount of allowed hops. The problem lays in the information each node is acting on, i.e. while nodes to ignore already handled *REQ* messages it cannot know whether a node has already received this message from a different node or not. A simple solution would be to add information to the *REQ*/*REP* message *where* it has already been. However, while this would reduce the amount of messages, it would increase the size of it. Interestingly, while for 500 nodes and 1,000 nodes we were able to achieve a reachability of over 99% for 5,000 nodes we were able to achieve a similar number only with 7 hops. This lead to a dramatic increase of *REQ* messages which were send around, i.e. while we had a reachability of 80% and 6,405 *REQ* messages for 6 hops we had a reachability of 98% and 12,097 *REQ* messages for 7 hops. Hence, we would recommend to have a dynamic value for the maximum allowed hops, i.e. if the network grows the maximum hop count should increase. Notably, the amount of *REQ* is decreasing over time heavily as nodes cache route information for some time. In the example in Fig. 1c at least 5 *REQ* messages are broadcast as A wants to pay E initially. If B wants to pay E in a later phase as well, it might have the information already available and does not need to send a *REQ* message but can execute a payment straight away.

While the numbers show that the adaptation of AODV for payment routing in off-chain channel networks is quite feasible for networks of nodes up to a few thousand participants we doubt it is scalable for millions of users or more. As



(a) 500 Nodes (b) 5000 Nodes

Fig. 2. Evaluation results in % (left scale) and number of messages (right scale)

in AODV the message size is comparably small (∼80 bytes) a network should be able to handle easily several thousand requests simultaneously as this would end up in only a few megabytes. Although established routes may expire over time, using a route maintenance message, crucial information can be updated. For that, the original route request issuer sends out a maintenance message along the desired route. Each intermediate node updates the information with its current fees and the distribution of funds in its channels and forwards the message along the path. Hence, once a route is established between two nodes, it may be reused infinitely if updated regularly.

Compared to Flare, our AODV-based protocol does not consider security protection of the message or its content. Hence, the sender and receiver may be publicly known along the payment route. However, significant attempts have been done to secure the AODV routing protocol [6]. Different extensions to AODV to increase the security have been proposed in the past: e.g. SAODV or ARAN which authenticates non-mutable fields and mutable information (hop count) of the message, using digital signature and hash chain [25,27,30]. These extensions can prevent tampering of control messages and data dropping attacks. We argue that a protection of data tempering in a *REP* or *REQ* message is not necessarily required as a route execution follows the atomic principle, i.e. either all transaction are successful or all fail. This means, if an intermediate node lies about the fees it would take (it promises a lower fee but would take a higher fee by falsifying the *REP* message), the payment will fail later on as the sender will only attach enough money so that the desired amount arrives at the receiver who in turn will reject the payment. Privacy and anonymity has been a direct focus of Flare which integrates Onion Routing in a way that only the last node actually knows who has been sending something to whom, all the nodes in the middle just forward it to the next hop. There is more privacy in such a system, however it has been shown that in cases where every node broadcasts the transaction onto the blockchain in similar time frames, clues of the routing can be deducted by the entire network. Even more, in source routing, edge nodes can misuse this anonymity and attack an intermediate node in a way that it is unable to forward future payments. To do so, the attacker will need to have a higher funding (either concentrated on one node or several nodes) than the victim and it will need to be able to controller the payee. The attacker issues a payment request via an intermediate node, which will need to lock up funding for some time. This lock will be released automatically in case the payment was not successfully. However, in the meantime this node cannot use this funding for other purposes such as participating in other payment processes and will not earn money through additional fees. The same attack can be done using AODV. However, the difference is that in AODV each sender is known to intermediate nodes while this is not the case if using Onion Routing where the sender (and the final receiver) remains anonymously. Hence, using AODV each node can protect itself by either accepting or rejecting route requests by specific nodes.

Last but not least, a general problem of decentralised routing requires each participating node to be online as offline nodes are not able to forward any requests. This is why, the LN (or other PCNs) incentives nodes to stay online as they can earn transaction fees by routing payments through them.

7 Conclusion

In this paper we presented an adaptation of AODV for payment routing in payment channel networks such as Lightning, Raiden, or COMIT. We enhanced the messages with information on fees and exchanges rates in order to find a economical route through the network. AODV is a reactive routing protocol that only establishes a route when needed, thus avoiding the overhead of superfluous messages sent in a proactive routing protocol. However, AODV carries the risk of flooding the network if the maximal amount of hops is not set correctly. Our experiments reveal that the adapted AODV can easily be used in a network up to a few thousand nodes. Hence, AODV-based routing can be integrated into PCN. In future work we will focus on routing protocols that scalable further and evaluate how fee management of nodes impact the liquidity flow. Among respective developers, limited liquidity is a well-known problem for off-chain channel networks which remains to be solved.

References

1. Albrightson, B., Garcia-Luna-Aceves, J., Boyle, J.: EIGRP - a fast routing protocol based on distance vectors (1998)
2. Chao, L., Aiqun, H.: Reducing the message overhead of AODV by using link availability prediction. In: Zhang, H., Olariu, S., Cao, J., Johnson, D.B. (eds.) MSN 2007. LNCS, vol. 4864, pp. 113–122. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-77024-4_12
3. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 3rd edn. MIT Press, Cambridge (2009)

4. Decker, C., Wattenhofer, R.: A fast and scalable payment network with Bitcoin duplex micropayment channels. In: Pelc, A., Schwarzmann, A.A. (eds.) SSS 2015. LNCS, vol. 9212, pp. 3–18. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21741-3_1
5. Fonseca, R., Ratnasamy, S., Zhao, J., Ee, C.T., Culler, D., Shenker, S., Stoica, I.: Beacon vector routing: scalable point-to-point routing in wireless sensornets. In: Proceedings of Symposium on Networked Systems Design and Implementation (2005)
6. Gharehkoolchian, M., Hemmatyar, A.M.A., Izadi, M.: Improving security issues in MANET AODV routing protocol. In: Mitton, N., Kantarci, M.E., Gallais, A., Papavassiliou, S. (eds.) ADHOCNETS 2015. LNICST, vol. 155, pp. 237–250. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-25067-0_19
7. Haas, Z.J., Pearlman, M.R., Samar, P.: The Zone Routing Protocol (ZRP) for Ad Hoc Networks. IETF Internet Draft (2002)
8. Hosp, J., Hoenisch, T., Kittiwongsunthorn, P.: COMIT - cryptographically-secure off-chain multi-asset instant transaction network (2017). http://www.comit.network/doc/COMIT%20white%20paper%20v1.0.2.pdf
9. Johnson, D.B., Maltz, D.A.: Dynamic source routing in ad hoc wireless networks. In: Imielinski, T., Korth, H.F. (eds.) Mobile Computing. Springer, Boston (1996). https://doi.org/10.1007/978-0-585-29603-6_5
10. Karp, B., Kung, H.T.: GPSR: greedy perimeter stateless routing for wireless networks. In: International Conference on on Mobile Computing and Networking. ACM (2000)
11. Medhi, D.: Network routing: algorithms, protocols, and architectures (2010)
12. Miller, A., Bentov, I., Kumaresan, R., McCorry, P.: Sprites: payment channels that go faster than lightning (2017)
13. Mistry, N., Jinwala, D.C., Zaveri, M., et al.: Improving AODV protocol against blackhole attacks. In: International Multi Conference of Engineers and Computer Scientists (2010)
14. Mitton, N., Fleury, E.: Distributed node location in clustered multi-hop wireless networks. In: Cho, K., Jacquet, P. (eds.) AINTEC 2005. LNCS, vol. 3837, pp. 112–127. Springer, Heidelberg (2005). https://doi.org/10.1007/11599593_9
15. Murthy, S., Garcia-Luna-Aceves, J.J.: An efficient routing protocol for wireless networks. Mob. Netw. Appl. **1**, 183–197 (1996)
16. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system (2008). https://bitcoin.org/bitcoin.pdf. Accessed 17 Apr 2017
17. Pacia, C.: Lightning network skepticism (2015). https://chrispacia.wordpress.com/2015/12/23/lightning-network-skepticism/. Accessed 22 Mar 2018
18. Pei, G., Gerla, M., Hong, X.: LANMAR: landmark routing for large scale wireless ad hoc networks with group mobility. In: ACM International Symposium on Mobile Ad Hoc Networking and Computing (2000)
19. Perkins, C.E., Royer, E.M.: Ad-hoc on-demand distance vector routing. In: Second IEEE Workshop on Mobile Computing Systems and Applications (1999)
20. Poon, J., Dryja, T.: The Bitcoin lightning network: scalable off-chain instant payments (2015)
21. Prihodko, P., Zhigulin, S., Sahno, M., Ostrovskiy, A., Osuntokun, O.: Flare: an approach to routing in lightning network (2016)
22. Raiden: Raiden network (2016). http://raiden.network/. Accessed 07 Aug 2017
23. Reed, M.G., Syverson, P.F., Goldschlag, D.M.: Anonymous connections and onion routing. IEEE J. Sel. Areas Commun. **16**, 482–494 (1998)

24. Ripple: Ripple paths. https://ripple.com/build/paths. Accessed 12 Sept 2017
25. Sanzgiri, K., Dahill, B., Levine, B.N., Shields, C., Belding-Royer, E.M.: A secure routing protocol for ad hoc networks. In: IEEE International Conference on Network Protocols (2002)
26. Song, R., Korba, L., Yee, G.: AnonDSR: efficient anonymous dynamic source routing for mobile ad-hoc networks. In: ACM Workshop on Security of Ad Hoc and Sensor Networks (2005)
27. Wadbude, D., Richariya, V.: An efficient secure AODV routing protocol in MANET. Int. J. Eng. Innov. Technol. **1**, 274–279 (2012)
28. Wang, L., Shu, Y., Dong, M., Zhang, L., Yang, O.W.: Adaptive multipath source routing in ad hoc networks. In: IEEE International Conference on Communications (2001)
29. Watts, D.J., Strogatz, S.H.: Collective dynamics of ‘small-world’ networks. Nature **393**, 440–442 (1998)
30. Zapata, M.G., Asokan, N.: Securing ad hoc routing protocols. In: Proceedings of the 1st ACM workshop on Wireless Security (2002)

Application Track: Blockchain Solutions

Faster Dual-Key Stealth Address for Blockchain-Based Internet of Things Systems

Xinxin Fan(✉)

IoTeX, Menlo Park, USA
xinxin@iotex.io

Abstract. Stealth address prevents public association of a blockchain transaction's output with a recipient's wallet address and hides the actual destination address of a transaction. While stealth address provides an effective privacy-enhancing technology for a cryptocurrency network, it requires blockchain nodes to actively monitor all the transactions and compute the purported destination addresses, which restricts its application for resource-constrained environments like Internet of Things (IoT). In this paper, we propose DKSAP-IoT, a faster dual-key stealth address protocol for blockchain-based IoT systems. DKSAP-IoT utilizes a technique similar to the TLS session resumption to improve the performance and reduce the transaction size at the same time between two communication peers. Our theoretical analysis as well as the extensive experiments on an embedded computing platform demonstrate that DKSAP-IoT is able to reduce the computational overhead by at least 50% when compared to the state-of-the-art scheme, thereby paving the way for its application to blockchain-based IoT systems.

Keywords: Dual-key stealth address · Blockchain · Internet of Things

1 Introduction

The Internet of Things (IoT) has been connecting extraordinarily large number of smart devices to the Internet and driving the digital transformation of industry. Unfortunately, existing cloud-centric IoT systems have a number of significant disadvantages such as high system maintenance costs, slow response time, security and privacy concerns, etc. Blockchain, a form of distributed, immutable and time-stamped ledger technology, has been perceived as a promising solution to address the aforementioned problems and to securely unlock the business and operational values of IoT. The combination of blockchain and IoT facilitates the sharing of services and resources, creates audit trails and enables automation of time-consuming workflows in various applications. While combining these two technologies is creating new levels of trust, the decentralized network and public verifiability of blockchain transactions often do not provide the strong security and privacy properties required by the users.

S. Chen et al. (Eds.): ICBC 2018, LNCS 10974, pp. 127–138, 2018.
https://doi.org/10.1007/978-3-319-94478-4_9

During the past few years, quite a few cryptographic techniques such as ring signature [10], stealth address [1], and zero-knowledge proof [4] have been employed to ensure transaction privacy for senders, receivers and transaction amount in blockchains [8,11,13,22]. This work focuses on stealth address, a privacy protection technique for receivers of cryptocurrencies. Stealth address requires the sender to create random one-time addresses for every transaction on behalf of the recipient so that different payments made to the same payee unlinkable. The most basic stealth address scheme [1] was first sketched by a Bitcoin Forum member named 'ByteCoin' in 2011, which was then improved in [13,19] by introducing the random ephemeral key pair and fixing the issue that the sender might change the mind and reverse the payment. Later on, a dual-key enhancement [18] to the previous stealth address schemes was implemented in 2014, which utilized two pairs of cryptographic keys for designated third parties (e.g., auditors, proxy servers, read-only wallets, etc.) removing the unlinkability of the stealth addresses without simultaneously allowing payments to be spent.

The dual-key stealth address protocol (DKSAP) provides strong anonymity for transaction receivers and enables them to receive unlinkable payments in practice. However, this approach does require blockchain nodes to constantly compute the purported destination addresses and find the corresponding matches in the blockchain. While this process works well for full-fledged computers, it poses new challenges for resource-constrained IoT devices. Considering the limited energy budget of smart devices, we propose a lightweight variant of DKSAP, namely DKSAP-IoT, which is based on the similar idea as the TLS session resumption [2,14] and requires both the sender and receiver to keep track of the continuously updated pairwise keys for each payment session. DKSAP-IoT is able to improve the performance of DKSAP by at least 50% and reduce the transaction size simultaneously, thereby providing an efficient solution to protecting the privacy of recipients in blockchain-based IoT systems.

The rest of the paper is organized as follows: Sect. 2 gives a brief overview of the elliptic curve cryptography, followed by the description of the dual-address stealth address protocol (DKSAP) in Sect. 3. In Sect. 4, we present DKSAP-IoT, a faster dual-key stealth address protocol for blockchain-based IoT systems. Section 5 analyzes the security and performance of the proposed scheme. Finally, Sect. 6 concludes this contribution.

2 Preliminaries

An elliptic curve E over a field $\mathbb{F}$ is defined by the Weierstrass equation:

$$E(\mathbb{F}) : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6,$$

where $a_1, a_2, a_3, a_4, a_6 \in \mathbb{F}$ and the curve discriminant $\Delta \neq 0$. The set of solutions $(x, y) \in \mathbb{F} \times \mathbb{F}$ satisfying the above equation along with the identity element $\mathcal{O}$, or point-at-infinity, form an abelian group under the addition operation $+$ (i.e., the chord-and-tangent group law). It is this abelian group that is used in the construction of elliptic curve cryptosystems. Given an elliptic curve point

$G \in E(\mathbb{F})$ and an integer k, the scalar multiplication kG is defined by the addition of the point G to itself $k-1$ times, i.e.,

$$kG = \underbrace{G + G + \cdots + G}_{k-1 \text{ additions}}.$$

The scalar multiplication is the fundamental operation in elliptic curve based cryptographic protocols such as the Elliptic Curve Diffie-Hellman (ECDH) key agreement [16] and the Elliptic Curve Digital Signature Algorithm (ECDSA) [16], etc. The security of elliptic curve cryptosystems is based on the difficulty of solving the Elliptic Curve Discrete Logarithm Problem (ECDLP) [6,7]. This problem involves finding the integer k ($0 < k < n$) given a point kG, where n is the group order of $E(\mathbb{F})$. The 15 elliptic curves have been recommended by NIST in the FIPS 186-2 standard for U.S. federal government [3], which are also contained in the specification defined by the Standards for Efficient Cryptography Group (SECG) [17]. For example, the elliptic curve used in Bitcoin is called secp256k1 with parameters specified by SECG [17]. For more details about elliptic curve cryptography, the interested reader is referred to [5].

3 Dual-Key Stealth Address Protocol (DKSAP)

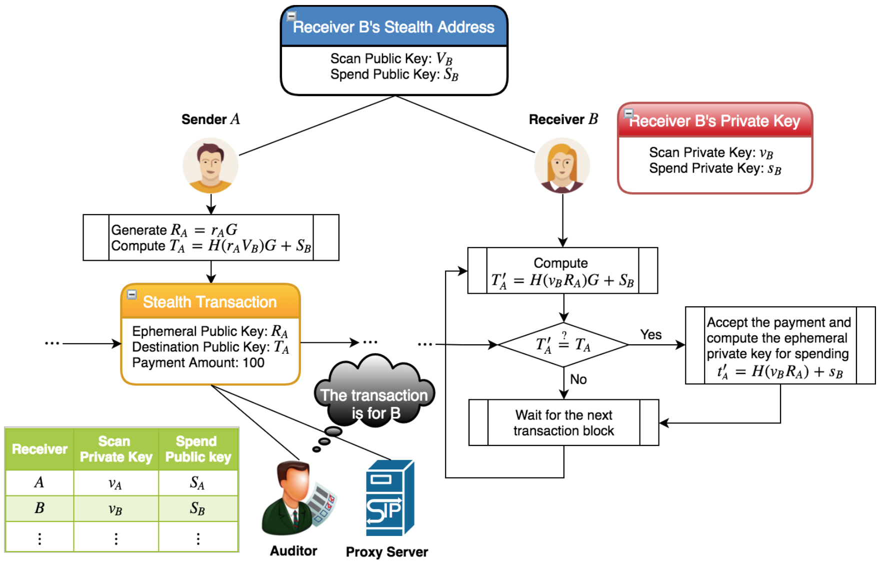
The first full working implementation of DKSAP was announced by a developer known as rynomster/sdcoin in 2014 for **ShadowSend** [18], a capable, efficient and decentralized anonymous wallet solution. The DKSAP has been realized in a number of cryptocurrency systems since then, including **Monero** [8], **Samourai Wallet** [15], **TokenPay** [20], just to name a few. The protocol takes advantage of two pairs of cryptographic keys, namely a 'scan key' pair and a 'spend key' pair, and computes a one-time payment address per transaction, as illustrated in Fig. 1.

When a sender A would like to send a transaction to a receiver B in a stealth mode [18], DKSAP works as follows:

1. The receiver B has a pair of private/public keys (v_B, V_B) and (s_B, S_B), where v_B and s_B are called B's 'scan private key' and 'spend private key', respectively, whereas $V_B = v_B G$ and $S_B = s_B G$ are the corresponding public keys. Note that none of V_B and S_B ever appear in the blockchain and only the sender A and the receiver B know those keys.
2. The sender A generates an ephemeral key pair (r_A, R_A) with $R_A = r_A G$ and $0 < r_A < n$, and sends R_A to the receiver B.
3. Both the sender A and the receiver B can perform the ECDH protocol to compute a shared secret:

$$c_{AB} = H(r_A v_B G) = H(r_A V_B) = H(v_B R_A),$$

where $H(\cdot)$ is a cryptographic hash function.


Receiver B's Stealth Address
Scan Public Key: V_B
Spend Public Key: S_B
Sender A
Receiver B
Receiver B's Private Key
Scan Private Key: v_B
Spend Private Key: s_B
Generate $R_A = r_A G$
Compute $T_A = H(r_A V_B)G + S_B$
Compute $T'_A = H(v_B R_A)G + S_B$
Stealth Transaction
Ephemeral Public Key: R_A
Destination Public Key: T_A
Payment Amount: 100
$T'_A \overset{?}{=} T_A$
Yes
No
Accept the payment and compute the ephemeral private key for spending $t'_A = H(v_B R_A) + s_B$
Wait for the next transaction block
The transaction is for B
Auditor
Proxy Server

Receiver	Scan Private Key	Spend Public key
A	v_A	S_A
B	v_B	S_B
⋮	⋮	⋮

Fig. 1. The Dual-Key Stealth Address Protocol (DKSAP)

4. The sender A can now generate the destination address of the receiver B to which A should send the payment:

$$T_A = c_{AB}G + S_B.$$

Note that the one-time destination address T_A is publicly visible and appears on the blockchain.

5. Depending on whether the wallet is encrypted, the receiver B can compute the same destination address in two different ways:

$$T'_A = c_{AB}G + S_B = (c_{AB} + s_B)G.$$

The corresponding ephemeral private key is

$$t'_A = c_{AB} + s_B,$$

which can only be computed by the receiver B, thereby enabling B to spend the payment received from A later on.

In DKSAP, the receiver B needs to actively scan the blockchain transactions, calculate the purported destination address and compare it with the one in each block until a match is found. In the case that an auditor or a proxy server exists in the system, the receiver B can share the 'scan private key' v_B and the 'spend public key' S_B with the auditor/proxy server so that those entities can scan the blockchain transactions on behalf of the receiver B. However, they are not able to compute the ephemeral private key t'_A and spend the payment received from the sender A.

4 Faster Dual-Key Stealth Address Protocol for Internet of Things (DKSAP-IoT)

In this section, we describe a faster dual-key stealth address protocol called DKSAP-IoT, which is dedicatedly designed for blockchain-based IoT systems.

4.1 Design Rationale

In DKSAP, the receiver B scans the blockchain and calculates the purported destination address for each transaction, which requires computations of two scalar multiplications, including one random-point scalar multiplication with the ephemeral public key R_A and one fixed-point scalar multiplication with the base point G. For resource-constrained IoT devices, computing two scalar multiplications continuously for each blockchain transaction is going to drain battery power of smart devices dramatically. Furthermore, containing an ephemeral public key in each stealth payment increases the size of the transaction and incurs additional communication overhead for IoT devices as well.

Motivated by the TLS session resumption techniques [2,14], we aim to accelerate the process for receivers finding the matched destination address by extending the lifetime of the shared secret between senders and receivers. While both the session ID [2] and the session ticket [14] are fixed in TLS for a given period of time between the client and server, the sender does need to generate a one-time destination address for each payment sent to the same recipient in our case. To this end, both the sender and receiver will apply the cryptographic hash function to their shared secret for subsequent N transactions before the sender initiates a shared secret update with a fresh ephemeral public key. This key evolving process is shown in Fig. 2, which leads to the design of DKSAP-IoT, a faster dual-key stealth address protocol for blockchain-based IoT systems, as detailed in the next subsection.

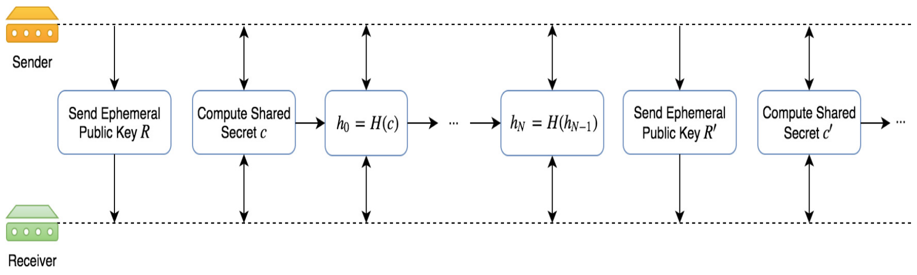

Sender
Receiver
Send Ephemeral Public Key R
Compute Shared Secret c
$h_0 = H(c)$
...
$h_N = H(h_{N-1})$
Send Ephemeral Public Key R'
Compute Shared Secret c'
...

Fig. 2. The key evolving process between the sender and receiver in DKSAP-IoT

4.2 DKSAP-IoT Specification

DKSAP-IoT is similar to DKSAP except that whenever the sender and receiver establish a shared secret using ECDH it will be continuously and pseudorandomly updated with a cryptographic hash function and used in their subsequent

N stealth transactions. Both the sender and receiver maintain the transaction state (i.e., shared secret, counter, etc.) locally and update it after each stealth transaction. A high-level description of DKSAP-IoT is depicted in Fig. 3.

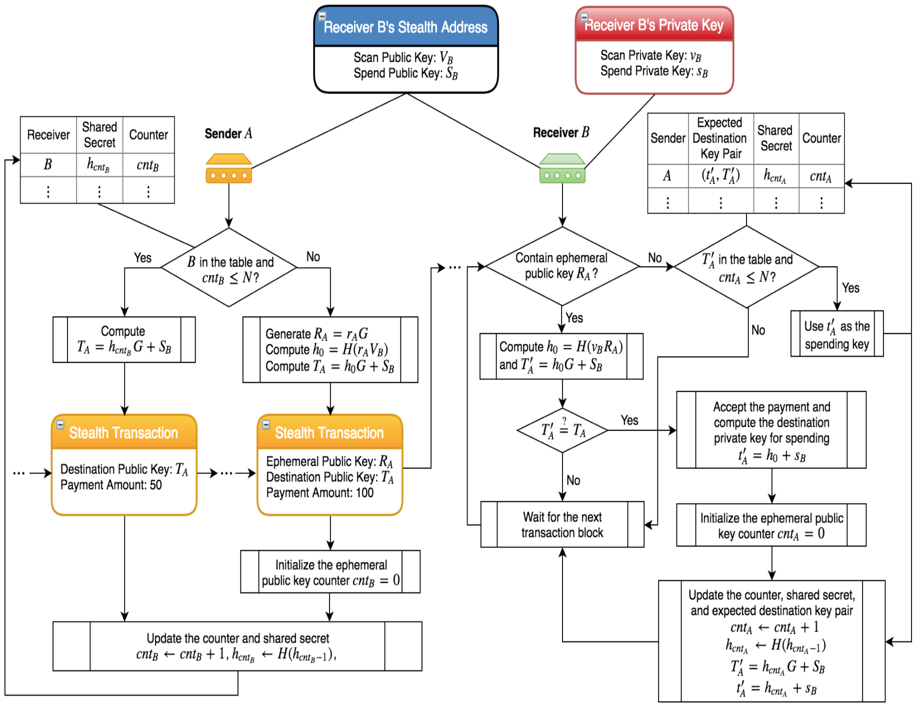

Receiver B's Stealth Address
Scan Public Key: V_B
Spend Public Key: S_B
Receiver B's Private Key
Scan Private Key: v_B
Spend Private Key: s_B
Sender A
Receiver B
Receiver | Shared Secret | Counter
B | h_{cnt_B} | cnt_B
Sender | Expected Destination Key Pair | Shared Secret | Counter
A | (t'_A, T'_A) | h_{cnt_A} | cnt_A
B in the table and $cnt_B \leq N$?
Yes
No
Compute $T_A = h_{cnt_B}G + S_B$
Generate $R_A = r_A G$ Compute $h_0 = H(r_A V_B)$ Compute $T_A = h_0 G + S_B$
Stealth Transaction
Destination Public Key: T_A
Payment Amount: 50
Stealth Transaction
Ephemeral Public Key: R_A
Destination Public Key: T_A
Payment Amount: 100
Initialize the ephemeral public key counter $cnt_B = 0$
Update the counter and shared secret $cnt_B \leftarrow cnt_B + 1, h_{cnt_B} \leftarrow H(h_{cnt_B - 1})$,
Contain ephemeral public key R_A?
T'_A in the table and $cnt_A \leq N$?
Use t'_A as the spending key
Compute $h_0 = H(v_B R_A)$ and $T'_A = h_0 G + S_B$
$T'_A \overset{?}{=} T_A$
Accept the payment and compute the destination private key for spending $t'_A = h_0 + s_B$
Wait for the next transaction block
Initialize the ephemeral public key counter $cnt_A = 0$
Update the counter, shared secret, and expected destination key pair
$cnt_A \leftarrow cnt_A + 1$
$h_{cnt_A} \leftarrow H(h_{cnt_A - 1})$
$T'_A = h_{cnt_A}G + S_B$
$t'_A = h_{cnt_A} + s_B$

Fig. 3. The Dual-Key Stealth Address Protocol for IoT (DKSAP-IoT)

In a blockchain-based IoT system, two smart devices A and B can process a transaction in a stealth mode using DKSAP-IoT as described below:

1. The receiver device B is pre-installed with a 'scan key' pair (v_B, V_B) and a 'spend key' pair (s_B, S_B) as in DKSAP, where $V_B = v_B G$ and $S_B = s_B G$.
2. For sending a transaction to B, the sender device A first checks whether B is in A's receiver list. If B is in the list and the counter value cnt_B is less than N (i.e., A has communicated with B before), A retrieves the shared secret h_{cnt_B} from the table and computes the destination public key:

$$T_A = h_{cnt_B} G + S_B.$$

The stealth transaction that only contains the destination public key T_A as well as the payment amount is then added into the blockchain. In the case that B is not in the list or the counter value is greater than N, A generates a fresh ephemeral public key $R_A = r_A G$ and calculates the shared secret $h_0 = H(r_A V_B)$ as well as the destination public key as in DKSAP:

$$T_A = h_0 G + S_B.$$

Here the stealth transaction is composed of the ephemeral public key R_A, the destination public key T_A and the payment amount. After putting the transaction on the blockchain, the sender A will initialize the ephemeral public key counter $cnt_B = 0$. In both cases, the counter cnt_B and the shared secret h_{cnt_B} will be updated as well:

$$cnt_B \leftarrow cnt_B + 1, \;\; h_{cnt_B} \leftarrow H(h_{cnt_B - 1}).$$

Note that only the counter cnt_B is updated when it reaches N.

3. Upon receiving a stealth transaction, the receiver B first checks whether the transaction contains an ephemeral public key R_A. If it is, B computes the purported shared secret and destination public key:

$$h_0 = H(v_B R_A), \;\; T'_A = h_0 G + S_B.$$

If the purported destination public key T'_A matches the received one (i.e., $T'_A = T_A$), B accepts the payment from A and computes the corresponding private key for spending:

$$t'_A = h_0 + s_B.$$

B also sets the ephemeral public key counter cnt_A to be 0, updates the counter and shared secret, and precomputes the expected destination key pair for the next stealth transaction from A:

$$cnt_A \leftarrow cnt_A + 1, \;\; h_{cnt_A} \leftarrow H(h_{cnt_A - 1}), \tag{1}$$

$$T'_A = h_{cnt_A} G + S_B, \;\; t'_A = h_{cnt_A} + s_B. \tag{2}$$

When B receives a stealth transaction without an ephemeral public key, B will check whether the received destination public key T_A is contained in its list of senders. If a match is found and the value of the counter cnt_A is less than or equal to N, B retrieves the corresponding destination private key t'_A as the spending key and updates the transaction state information accordingly with the Eqs. (1) and (2). Again only the counter cnt_A is updated when it reaches N.

In DKSAP-IoT, stealth transactions are divided into two categories depending on whether ephemeral public keys are included in the blocks. For the first stealth transaction between two blockchain nodes, the receiver needs to conduct the same operations as DKSAP, followed by a more efficient preparation process for the next transaction. For the subsequent N stealth transactions between the same peers, generating a fresh ephemeral key is no longer needed on the sender side. Meanwhile, the receiver only performs a fast table look-up as well as the transaction state updates, which facilitates the receiver to quickly filter out the designated transactions.

Given the 'scan private key' v_B and the 'spend public key' S_B, the auditor/proxy server is able to calculate all the destination addresses for the receiver B, thereby tracking or forwarding all the transactions to B. However, both the auditor or the proxy server cannot derive the corresponding ephemeral private keys and spend the funds.

5 Security Analysis and Performance Evaluation

In this section, we analyze the security and performance of DKSAP-IoT and report its implementation on a Raspberry Pi 3 Model B, a good representative of moderately resource-constrained embedded devices.

5.1 Security Analysis

DKSAP-IoT follows the same threat model as DKSAP, in which the adversary aims to determine the corresponding recipients by observing the transactions on the blockchain. DKSAP-IoT provides the following security properties:

- **Receiver Anonymity**: DKSAP-IoT offers strong anonymity for receivers and ensures the unlinkability of payments received by the same payee. For each payment to a stealth address, the sender computes a new normal address T_A on which the funds ought to be received. Given two destination addresses $T_A^{(i)} = h_i G + S_B$ and $T_A^{(j)} = h_j G + S_B$ $(0 \le i, j \le N)$ for the same receiver B, the adversary is not able to link them thanks to the difficulty of ECDLP.
- **Forward Privacy**: DKSAP-IoT provides forward secrecy due to the usage of a cryptographic hash function for updating the shared secret continuously for N stealth transactions. If the adversary compromises the device and obtains h_l for the l^{th} $(0 < l < N)$ stealth transaction, he/she is still not able to link previous transactions because of the properties of the hash function.
- **Stealth Transaction Hiding**: In DKSAP, transactions in the stealth mode can be easily distinguished from regular ones in the blockchain due to the presence of ephemeral public keys, thereby resulting in some loss of privacy. However, the ephemeral public key only needs to be updated every N stealth transactions for two communication peers in DKSAP-IoT and those stealth transactions in between are not distinguishable from regular ones.

Since both the sender and receiver need to locally maintain the state information for their peers in DKSAP-IoT (See Fig. 3), these tables, together with the device private keys, should be stored in the encrypted form for mitigating the risk that IoT devices might get compromised. Considering that the hardware AES engine is widely available on many IoT devices, the computational overhead for encrypting/decrypting those sensitive information is quite small.

5.2 Performance Evaluation

Computational and Communication Overhead. We assume that a sender is going to send N stealth transactions to a receiver using blockchain. Let RP, FP and H denote the computation of a random-point scalar multiplication, a fixed-point scalar multiplication and a cryptographic hash function, respectively. Table 1 gives a comparison between the DKSAP and DKSAP-IoT in terms of their computational overhead.

Table 1. Computational overhead of DKSAP and DKSAP-IoT for sending N stealth transactions between two blockchain nodes

Scheme	Sender			Receiver		
	#RP	#FP	#H	#RP	#FP	#H
DKSAP	N	$2N$	N	N	N	N
DKSAP-IoT	1	$N+1$	N	1	N	N

From Table 1, one can see that DKSAP-IoT is able to reduce the number of RP and FP by $N-1$ on the sender side, respectively, when compared to the DKSAP. Moreover, DKSAP-IoT can also save $N-1$ RP on the receiver side. With respect to the communication overhead, the sender in DKSAP-IoT only needs to contain a fresh ephemeral public key in the first stealth transaction, thereby saving the transmission of $N-1$ elliptic curve points.

Software Implementation. To validate the performance improvements of DKSAP-IoT, we implemented an optimized elliptic curve cryptography library, namely libsect283k1[1], using the 283-bit binary Koblitz curve specified in [17]:

$$E(\mathbb{F}_{2^{283}}) : y^2 + xy = x^3 + 1,$$

where the binary field $\mathbb{F}_{2^{283}}$ is defined by $f(x) = x^{283} + x^{12} + x^7 + x^5 + 1$. The library was written in C and compiled using the GNU C Compiler (GCC). A number of efficient techniques, such as the lambda coordinates [9], the window τNAF method [5], the pre-computation [21], etc., have been utilized to optimized the performance of the libsect283k1 library. Moreover, BLAKE-256 [12] is chosen as the hash function in our library due to its high performance cross multiple computing platforms. When running our library on a Raspberry Pi 3 Model B, the timings for the computation of RP, FP and H are shown in Table 2.

Table 2. Timings for computing RP, FP and H on a Raspberry Pi 3 model B

RP	FP	H
3.67 ms	3.12 ms	5.26 µs

[1] libsect283k1 will be integrated into the IoTeX testnet and mainnet as part of the iotex-core (see https://github.com/iotexproject/iotex-core).

Note that the computation of the hash function is about three orders of magnitude faster than that of the scalar multiplication over an elliptic curve. Therefore, using the hash function to update the shared secret and extend its lifetime brings significant performance benefits for IoT devices. Figure 4 compares the performance of the DKSAP and DKSAP-IoT on both sender and receiver sides for sending $N = 10, 20$ and 30 stealth transactions, respectively.

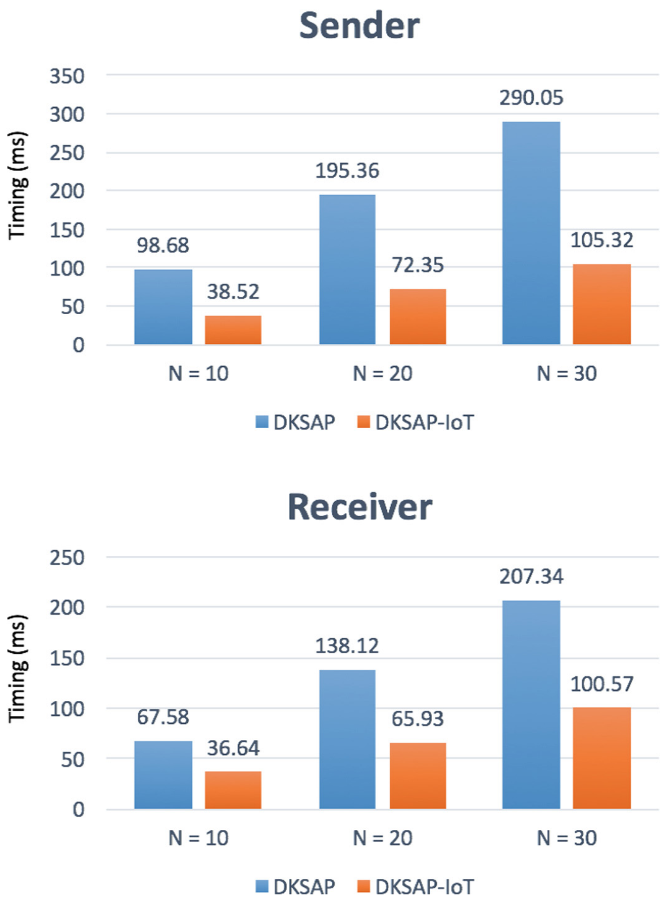


Fig. 4. Performance comparison of the DKSAP and DKSAP-IoT for sending $N = 10, 20$ and 30 stealth transactions

From Fig. 4, one notices that the overall cost of DKSAP-IoT is less than 50% of DKSAP, mainly because extending the lifetime of the shared secret with a cryptographic hash function enables both the sender and receiver to reduce the number of RP from N to 1. Moreover, the computation of the hash function is almost negligible compared to the scalar multiplication over the elliptic curve. In addition, DKSAP-IoT can save the transmission of $72 \cdot (N - 1)$ bytes for N stealth transactions. For resource-constrained IoT devices, the improved performance and reduced transaction size by DKSAP-IoT leads to significant power savings and extended battery life.

6 Conclusion

In this paper, we propose an efficient dual-key stealth address protocol DKSAP-IoT for blockchain-based IoT systems. Motived by the TLS session resumption techniques, we apply a cryptographic hash function to continuously update a shared secret between two communication peers and extend the lifetime of this shared secret for additional N transactions. Both the sender and receiver need to maintain the state information locally in order to keep track of the pair-wise session keys. The security analysis shows that DKSAP-IoT provides receiver anonymity and forward privacy. When implementing DKSAP-IoT on a Raspberry Pi 3 Model B, we demonstrate that DKSAP-IoT can achieve at least 50% performance improvement when compared to the original DKSAP, besides significant reduction of the transaction size in the block. Our work is another logic step towards providing strong privacy protection for blockchain-based IoT systems.

Acknowledgement. The author would like to thank the IoTeX team for the great support during the course of writing this paper and the anonymous reviewers for their insightful comments.

References

1. ByteCoin: Untraceable transactions which can contain a secure message are inevitable. Bitcoin Forum. https://bitcointalk.org/index.php?topic=5965.0
2. Dierks, T., Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard) (2008). https://tools.ietf.org/html/rfc5246
3. Federal Information Processing Standards Publication 186–2. Digital Signature Standard (DSS). National Institute of Standards and Technology (2000)
4. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM J. Comput. **18**(1), 186–208 (1989)
5. Hankerson, D., Menezes, A.J., Vanstone, S.: Guide to Elliptic Curve Cryptography. Springer, Secaucus (2003). https://doi.org/10.1007/b97644
6. Koblitz, N.: Elliptic curve cryptosystems. Math. Comput. **48**(177), 203–209 (1987)
7. Miller, V.S.: Use of elliptic curves in cryptography. In: Williams, H.C. (ed.) CRYPTO 1985. LNCS, vol. 218, pp. 417–426. Springer, Heidelberg (1986). https://doi.org/10.1007/3-540-39799-X_31
8. Monero. Stealth Address. https://getmonero.org/resources/moneropedia/stealthaddress.html
9. Oliveira, T., López, J., Aranha, D.F., Rodríguez-Henríquez, F.: Two is the fastest prime: lambda coordinates for binary elliptic curves. J. Cryptographic Eng. **4**(1), 3–17 (2014)
10. Rivest, R.L., Shamir, A., Tauman, Y.: How to leak a secret. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 552–565. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45682-1_32

11. Rynomster, Tecnovert.: Shadow: Zero-knowledge Anonymous Distributed Electronic Cash via Traceable Ring Signatures. https://coss.io/documents/white-papers/shadowcash.pdf
12. Saarinen, M.-J., Aumasson, J.-P.: The BLAKE2 Cryptographic Hash and Message Authentication Code (MAC). RFC 7693 (Informational) (2015). https://tools.ietf.org/html/rfc7693
13. van Saberhagen, N.: CryptoNote v 2.0. https://cryptonote.org/whitepaper.pdf
14. Salowey, J., Zhou, H., Eronen, P., Tschofenig, H.: Transport Layer Security (TLS) Session Resumption without Server-Side State. RFC 5077 (Proposed Standard) (2008). https://tools.ietf.org/html/rfc5077
15. Samourai. https://samouraiwallet.com/index.html
16. Standards for Efficient Cryptography Group, SEC 1: Elliptic Curve Cryptography, Version 2.0. http://www.secg.org/sec1-v2.pdf
17. Standards for Efficient Cryptography Group, SEC2: Recommended Elliptic Curve Domain Parameters version 2.0. http://www.secg.org/sec2-v2.pdf
18. The Shadow Project. Dual-key Stealth Addresses, in part of Shadow Documentation. https://doc.shadowproject.io/#dual-key-stealth-addresses
19. Todd, P.: [Bitcoin-development] Stealth Addresses. https://www.mail-archive.com/bitcoin-development@lists.sourceforge.net/msg03613.html
20. TokenPay: TokenPay - The World's Most Secure Coin (Whitepaper). https://www.tokenpay.com/whitepaper.pdf
21. Yu, W., Musa, S.A., Xu, G., Li, B.: A Novel Pre-Computation Scheme of Window τNAF for Koblitz Curves. IACR Cryptology ePrint Archive, Report 2017/1020 (2017). https://eprint.iacr.org/2017/1020
22. Zcash. https://z.cash/technology/index.html

Blockchain-Based Solution for Proof of Delivery of Physical Assets

Haya R. Hasan(✉) and Khaled Salah(✉)

Department of Electrical and Computer Engineering,
Khalifa University of Science, Technology and Research, Abu Dhabi, UAE
{haya.hasan,khaled.salah}@kustar.ac.ae

Abstract. To date, building a highly trustworthy, credible, and decentralized proof of delivery (POD) systems to trace and track physical items is a very challenging task. This paper presents a blockchain based POD solution of shipped physical items that uses smart contracts of Ethereum blockchain network, in which tracking, and tracing activities, logs, and events can be done in a decentralized manner, with high integrity, reliability, and immutability. Our solution incentivizes each participating entity including the seller, transporter, and buyer to act honestly, and it totally eliminates the need for a third party as escrow. Our proposed POD solution ensures accountability, punctuality, integrity and auditability. Moreover, the proposed solution makes use of a Smart Contract Attestation Authority to ensure that the code follows the terms and conditions signed by the participating entities. It also allows the cancellation of the transaction by the seller, buyer and transporter based on the contract state. Furthermore, the buyer can also ask for a refund in certain justifiable cases. The full code, implementation discussion with sequence diagrams, testing and verification details are all included as part of the proposed solution.

Keywords: Proof of delivery · Blockchain · Ethereum · Smart contracts

1 Introduction

With the widespread of technology and the internet, online shopping or trading have become part of people's daily activity. Often at the comfort of their homes, people start searching for a desirable item and wonder if there is an online vendor that can provide the item in a perfect condition to their door step. Meeting the needs of today's world, lots of online stores have launched and provided delivery services and even world-wide shipping. Thus, there is an immense need to have a solution that provides proof of delivery of any physical item such as a piece of clothing, book, home essentials etc. delivered between any two parties.

Proof of Delivery (POD) or 'last mile' of delivery is crucial as it shows that an item has reached its final and required destination. In real world, courier and delivery service companies use trackers and proof of delivery systems to ensure that their customers' needs are met on time and without delays. Not only punctuality is important, but also the delivery of the item as is from the initial source is extremely vital.

Today's proof of delivery systems are typically based on signed papers and documents which are carried with the transporter to the recipient. Other courier services might depend on an electronic hand-held device used to obtain the signature of the recipient with a valid ID. This is cumbersome and does not provide total trust for delivery, whereby there is no true and genuine verification by the courier for the signature and the ID of the recipient which can be fake. Even, the courier cannot be trusted. Furthermore, online retailers depend on a third party for shipment. For instance, Amazon depends on several regional courier companies for their delivery services, such as UPS, FedEx, DHL, Pilot and many others [1, 2]. In addition, today's delivery service is completely centralized, costly, and extremely hard to manage. In general, centralized systems suffer from privacy invasion, being a single point of failure, and mistrust which can lead to corruption and attacks.

Problem Statement. *To date, trusted online trading between two unknown parties is not yet established without a centralized trusted third party. There is an immense need for a proof of delivery and tracking of shipped items with a highly trusted, secure, and decentralized traceability and auditability.*

Blockchain is a new disruptive technology that possesses a lot of features that make it ideal for tracking systems. Hence, making it a perfect candidate for creating proof of delivery solutions. Blockchain is an immutable, distributed ledger that is tamper-proof with ordered logs [3, 4]. Ethereum smart contracts made blockchain programmable, as it allowed the execution of code, making it even more powerful [4]. An ideal proof of delivery system should satisfy the following desirable characteristics:

- **Accountability:** It should be possible to trace back the actions performed to the system to the actual initiating entity.
- **Authorization:** Each party in the system is allowed to only perform certain roles.
- **Auditability:** It should be possible to track all activities performed by the acting entities and hence, to trace back the item's state and route.
- **Integrity:** No one should be able to modify the audits and agreed upon terms and conditions.
- **Punctuality:** The system should be able to time every action and deliver the item on time to the customer.
- **Honesty:** Each of the participating entities (seller, transporter and buyer) should be incentivized to act honest and do their part legitimately.

In this paper, we propose a blockchain solution for the proof of delivery of physical items that solves the issue of trust, tracking, tracing back and proves the item reached its final and legitimate destination. The solution can be extended to include intermediary destinations before the final one and can be easily integrated with a Know Your Customer (KYC) protection to add an extra layer of security. We show in our solution that all parties involved in the contract are all equally trusted with the same authority levels. Their roles allow them to execute certain functions only, hence managing the contract flow using role restriction. The main contributions of the paper can be summarized as follows:

- We propose a blockchain-based proof of delivery solution that utilizes tamper-proof logs for auditability and traceability.

- Our solution utilizes an equal agreed upon collateral to incentivize each of the participating entities to act honest. Our PoD solution ensures integrity of the signed terms and conditions form by using InterPlanetary File System (IPFS) hash in the smart contract. Also, the smart contract code is attested by the smart contract attestation authority to ensure that it follows the signed terms and conditions.
- We demonstrate accountability in our solution by using keys and hashes for verification of the true legitimate receiver. We also show how our solution addresses issues related to customer loyalty and delivery punctuality. Refund and cancellation are also taken care of to preserve the rights of the seller, buyer and transporter.
- We present a complete system architecture, sequence diagrams and the full code with the implementation and testing details.

The remainder of this paper is organized as follows. Section 2 provides the related work. Section 3 presents the system architecture of the proposed solution. Section 4 describes the important aspects of the implementation. Section 5 illustrates the testing and validation of the smart contract code and Sect. 6 concludes the paper.

2 Related Work

This section presents the work that is available in the literature for proposed algorithms and techniques for proof of delivery of physical items. All blockchain solutions try to solve the trust issue between the seller, buyer and transporter while tracking the item's state through the logged events.

For instance, [5] proposes using a secure hash and a key that is given by the seller to the transporter along with the item. Once the transporter arrives to the destination, the buyer would enter the key and the hash of the key is compared with the hash already available in the contract. This method is simple and easy to implement. It uses the contract as an escrow. Therefore, the item's price would only be given to the seller once the hash verification is done. However, it involves trusting the transporter not to manipulate the key. Furthermore, its success depends on all the parties acting honest and trust worthy and this cannot be guaranteed.

On the other hand, localethereum [6], an Ethereum market place, does not use the contract as an escrow. It uses a third party as a funded escrow. The seller and buyer or traders agree upon a trusted third party as escrow that they would trust till the end of the transaction. Also, in case of dispute, the involved parties can allow localethereum to act as an arbitrator. Although using a third party as an escrow costs more and also requires more incentives for trust and honesty, having an arbitrator incase of a dispute makes it more convenient.

Furthermore, BitBay [7], a decentralized marketplace which provides a platform for users to trade and sell, uses a 'double-deposit escrow'. This means that a collateral is deposited by the traders or buyer and seller that is equal to double the item price being traded or sold. BitBay eliminates the need of a trusted third party as an escrow. Hence, it uses the contract itself as an escrow that takes all the deposits until the transaction is completed. BitBay also does not act as a moderator themselves in case of issues unlike localethereum. However, using BitBay the delivery of a physical item would require trusting a third party for the transportation. There is no guarantee that the transporter would not manipulate the delivered item or create a delay.

An important point is also to be able to integrate the system with a KYC procedure to verify the identity of the parties involved in the contract. For instance, this can be achieved by using a certification body that takes the signed personal information of a user and stores their hash in the blockchain. Later on, when a user would like to register in a service, the service providers would verify the submitted signature hash with the hash value of the personal information acquired from the certification body. If both the hashes match, then the identification is complete [8].

3 Proposed Blockchain Solution

In this section, we propose a solution that utilizes the Ethereum blockchain to create a system that is decentralized, provides trust, and uses immutable logs and events. This system is a solution for selling physical items with a proof of delivery and without the need of an intermediary service such as a broker or an agent. This is done by using smart contracts which facilitate the automation of the process and help in saving the history of all transactions without alterations.

3.1 System Overview and Design

The proposed solution is focused on proof of delivery of physical items between a seller and an interested buyer. A transporter is also part of the contract to ensure the item delivery is carried out and all other parties are notified about the status of the item during delivery. An agreed upon and trusted arbitrator by the seller, buyer and transporter is also part of the contract and can only step in incase of a dispute. Each of the mentioned participants possess an Ethereum address and they all sign an agreement form which has all the terms and conditions of the contract. Furthermore, an important part of our solution, is the attestation of the contract from a certified trusted authority, Smart Contract (SC) Attestation Authority. The attestation guarantees to all participating entities that the smart contract follows the signed agreement terms and conditions. The roles of the Ethereum entities in the smart contract are as follows:

Seller: The seller is the owner of the item to be sold. The seller creates the contract, signs the agreement terms and conditions form, deposits a collateral, and provides keys to both the buyer and transporter.

Buyer: The buyer is the entity that shows interest in the item being sold. The buyer agrees to the terms and conditions, deposits a collateral along with the item price, and requests an item key to be able to take the item from the transporter when delivered.

Transporter: The transporter's main role is to deliver the item as received between the seller and buyer.

Arbitrator: The arbitrator is a trusted third party that is agreed upon by the seller, buyer and transporter who will only get involved in case of a dispute to solve the issues off the chain.

SC Attestation Authority: The smart contract attestation authority ensures that the smart contract complies with the terms and conditions signed by the involved parties in the agreement form.

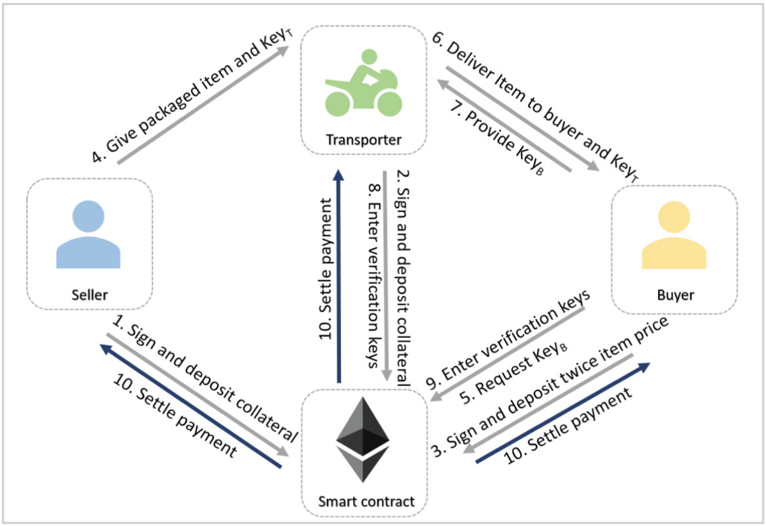

Transporter
Seller
Buyer
Smart contract
1. Sign and deposit collateral
2. Sign and deposit collateral
3. Sign and deposit twice item price
4. Give packaged item and Key$_T$
5. Request Key$_B$
6. Deliver Item to buyer and Key$_T$
7. Provide Key$_B$
8. Enter verification keys
9. Enter verification keys
10. Settle payment
10. Settle payment
10. Settle payment

Fig. 1. System architecture of the Ethereum-based solution showing the main participating entities participating in a successful transaction

The smart contract of the proposed solution follows a certain algorithm that flows in a sequence that should be followed by the participating entities to preserve everyone's right. All actions that take place off the chain are accompanied by functions in the contract that trigger logged events and notifications. The contract receives the agreed upon collateral which is in our case twice the item price from the seller, buyer and transporter as they are equally trusted by each other and by the smart contract. The collateral can be of any reasonable value as long as it is equal among all entities. The funds are held by the contract, so the contract acts as escrow until the payment is settled based on the events and verification results. The smart contract contains the following:

- **Modifiers:** Modifiers ensure that transactions and functions are executed by the right legitimate entities and that the payable functions only accept the correct intended payment before proceeding. Modifiers change the function that uses them to allow it to execute based on the result of another code that is first executed inside the modifier. For example, requiring the collateral by each entity to be twice the actual price of the item or requiring a certain function to be executed by only the seller, buyer, transporter or any one of them is specified using the modifiers: `costs()`, `OnlySeller()`, `Only-Buyer()`, `OnlyTranspoter()`, `OnlySeller_Buyer_Tranporter()`.
- **Events:** After the execution of a function, an event is used to create notifications and saved logs. Events help in tracing and in notifying all participating parties about the current state of the contract and current activities.

- **Variables:** Variables help in saving important values that preserve the state of the contract as it changes along with the functions. Variables used in the contract hold addresses of the participating entities, IPFS hash of the agreement terms and conditions form, item's details such as price and ID, and current contract state.

Figure 1 shows the system architecture of the proposed blockchain solution that shows the transactions between the seller, buyer and transporter. All the parties sign the terms and conditions agreement form and agree to its content by depositing a collateral that is twice the item price. Then the seller prepares the item and hands it over to the transporter along with a key ($\mathrm{Key_T}$). Every item has two keys that are given by the seller, i.e. a key that is given to the transporter and a key that is handed over to the buyer ($\mathrm{Key_B}$). The transporter delivers the item to the buyer and they both exchange their keys. This ensures that the transporter has reached the intended buyer. Both of the transporter and buyer enter the keys to the contract and verification takes place. The smart contract computes the hashes of the keys entered and if the hashes match, then the payment is settled. The buyer is refunded one item price, the transporter gets back twice the item price that was deposited as collateral with 10% of the item price additional payment for the delivery service. Lastly, the seller gets the rest of the ether deposits which include the rest of the deposited collateral and the payment of the item price by the buyer.

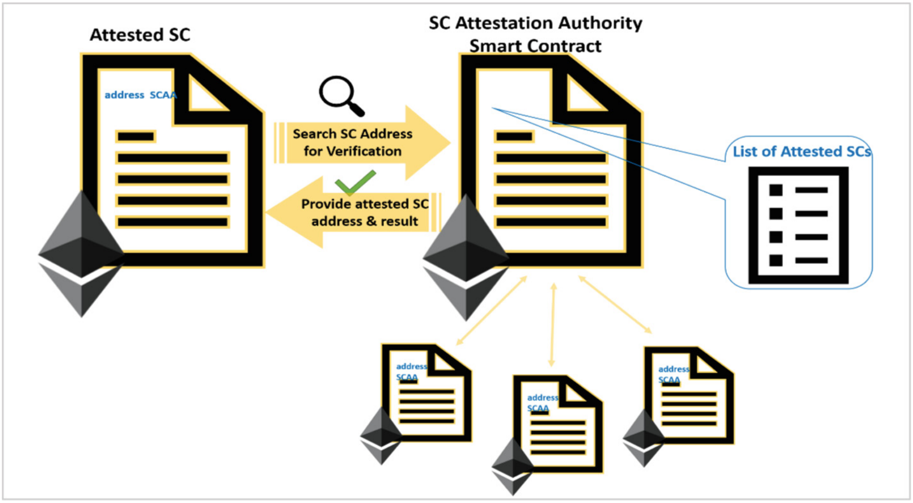

Attested SC
address SCAA
Search SC Address for Verification
Provide attested SC address & result
SC Attestation Authority Smart Contract
List of Attested SCs
address SCAA
address SCAA
address SCAA

Fig. 2. Attested SC pointing to the Ethereum address of the SC Attestation Authority which also has a list of all the Ethereum addresses of the attested contracts

The contract used in the blockchain solution is an attested contract. Hence, the contract code and flow was verified and attested by the Smart Contract (SC) Attestation Authority. Therefore, the attested SC includes the Ethereum address of the attestation authority that verified the contract code and ensured that it agrees to the terms and conditions in the agreement form that is signed by all the parties of the contract. Moreover, the SC Attestation Authority has a complete list of the all the Ethereum addresses of the attested contracts. This provides a way for the participating entities to trust the contract content and to be able to verify that it has truly been attested and is now trustable. Figure 2 shows the attested smart contract and the SC Attestation Authority contract along with their relationship.

4 Implementation Details

The smart contract code is written in Solidity using the web-based IDE, Remix. The code focuses on three main entities, the seller, transporter and buyer to acquire the proof of delivery of a physical item. However, there are two other entities that also contribute in making the contract more trustable. The arbitrator steps in incase of a dispute and the SC Attestation Authority attests and verifies that the contract code matches the terms and conditions signed by each of the actively participating entities.

Figure 3, shows the sequence diagram that demonstrates the flow of the code. The code represents a seller who has an item and would like to sell it to an interested buyer. Therefore, a transporter is needed to deliver the packaged item between the seller and the buyer. At the beginning of the contract, the seller, transporter and buyer are all required to sign the attested terms and conditions agreement form. The agreed upon collateral which is twice the item price is automatically deducted from each party that signs the form. The form's IPFS hash is available in the contract so that each of the seller, buyer and transporter will be able to access the same form using the IPFS hash provided in the contract. This will ensure the data integrity of the content of the agreement form is well maintained.

The transporter is provided with a key along with the physical item. The transporter's key (Key_T) is handed over to the transporter while the key is kept unrevealed to the buyer. On the other hand, the buyer also has a key which the transporter is unaware of its content, (Key_B). The keys are exchanged between the transporter and the buyer upon the successful delivery of the item. The hashes of the concatenated strings is computed twice, once by the transporter and another time by the receiver. Verification is then done by the contract to check that the hashes match. If the hashes match, then the transaction is successful and the buyer gets back half of the deposited collateral. The transporter also gets back the deposited collateral in addition to the transportation fees and the rest goes to the seller. The above is demonstrated in Alternative 1, in Fig. 3. Alternative 2 and 3 that are illustrated in Fig. 3, take place if the verification hashes do not match or any of the transporter or receiver fail to enter the verification keys on time. In this case, the deposited collateral gets transferred to the trusted arbitrator that was agreed upon by all the parties at the beginning of the contract.

It is important to note that the contract code is written in a way that allocates a time window for each action to preserve the rights of the other parties involved. For instance, when the packaged item is received by the transporter, the time is stored in the contract and is later checked to verify if the delivery time was exceeded or not. The buyer has the right to request a refund if the time is exceeded. This can be done using the `refund()` function.

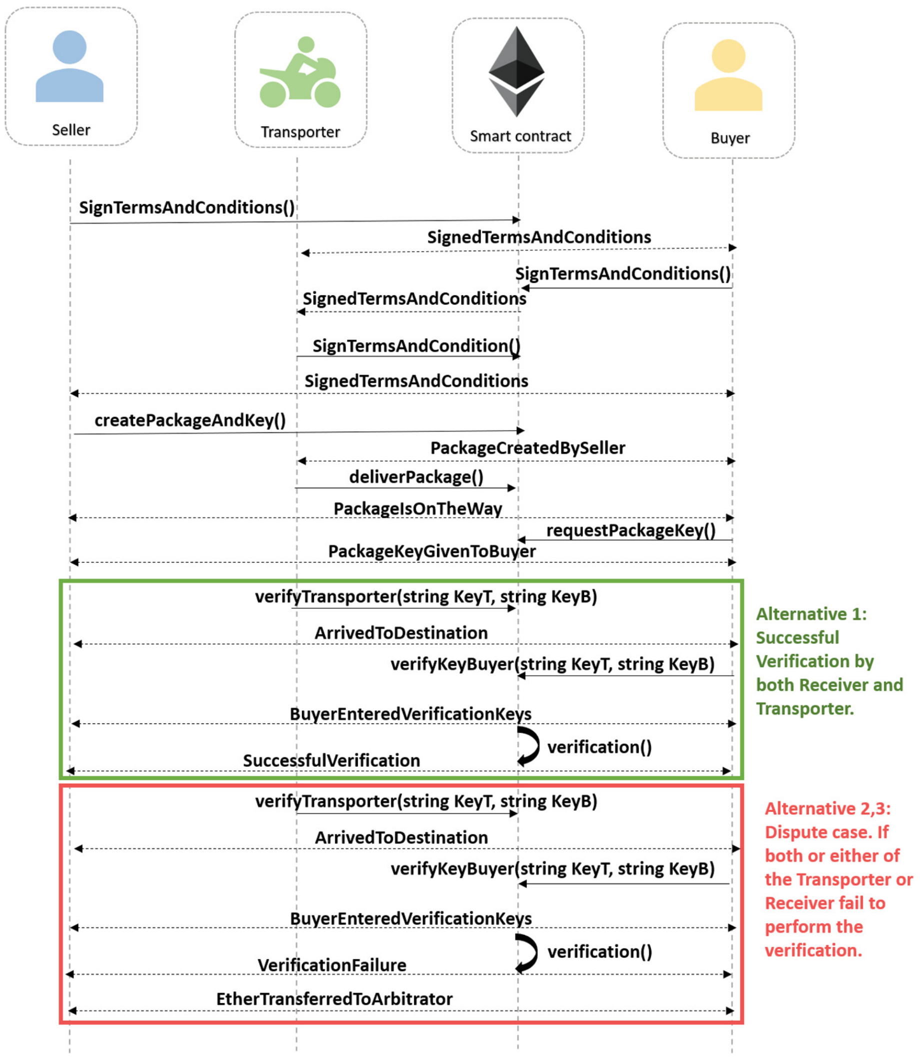
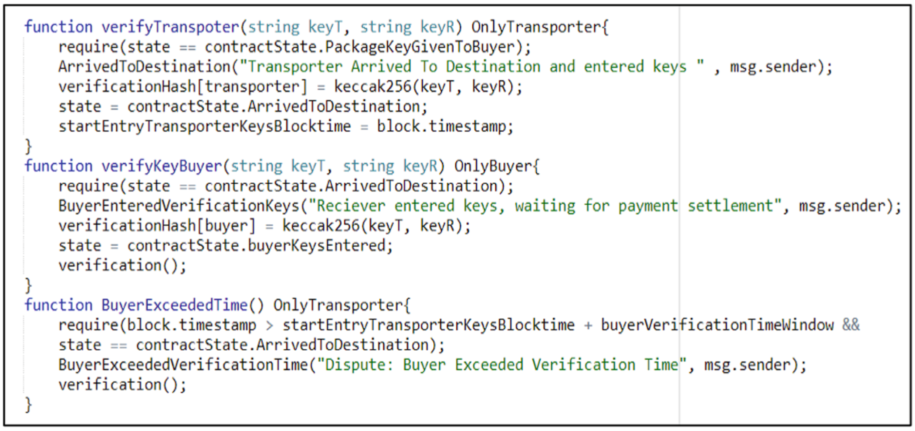

Seller
Transporter
Smart contract
Buyer
SignTermsAndConditions()
SignedTermsAndConditions
SignTermsAndConditions()
SignedTermsAndConditions
SignTermsAndCondition()
SignedTermsAndConditions
createPackageAndKey()
PackageCreatedBySeller
deliverPackage()
PackageIsOnTheWay
requestPackageKey()
PackageKeyGivenToBuyer
verifyTransporter(string KeyT, string KeyB)
ArrivedToDestination
verifyKeyBuyer(string KeyT, string KeyB)
BuyerEnteredVerificationKeys
verification()
SuccessfulVerification
Alternative 1: Successful Verification by both Receiver and Transporter.
verifyTransporter(string KeyT, string KeyB)
ArrivedToDestination
verifyKeyBuyer(string KeyT, string KeyB)
BuyerEnteredVerificationKeys
verification()
VerificationFailure
EtherTransferredToArbitrator
Alternative 2,3: Dispute case. If both or either of the Transporter or Receiver fail to perform the verification.

Fig. 3. Sequence diagram of the smart contract code that shows the flow for a successful and an unsuccessful transaction

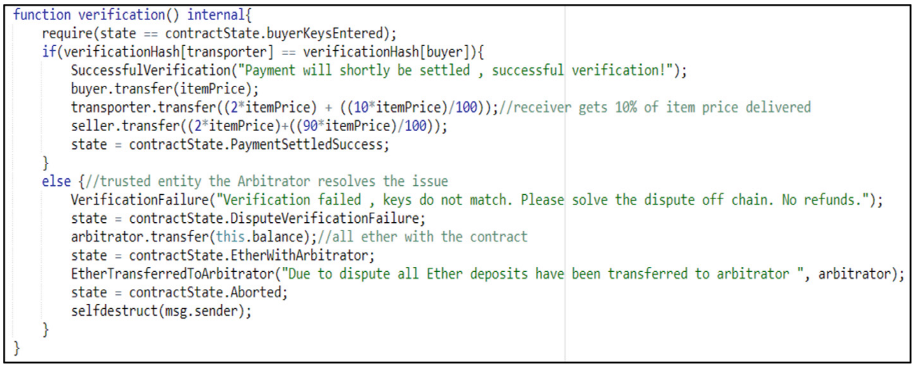
Furthermore, the buyer also needs to enter the keys for verification after the transporter has called the `verifyTrasnporter()` function by a maximum of 15 min. If the buyer exceeds the time window allocated for verifying the keys, the transporter has the right to leave after calling the `BuyerExceededTime()` function. In this function, the time window is checked, and the arbitrator gets involved to solve the issue off the chain. Figure 4, shows the implementation of the verification functions of both the transporter and buyer.



```
function verifyTranspoter(string keyT, string keyR) OnlyTransporter{
    require(state == contractState.PackageKeyGivenToBuyer);
    ArrivedToDestination("Transporter Arrived To Destination and entered keys " , msg.sender);
    verificationHash[transporter] = keccak256(keyT, keyR);
    state = contractState.ArrivedToDestination;
    startEntryTransporterKeysBlocktime = block.timestamp;
}
function verifyKeyBuyer(string keyT, string keyR) OnlyBuyer{
    require(state == contractState.ArrivedToDestination);
    BuyerEnteredVerificationKeys("Reciever entered keys, waiting for payment settlement", msg.sender);
    verificationHash[buyer] = keccak256(keyT, keyR);
    state = contractState.buyerKeysEntered;
    verification();
}
function BuyerExceededTime() OnlyTransporter{
    require(block.timestamp > startEntryTransporterKeysBlocktime + buyerVerificationTimeWindow &&
    state == contractState.ArrivedToDestination);
    BuyerExceededVerificationTime("Dispute: Buyer Exceeded Verification Time", msg.sender);
    verification();
}
```

Fig. 4. Contract code showing the key verification functions and time window checking (The full code is available at: https://github.com/smartcontract694/POD_PhysicalItems).

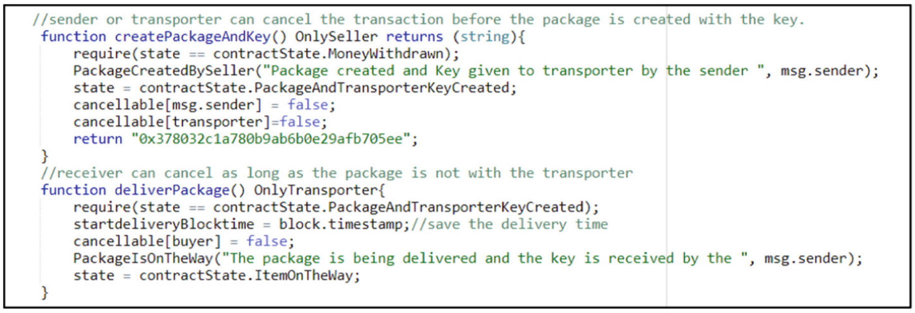
Once the buyer enters the keys, the `verification()` function is automatically called as it is an internal function. The function compares the hashes of the transporter and buyer. If the hashes are equal, the verification is successful and the buyer receives half of the deposited collateral, the transporter receives his full collateral in addition to his transportation fees and the seller receives back the rest of the deposits. However, if the hashes are not equal, which could mean that either of the transporter or receiver failed to enter the right keys, all deposits are transferred to the arbitrator and the dispute is solved off the chain. This is demonstrated in Fig. 5 which shows the code of the `verification()` function.



```
function verification() internal{
    require(state == contractState.buyerKeysEntered);
    if(verificationHash[transporter] == verificationHash[buyer]){
        SuccessfulVerification("Payment will shortly be settled , successful verification!");
        buyer.transfer(itemPrice);
        transporter.transfer((2*itemPrice) + ((10*itemPrice)/100));//receiver gets 10% of item price delivered
        seller.transfer((2*itemPrice)+((90*itemPrice)/100));
        state = contractState.PaymentSettledSuccess;
    }
    else {//trusted entity the Arbitrator resolves the issue
        VerificationFailure("Verification failed , keys do not match. Please solve the dispute off chain. No refunds.");
        state = contractState.DisputeVerificationFailure;
        arbitrator.transfer(this.balance);//all ether with the contract
        state = contractState.EtherWithArbitrator;
        EtherTransferredToArbitrator("Due to dispute all Ether deposits have been transferred to arbitrator ", arbitrator);
        state = contractState.Aborted;
        selfdestruct(msg.sender);
    }
}
```

Fig. 5. Contract code showing the verification function (The full code is available at: https://github.com/smartcontract694/POD_PhysicalItems).

Furthermore, each function in the contract requires a certain previous state for it to be successfully executed. This plays a vital role in the ability of a certain entity to cancel the purchase. The buyer can cancel the purchase without penalty if the item has not yet been delivered and a transporter can cancel the delivery before taking the item from the seller. A mapping that maps the address of each of the buyer, seller and transporter along with a boolean is used to control the cancellation. At the beginning all items of the mapping are initialized with true. After the package is created and the key$_T$ is handed to the transporter, the transporter and seller cannot cancel the purchase. Hence, their mapping items are changed to false as illustrated in Fig. 6. Moreover, when the package has been handed over to the transporter, the buyer cannot cancel the purchase and the mapping of the buyer also gets changed to false. Hence, the cancellation function always checks the state to know which stage the item is in and if the cancelling entity has the right to cancel or not using the mapping details.



```
//sender or transporter can cancel the transaction before the package is created with the key.
 function createPackageAndKey() OnlySeller returns (string){
     require(state == contractState.MoneyWithdrawn);
     PackageCreatedBySeller("Package created and Key given to transporter by the sender ", msg.sender);
     state = contractState.PackageAndTransporterKeyCreated;
     cancellable[msg.sender] = false;
     cancellable[transporter]=false;
     return "0x378032c1a780b9ab6b0e29afb705ee";
 }
 //receiver can cancel as long as the package is not with the transporter
 function deliverPackage() OnlyTransporter{
     require(state == contractState.PackageAndTransporterKeyCreated);
     startdeliveryBlocktime = block.timestamp;//save the delivery time
     cancellable[buyer] = false;
     PackageIsOnTheWay("The package is being delivered and the key is received by the ", msg.sender);
     state = contractState.ItemOnTheWay;
 }
```

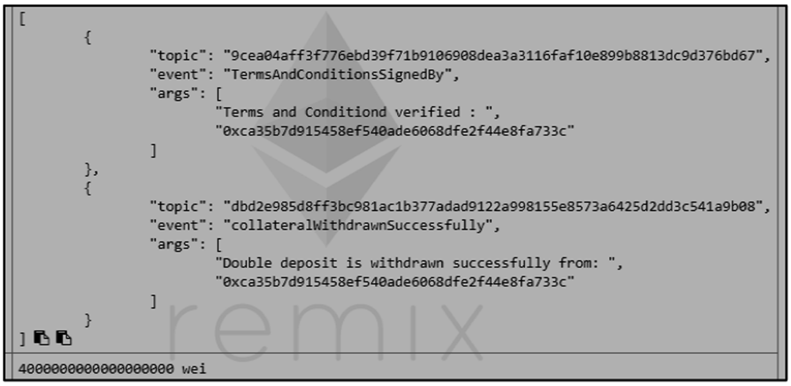
Fig. 6. Functions in the contract code that show 'state' requirements (The full code is available at: https://github.com/smartcontract694/POD_PhysicalItems).

5 Testing and Validation

The smart contract code has been tested for several important aspects and test cases that are discussed in this section.

5.1 Test Case 1: Payable Collateral Amount

The contract code has one payable function. This function uses a modifier called `costs()` which ensures that the ether deposited is equal to the agreed upon collateral which is twice the item price. Figure 7 shows the logs of a successful deposit withdrawal made by the sender upon signing the terms and conditions agreement form. The figure also shows that the successful transaction took a value of 4 ether since the item price is 2 ether in this contract.

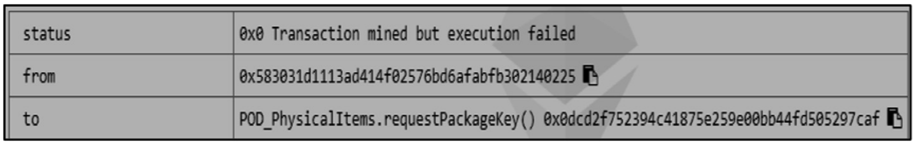


```
[
	{
		"topic": "9cea04aff3f776ebd39f71b9106908dea3a3116faf10e899b8813dc9d376bd67",
		"event": "TermsAndConditionsSignedBy",
		"args": [
			"Terms and Conditiond verified : ",
			"0xca35b7d915458ef540ade6068dfe2f44e8fa733c"
		]
	},
	{
		"topic": "dbd2e985d8ff3bc981ac1b377adad9122a998155e8573a6425d2dd3c541a9b08",
		"event": "collateralWithdrawnSuccessfully",
		"args": [
			"Double deposit is withdrawn successfully from: ",
			"0xca35b7d915458ef540ade6068dfe2f44e8fa733c"
		]
	}
]
4000000000000000000 wei
```

Fig. 7. Log details of a successful deposit transaction by the seller.

5.2 Test Case 2: Role Restriction

To ensure the proper functionality of the code based on the actions of the participating entities, the contract's functions are restricted based on the role. All the functions of the contract have been tested successfully for role restriction. Hence, if the arbitrator for instance whose address is "0x583031d1113ad414f02576bd6afabfb302140225" tries to request the package key of the buyer, the transaction fails as illustrated in Fig. 8.

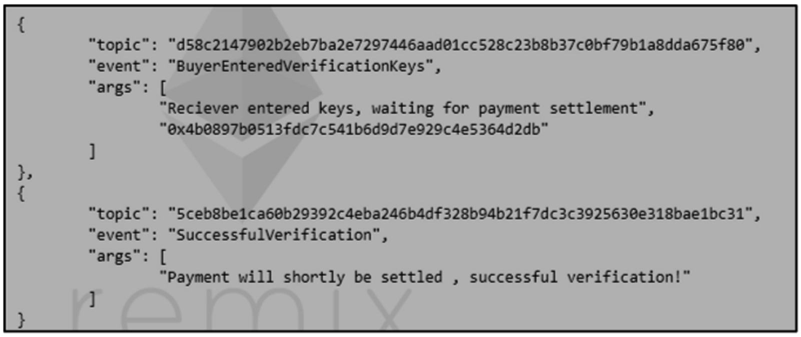


status	0x0 Transaction mined but execution failed
from	0x583031d1113ad414f02576bd6afabfb302140225
to	POD_PhysicalItems.requestPackageKey() 0x0dcd2f752394c41875e259e00bb44fd505297caf

Fig. 8. Log details of a failed transaction due to role restriction.

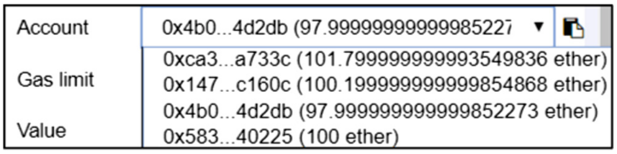
5.3 Test Case 3: Matching Verification Keys

A successful test case was tested to ensure that the verification of the hashes works as expected and the payment settlement. As can be seen in Fig. 9, the keys entered provide similar hashes and as a result, the transaction is successful. The buyer gets back a deposit of 2 ether, the transporter gets back 2 ether in addition to 10% of the item price as transportation fees and the seller gets back the rest of the deposited collateral. Figure 10 shows the ether deposits at the end of the successful transaction, with the seller, transporter and buyer having 101.8, 100.2 and 98 ether respectively.



```
{
        "topic": "d58c2147902b2eb7ba2e7297446aad01cc528c23b8b37c0bf79b1a8dda675f80",
        "event": "BuyerEnteredVerificationKeys",
        "args": [
                "Reciever entered keys, waiting for payment settlement",
                "0x4b0897b0513fdc7c541b6d9d7e929c4e5364d2db"
        ]
},
{
        "topic": "5ceb8be1ca60b29392c4eba246b4df328b94b21f7dc3c3925630e318bae1bc31",
        "event": "SuccessfulVerification",
        "args": [
                "Payment will shortly be settled , successful verification!"
        ]
}
```

Fig. 9. Log details of a successful transaction and payment settlement.

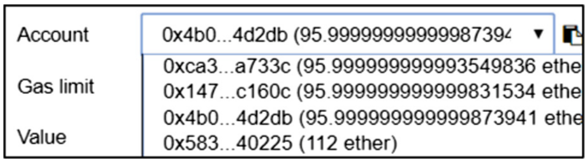


```
Account      0x4b0...4d2db (97.99999999999985227
             0xca3...a733c (101.799999999993549836 ether)
Gas limit    0x147...c160c (100.199999999999854868 ether)
             0x4b0...4d2db (97.999999999999852273 ether)
Value        0x583...40225 (100 ether)
```

Fig. 10. Ether deposits at the end of a successful transaction.

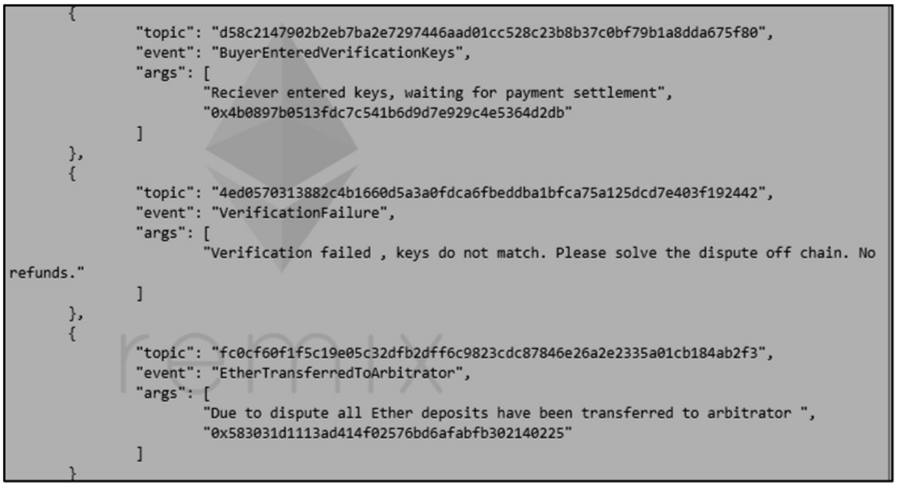
5.4 Test Case 4: Dispute and Arbitrator Role

In the case of the keys entered by the buyer and transporter are not matching, a dispute occurs and the arbitrator steps in. All the ether deposits that are held with the contract are transferred to the arbitrator and the dispute is solved off the chain. As can be seen in Fig. 11, the arbitrator at the end of an unsuccessful transaction has 12 ether, since each of the three parties has deposited 4 ethers when signing the agreement form. Figure 12, shows the events that take place during a dispute and before the contract gets to the ‘aborted’ state.



```
Account      0x4b0...4d2db (95.99999999999987394
             0xca3...a733c (95.999999999993549836 ethe
Gas limit    0x147...c160c (95.999999999999831534 ethe
             0x4b0...4d2db (95.999999999999873941 ethe
Value        0x583...40225 (112 ether)
```

Fig. 11. Ether deposits at the end of an unsuccessful transaction.



```
        {
                "topic": "d58c2147902b2eb7ba2e7297446aad01cc528c23b8b37c0bf79b1a8dda675f80",
                "event": "BuyerEnteredVerificationKeys",
                "args": [
                        "Reciever entered keys, waiting for payment settlement",
                        "0x4b0897b0513fdc7c541b6d9d7e929c4e5364d2db"
                ]
        },
        {
                "topic": "4ed0570313882c4b1660d5a3a0fdca6fbeddba1bfca75a125dcd7e403f192442",
                "event": "VerificationFailure",
                "args": [
                        "Verification failed , keys do not match. Please solve the dispute off chain. No
refunds."
                ]
        },
        {
                "topic": "fc0cf60f1f5c19e05c32dfb2dff6c9823cdc87846e26a2e2335a01cb184ab2f3",
                "event": "EtherTransferredToArbitrator",
                "args": [
                        "Due to dispute all Ether deposits have been transferred to arbitrator ",
                        "0x583031d1113ad414f02576bd6afabfb302140225"
                ]
        }
```

Fig. 12. Events at the end of an unsuccessful transaction.

6 Conclusion

This paper has presented a blockchain solution which facilitates the trading and tracking of sold items between two parties in a decentralized way. The solution provides a proof of delivery of physical items taking advantage of the security and immutability that blockchain provides. Our proposed solution is generic enough and can be applied to almost all shipped physical items and assets. In this paper, we focused on providing, implementing, and testing the smart contract code and algorithm of the PoD solution that show cases the ability to prove the delivery of an item using an equal deposited collateral by the seller, transporter and buyer. In the paper, we showed and discussed how our solution can provide PoD key features and requirement that include integrity, accountability, authorization, punctuality and honesty. As a future work, we plan to extend our solution to implement aspects related to confidentiality and privacy. Also, work is underway to develop completed DApps with different views for seller, buyer, and transporter.

References

1. "Help & Customer Service", Amazon Shipment Updates via Text Terms and Conditions. https://www.amazon.com/gp/help/customer/display.html/ref=hp_left_v4_sib?ie=UTF8&nodeId=201910790
2. "Help & Customer Service", Shipping Carrier Contacts. https://www.amazon.com/gp/help/customer/display.html/ref=hp_ss_qs_v3_rt_ci?ie=UTF8&nodeId=201117350
3. Toyoda, K., et al.: A novel blockchain-based product ownership management system (POMS) for anti-counterfeits in the post supply chain. IEEE Access **PP**(99), 1 (2017)
4. Christidis, K., Devetsikiotis, M.: Blockchains and smart contracts for the internet of things. IEEE Access **4**, 2292–2303 (2016)

5. “Two party contracts”, Dapps for Beginners (2018). https://dappsforbeginners.wordpress.com/tutorials/two-party-contracts/
6. “How Our Escrow Smart Contract Works”, localethereum.com’s official blog (2018). https://blog.localethereum.com/how-our-escrow-smart-contract-works/
7. “Double Deposit Escrow – BitBay”, BitBay (2018). https://bitbay.market/double-deposit-escrow. Accessed 28 Mar 2018
8. “Open Source Products”, KYC (Know Your Customer) (2018). https://guide.blockchain.z.com/en/docs/oss/kyc/

Towards Legally Enforceable Smart Contracts

Dhiren Patel[1], Keivan Shah[1], Sanket Shanbhag[1], and Vasu Mistry[2(✉)]

[1] Veermata Jijabai Technological Institute, Mumbai 400019, India
[2] National Institute of Technology, Surat 395007, India
vasu5235@gmail.com

Abstract. A smart contract is a computer program that is stored and executed on a decentralized system such as a Blockchain. At present, smart contracts have a unique value proposition but cannot be enforced in some of the existing judicial frameworks. In this paper, we propose a framework to create and execute legally binding smart contracts. We experimented with a Distributed Outsourcing Developer Marketplace aka Freelancer application use case deployed on Ethereum Blockchain. Our findings are useful in the sense that as per respective national legal frameworks, smart contracts can be made legally enforceable by incorporating crypto primitive like digital signature.

Keywords: Blockchain and distributed ledger technology · Distributed software · Smart contract

1 Introduction

Blockchains are distributed digital ledgers of cryptographically signed transactions that are grouped into blocks. Each block is linked to the previous one after validation and consensus of all participating nodes. As new blocks are added, older blocks become more difficult to modify. New blocks are replicated across all copies of the ledger within the network, and any conflicts are resolved automatically using established rules [1].

A transaction on the blockchain is a digitally signed item broadcast to the P2P network of a blockchain. A transaction can be signed by one or more entities (multi-signature). In cryptocurrency, a standard transaction specifies sending tokens from one account to another. A transaction fee is a nominal amount paid to have a valid transaction verified and written in a block. A wallet contains every necessary information to generate the owner of public key(s), which is sufficient to transfer assets of the owner of the wallet in a blockchain and display the content of the associated account [2].

A smart contract is a computer program that is stored and executed on a decentralized system e.g. a blockchain. A smart contract can perform calculations, store information, and automatically send funds to other accounts. Thus, smart contracts can be seen as automating the marketplace system and allowing different parties to work without mutual trust between each other.

S. Chen et al. (Eds.): ICBC 2018, LNCS 10974, pp. 153–165, 2018.
https://doi.org/10.1007/978-3-319-94478-4_11

Smart contracts are a new form of pre-emptive self-help that should not be discouraged by the legislatures or courts [3]. A smart contract may give rise to legally enforceable obligations. These issues are treated differently from country to country. Meanwhile, at this time, different jurisdictions are grappling with, endorsing, or revoking different legislative provisions to regulate the use of DLT (Distributed Ledger Technology) systems in different contexts. For example, smart contracts that underpin transactions in ICOs (Initial Coin Offerings – e.g. KodakCoin) may be completely illegal in some jurisdictions, while a smart contract that handles intra-institutional banking and other financial transactions may be quite legal, in the same jurisdiction or elsewhere [4].

There are those who promote the "code is contract" approach (that is, that the entirety of a natural language contract can be encoded). On the other hand, there are those who see smart contracts as black boxes consisting of digitising performance of business logic (e.g. payment), which may or may not be associated with a natural language contract. In between these two extremes, a number of permutations are likely to emerge including, a "split" smart contract model under which natural language contract terms are connected to computer code via parameters that feed into computer systems for execution. Also, legally binding contractual effect depends on a number of variables. It is tempting to conclude that, just because the moniker "smart contract" includes the word contract, it is a legally binding contract as a matter of law. This is not necessarily correct [5].

In this paper, we propose a framework to create and execute legally binding smart-contracts with a validation use-case of classical Freelancer application.

Rest of the paper is organized as follows: Sect. 2 discusses motivation and background of Blockchain, Smart Contract and its legal standing and discusses relationship between programmable contract and legal contract. In Sect. 3, we take use case of distributed outsourcing developer market place and discuss smart contract implementation in detail. In Sect. 4, we propose directives for how smart contract can be legally enforced using the same application using few cryptographic primitives, with conclusions and references at the end.

2 Relationship Between Programmable Smart Contracts and Legal Contracts: Motivation and Background

Blockchains are immutable digital ledger systems implemented in a distributed fashion (i.e. without a central repository) and usually without a central authority. At their most basic level, they enable a community of users to record transactions in a ledger that is public to that community, such that no transaction can be changed once published [1].

A block is an individual unit of a blockchain, composed of a collection of transactions and a block header. A block header keeps a collection of metadata about the block that contains a hash-value of its parent in the blockchain, and a hash of the aforementioned metadata and the data of the block itself [2].

The term "Smart Contracts" has existed for more than two decades – A set of promises, specified in digital form, including protocols within which the parties perform on the other promises [6]. However, in 2014, Vitalik Buterin invented a new generation of smart contracts: decentralized and immutable once it exists in DLT systems [7].

The endeavor of standards in the context of smart contracts, is to consider how contracts are written, how they are enforced, and how to ensure that the automated performance of a contract is faithful to the meaning of any relevant legal documentation. A smart contract is an automatable and enforceable agreement. Automatable by computer, although some parts may require human input and control. Enforceable either by legal enforcement of rights and obligations or via tamper-proof execution of computer code [8] (Fig. 1).

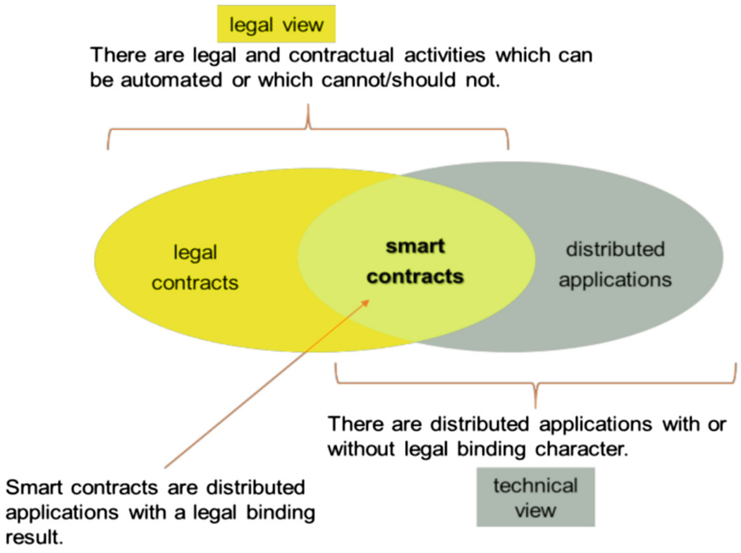

legal view
There are legal and contractual activities which can be automated or which cannot/should not.
legal contracts
smart contracts
distributed applications
There are distributed applications with or without legal binding character.
Smart contracts are distributed applications with a legal binding result.
technical view

Fig. 1. Relationship between legal and technical view [4]

Some smart contracts can be very simple, like putting a timestamp on a transaction while some are more complex and require the formal agreements of parties beforehand. In this case, they can be, depending on the case, considered as a legal contract. Smart contracts may also be legal contracts when they are enforceable by traditional legal methods. Contracts that do not require complex statements of terms and conditions or adjectives and adverbs to describe what is reasonable are better suited to automation than contracts that require complex legal terms to explain the nature of the parties' obligations [8]. As articulated by Farrell et al. – "It follows from this that if smart contracts are to be used meaningfully in commercial contracts then they will need to be blends of both coded and natural language terms" [9].

This delineation between types of contractual arrangement that may or may not be suited to automation in a smart contract will be useful when deciding whether it is appropriate or necessary to apply standards to DLT systems –based smart contracts, and if so, what that standards will look like [8].

3 Use Case: Distributed Outsourcing Developer Marketplace

The Distributed Outsourcing Developer Marketplace is a freelancing system. A client posts a particular project needed to be made by a developer. The developer who accepts this project is known as a freelancer. A freelancer or freelance worker is a term commonly used for a person who is self-employed and may not necessarily committed to a particular employer long-term. Freelance workers are sometimes represented by a company or a temporary agency that resells freelance labor to clients; others work independently or use professional associations or websites to get work [10].

Current freelancing systems are provided by third parties and rely on trust by both the freelancer and the client on this central third party. Using a Blockchain based system can completely remove this element of trust on a central entity. This will help in eliminating hefty commissions and replace it with smart contract deployment and update costs which are often significantly cheaper. Also, being decentralized, this system is completely unbiased and no single entity can monopolize the system under normal circumstances. Using a cryptocurrency such as ether leads to immensely faster processing times and quick fund transfers.

We have implemented a freelancing system using Solidity Smart Contracts deployed on Ethereum. The Smart Contract defines the milestones and the payment for each milestone which would be sent as ether. The contract is deployed with the client, freelancer and the milestone details. In the present format, this contract cannot be legally enforced since there is no contract in legal phrasing available.

3.1 Freelancer System

The first version of our project was based on the popular website freelancer.com [12]. Following terms are useful in understanding our proposal:

- Client: The person/entity posting the project to be completed
- Freelancer: The developer who takes a project posted by a client and gets paid to do it.
- Milestone: Smaller sub-tasks to be completed leading to partial realization of the amount.
- Smart-contract: The software code deployed on the blockchain which is running the distributed marketplace.

The workflow was as follows:

– The client posts his project online, with the task he wants to be completed, broken down into smaller tasks called "Milestones" [13]. The client then defines a stipulated amount of money to be paid to the freelancer on the completion of each milestone (in ether) along with their deadlines. With this information a smart contract is deployed on the blockchain by the client, containing information about the amount of each milestone, the deadlines. The client has to pay the cost of all the milestones upfront as well as the gas needed to deploy the contract.

– This contract now acts as an escrow account which holds the funds until completion of any milestone.
– Once the contract is deployed on the blockchain, this task is then advertised to all freelancers on the system. The freelancer can choose to accept the task using an API call to the smart contract.
– Upon completion of a milestone, the freelancer uses an API call to change the state of the smart contract and upon confirmation by the client, the stipulated amount is automatically paid to the freelancer by the smart contract.
– Our system also supports features like canceling any milestone and refunding the amount back to the client on a failed task.

This system is depicted in Fig. 2.

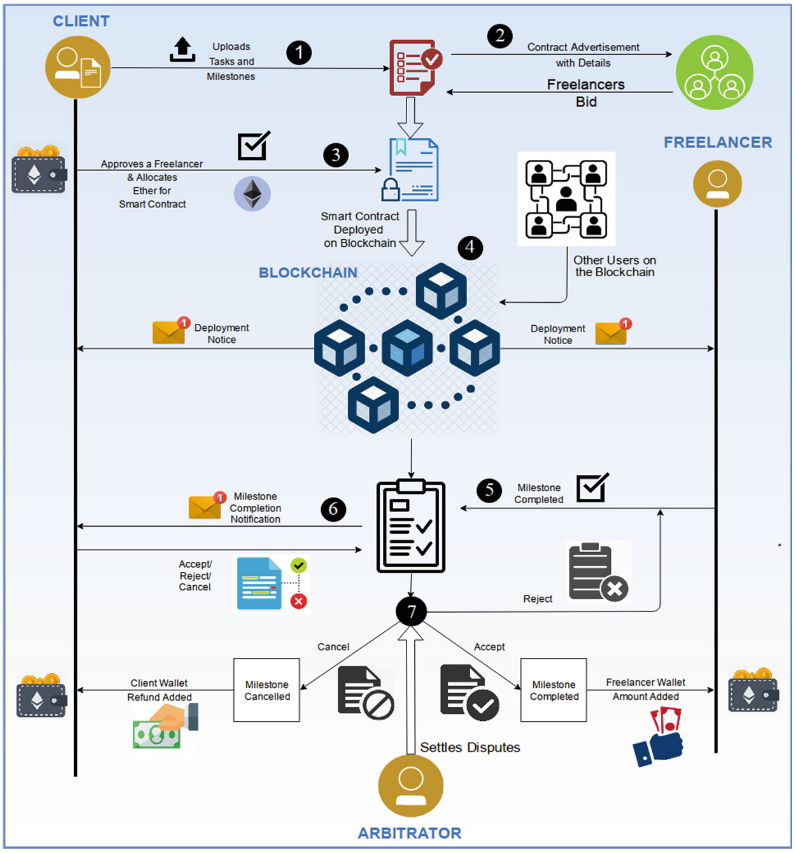

CLIENT
Uploads Tasks and Milestones
1
2
Contract Advertisement with Details
Freelancers Bid
FREELANCER
Approves a Freelancer & Allocates Ether for Smart Contract
3
Smart Contract Deployed on Blockchain
4
Other Users on the Blockchain
BLOCKCHAIN
Deployment Notice
Deployment Notice
Milestone Completion Notification
6
5
Milestone Completed
Accept/ Reject/ Cancel
Reject
7
Cancel
Accept
Client Wallet Refund Added
Milestone Cancelled
Milestone Completed
Freelancer Wallet Amount Added
Settles Disputes
ARBITRATOR

Fig. 2. Use case ‘Freelancer’: implementation workflow on Ethereum blockchain

The Basic structure of the Freelancer Smart Contract is as follows:
Attributes.

1. Client Address: This is the public key (address) of the client on the Ethereum Blockchain. For all purposes this is the identity of the client that is used by the blockchain.
2. Freelancer Address: This is the public key (address) of the freelancer on the Ethereum Blockchain. For all purposes this is the identity of the freelancer that is used by the blockchain.
3. Task Description: This is the link to the plain text description of the entire task along with the various milestones and amount as laid out by the Client.
4. Review Time: It is the time that the client is given to review a submission made by the freelancer after which the freelancer can collect the payment is the client fails to review.
5. Milestones: This are the various milestones of the outsourced task. Each of the milestone contains the following details.
   a. Amount: The amount in ether that is to be paid out to the freelancer on proper completion of the current milestone.
   b. Deadline: The time by which the freelancer is expected to complete the task.
   c. Status: The current status of the milestone. It has various possible states such as Completed, Canceled.

Functions.

1. Milestone Complete:
   The function that the freelancer calls to mark a milestone that he has done as complete. After this the milestone work is reviewed by the client.
2. Milestone Review:
   The function that the client calls to review the Milestone and mark it as complete, rejected or canceled.

4 Towards Legally Bound Smart Contracts

We demonstrated Freelancer use case implementation on Ethereum Blockchain (as depicted in Fig. 2) at the "Smart contract with Blockchain and IoT workshop" held at VJTI Mumbai in Feb 2018. And after feedback discussions and deliberations, we found that a smart contract cannot be upheld in the court of law as a valid legal document. This is a major concern as any dispute between the client and the freelancer cannot be upheld in the court of law. Even though the code on a smart contract is completely correct and immutable, it still cannot be treated as a legal document. Also, it is difficult to debate the technical details of a smart contract as lawyers and judges are usually not familiar with advanced cryptographic algorithms and their correctness.

Some important terminology:

- Legal-contract: The legal equivalent of the smart contract. It contains the signatories, the milestone and the payment schedule in judicial parlance.
- Template Legal-contract: This is a pseudo legal contract constructed with blank fields where changing parameters like client, freelancer details and milestone details will be filled in automatically.
- IPFS: Interplanetary File System, is a peer-to-peer distributed file system that seeks to connect all computing devices with the same system of files. In some ways, IPFS is similar to the World Wide Web, but IPFS could be seen as a single BitTorrent swarm, exchanging objects within one Git repository. In other words, IPFS provides a high-throughput, content-addressed block storage model, with content-addressed hyperlinks [11].

To incorporate a legal framework in our system, we modified our original system as follows:

Once the client and freelancer are chosen on the website, they are given a form, where they simply have to fill out their legally binding details. Using these details, a legally binding contract (written in plain text) is drafted between the client and freelancer which contains clauses for each milestone along with their deadlines and amounts. This is done automatically by dynamically creating clauses which are appended to a base template contract. For different use-cases, a more advanced template engine maybe required. In case of dispute, the wordings of the legal contract shall be treated as final. Thus, the base template contract needs to be carefully drafted by a professional to ensure its infallibility.

The legal contract is then stored on a publicly available distributed file-system e.g. IPFS. We have chosen to utilize IPFS because the design of IPFS ensures immutability and non-reliance on a central server [11].

The client then deploys the smart contract on Ethereum. He has to pay all the milestone fees upfront. The hash of the legal contract is included in the constructor when the smart contract is deployed. This hash field on the smart contract is constant once set and cannot be changed. This acts as a link between the smart contract and the legal document.

Now, both parties fetch the document from IPFS and sign it with their digital certificates and upload this to IPFS. The hash of their signed document is then added to the smart contract via an API call which can only be invoked by the client and freelancer. The smart contract then begins execution only after legal document is signed and uploaded by both the client and the freelancer. This ensures that both parties have verified and signed a legal contract in addition to accepting the smart contract code.

This modified implementation (incremental portion of legal binding) is depicted in Fig. 3.

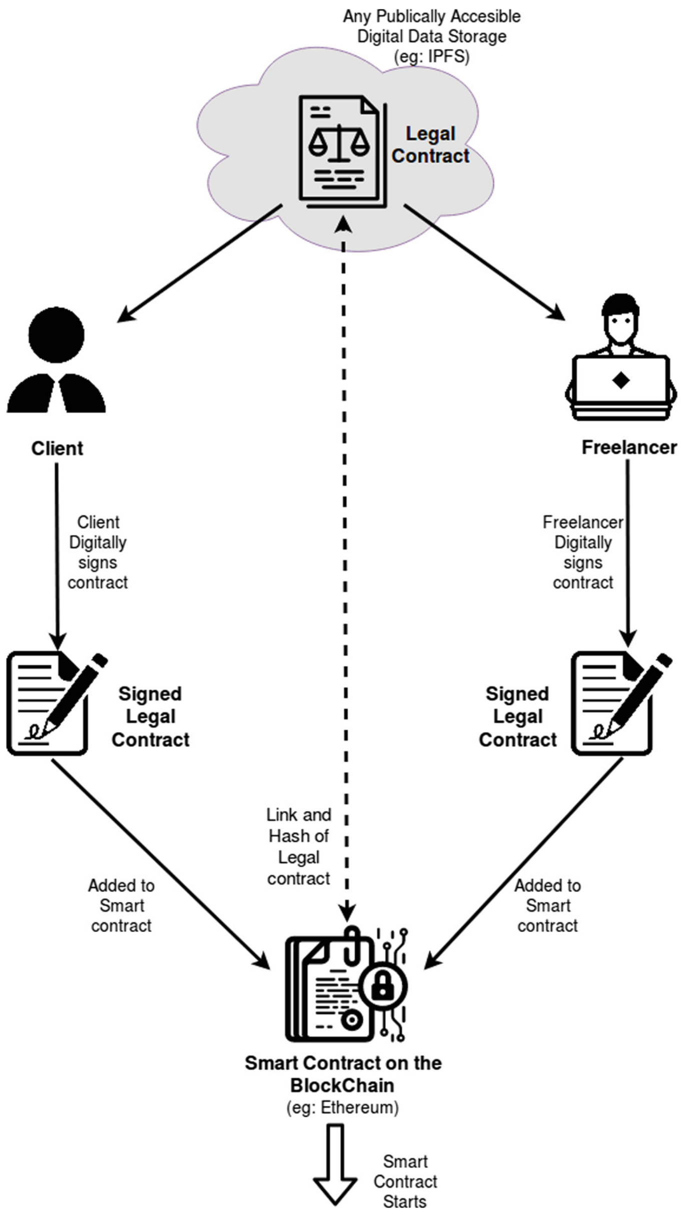

Any Publically Accesible Digital Data Storage (eg: IPFS)
Legal Contract
Client
Freelancer
Client Digitally signs contract
Freelancer Digitally signs contract
Signed Legal Contract
Signed Legal Contract
Link and Hash of Legal contract
Added to Smart contract
Added to Smart contract
Smart Contract on the BlockChain
(eg: Ethereum)
Smart Contract Starts

Fig. 3. Binding legality to smart contract

As depicted in Figs. 2 and 3, the Legal Contracts defined are enforceable because of the usage of digital-signatures which is a cryptographic method used to validate/sign documents in digital format.

The contract is deployed by the client with the appropriate task and participant details (Pseudocode for the same is given at the end of the section Figs. 4 and 5). Once the contract is deployed, the smart contract sets all global variables, but does not begin execution. Both the client and freelancer hash the legal document using their corresponding digital keys and call the `signContract(string signedData)` function with the hash. This function sets the corresponding global variables and marks the signed status as true. Once both the client and freelancer have signed the contract, it begins execution.

After completing a single milestone, the freelancer sends the completed work to the client and then calls `the markMilestoneComplete(uint id)` with the id of the milestone he has completed. This function makes appropriate checks on the id, status and deadline of the milestone and if everything is correct, sets the milestone status to 'Completed'. This change can be seen by the client on the blockchain without using any gas. If he finds the work satisfactory, the client calls the `reviewMilestone (id, Review)` function which sets the status of the milestone as per the clients review and sends the payment of the milestone to the freelancer if the client accepts the milestone.

Depending on the country this system may or may not be legally accepted. Thus we introduced the term verified signatures to mean signatures verified to that extent such that it is legally valid. We examined the situation of such digital signatures in India and how to enforce their legal validity [14].

In relation to a digital signature, electronic record or public key, with its grammatical variations and cognate expressions means to determine whether

- The initial electronic record was affixed with the digital signature by the use of private key corresponding to the public key of the subscriber;
- The initial electronic record is retained intact or has been altered since such electronic record was so affixed with the digital signature.

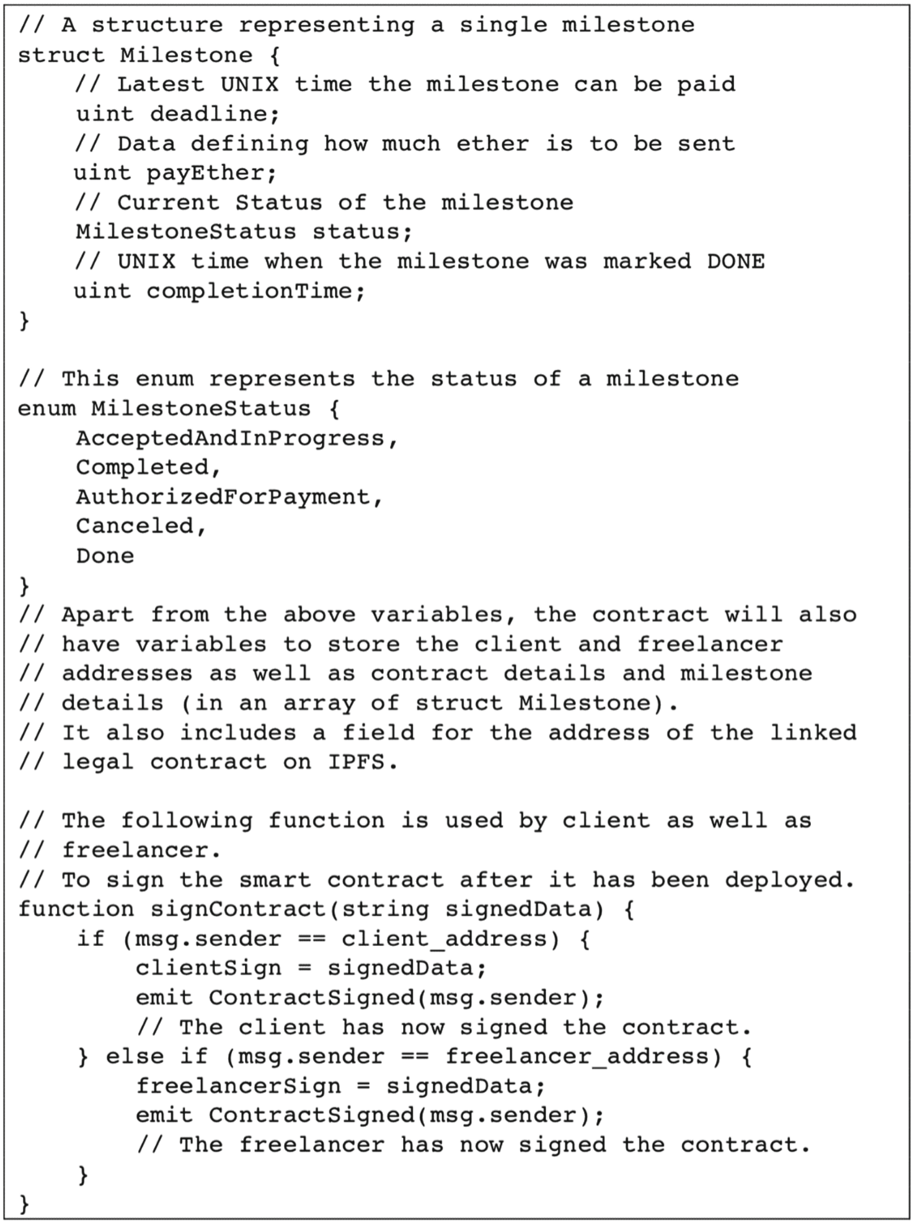
From the IT Bill-2000 [14] "Where any law provides that information or any other matter shall be authenticated by affixing the signature or any document shall be signed or bear the signature of any Person then, notwithstanding anything contained in such law, such requirement shall be Deemed to have been satisfied, if such information or matter is authenticated by means of Digital signature affixed in such manner as may be prescribed by the Central Government."

In simpler terms, any digital signature stands to be deemed verified and legally valid only when a digital certificate is issued by a Certifying Authority. There are provisions for new parties to apply and be approved as a certificate authority.

To resolve the issues of validity of digital signatures in the Indian context, we assume that there is a certified company (Certifying Authority) which can issue Digital Certificates for use on Ethereum network using the same Ethereum wallet public/private key pair for the Digital Certificate.

Considering above context, we introduce necessary provisions to make smart contract legally enforceable.

Pseudocode for legally enforceable smart contract.



```
// A structure representing a single milestone
struct Milestone {
    // Latest UNIX time the milestone can be paid
    uint deadline;
    // Data defining how much ether is to be sent
    uint payEther;
    // Current Status of the milestone
    MilestoneStatus status;
    // UNIX time when the milestone was marked DONE
    uint completionTime;
}

// This enum represents the status of a milestone
enum MilestoneStatus {
    AcceptedAndInProgress,
    Completed,
    AuthorizedForPayment,
    Canceled,
    Done
}
// Apart from the above variables, the contract will also
// have variables to store the client and freelancer
// addresses as well as contract details and milestone
// details (in an array of struct Milestone).
// It also includes a field for the address of the linked
// legal contract on IPFS.

// The following function is used by client as well as
// freelancer.
// To sign the smart contract after it has been deployed.
function signContract(string signedData) {
    if (msg.sender == client_address) {
        clientSign = signedData;
        emit ContractSigned(msg.sender);
        // The client has now signed the contract.
    } else if (msg.sender == freelancer_address) {
        freelancerSign = signedData;
        emit ContractSigned(msg.sender);
        // The freelancer has now signed the contract.
    }
}
```

Fig. 4. Pseudocode part 1

```
// The following function is called by the freelancer.
// It is used to mark a given milestone as complete.
function markMilestoneComplete(uint ID) {
    if (ContractSigned) {
        if (ID is valid && msg.sender is Freelancer) {
            if (milestone is not Completed) {
                if (milestone deadline has not passed) {
                    milestone.status = Completed;
                    milestone.completionTime = now();
                    emit ProposalStatusChanged(ID);
                }
            }
        }
    }
}

// This function is called by the client to approve a
// task completed by the freelancer
// This function would payout money to the freelancer of
// the current milestone if the Client
// reviews the Milestone as Accepted.
function reviewMilestone(uint ID, status Review) {
  if (ContractSigned) {
    if (ID is valid AND msg.sender is Client) {
      MilestoneStatus &status = milestone.status;
      if (status == Completed) {
        if (Review == ACCEPT) {
          milestone.status = AuthorizedForPayment;
          doPayment();
        }
        else if (Review == REJECT) {
          milestone.status = AcceptedAndInProgress;
        }
         else if (Review == CANCEL) {
           milestone.status = Canceled;
        }
        else {
           ABORT();
        }
      }
      emit ProposalStatusChanged(ID);
    }
  }

}
```

Fig. 5. Pseudocode part 2

5 Conclusions

In decentralized market place using Blockchain and Distributed ledger technology, transaction facilitation and matching are improved substantially due to unmodified access to information. This has enabled smart-contract based systems allowing trustless parties to transact directly adhering to the disclosed terms, without manipulation by intermediary platforms. Smart contracts are becoming integral part of many critical systems allowing exchange of payments and services with preset rules. In this paper, we have shown how freelancer application can be deployed in decentralized e-market place using Ethereum Blockchain and how smart contracts can be made legally enforceable with the help of digital signature; making them acceptable between involved parties as well as jurisdiction's legal framework. Our future work is focused on extension of this framework for multi-country cross border contracts involving different legal requirements.

Acknowledgements. *For icons.*

1. Contract by Dmitry Mirolyubov from the Noun Project
2. Court order by Anbileru Adaleru from the Noun Project
3. Developer by I Putu Kharismayadi from the Noun Project
4. Client by Miguel C Balandrano from the Noun Project
5. Contract signing by Vectors Market from the Noun Project

References

1. Yaga, D., Mell, P., Roby, N., Scarfone, K.: Blockchain technology overview. Draft NISTIR 8202, NIST, U.S. (2018)
2. Wurster, S., et al.: Specification on blockchain technology. ISO/TC 307, Tokyo (2017)
3. Raskin, M.: The law and legality of smart contracts. 1 Georgetown Law Technology Review 304, GeorgeTown (2017)
4. Frank, R.: Smart contracts PreDraft. ISO/TC 307, Tokyo (2017)
5. R3, Norton Rose: Can smart contracts be legally binding contracts? http://www.nortonrosefulbright.com/files/r3-and-norton-rose-fulbright-white-paper-full-report-144581.pdf. Accessed 25 Mar 2018
6. Szabo, N.: Smart contracts: building blocks for digital markets. http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart_contracts_2.html. Accessed 10 Mar 2018
7. Wood, G.: Ethereum: a secure decentralized generalized transaction ledger. http://gavwood.com/paper.pdf. Accessed 26 Feb 2018
8. Clack, C., Bakshi, V., Braine, L.: Smart contract templates: foundations, design landscape and research directions (2016). https://arxiv.org/pdf/1608.00771.pdf. Accessed 15 Mar 2018
9. Farrell, S., Machin, H., Hinchliffe, R.: Lost and found in smart contract translation – considerations in transitioning to automation in legal architecture. http://www.uncitral.org/pdf/english/congress/Papers_for_Programme/14-FARRELL_and_MACHIN_and_HINCHLIFFE-Smart_Contracts.pdf. Accessed 21 Feb 2018
10. Freelancer definition. https://en.wikipedia.org/wiki/Freelancer. Accessed 15 Mar 2018

11. Benet, J.: IPFS. https://ipfs.io/ipfs/QmR7GSQM93Cx5eAg6a6yRzNde1FQv7uL6X1o4k7zrJa3LX/ipfs.draft3.pdf. Accessed 27 Feb 2018
12. Freelancer website. https://www.freelancer.com. Accessed 12 Jan 2018
13. Thoen, L.: Milestone payments. https://blog.freelancersunion.org/2014/05/15/dont-get-stiffed-how-ask-milestone-payments. Accessed 7 Mar 2018
14. Information Technology Act, India (2000). http://www.dot.gov.in/sites/default/files/itbill2000_0.pdf. Accessed 28 Feb 2018

Border Control and Immigration on Blockchain

Dhiren Patel[1], Balakarthikeyan[1], and Vasu Mistry[2](✉)

[1] Veermata Jijabai Technological Institute, Mumbai 400019, India
[2] National Institute of Technology, Surat 395007, India
vasu5235@gmail.com

Abstract. In this paper, we propose a system using Blockchain technology to create a decentralized, secure, and scalable departure and arrival records of passengers. We provide a framework using Hyperledger Fabric, for maintaining the inter-port records of the passenger's entry and exit into a country as well as to facilitate gateless entry back to the passenger's country. We attempt to mitigate privacy and legal concerns over biometric data storage on the blockchain. We also explore the possibility of modifying the existing kiosks to work with the blockchain architecture at the backend so that passengers are not required to get familiar with a new procedure.

Keywords: Blockchain and distributed ledger technology · Immigration Gateless entry

1 Introduction

Borders between countries are strictly enforced to prevent illegal movement of people/goods into the country. A huge number of people cross international borders daily. This means effective, secure and scalable record keeping of entries and exits must be performed. For national security, it is crucial that these records are immutable to any attack/alterations. It is important to ensure that movement of people across borders happens easily and seamlessly. It is also imperative for nations to share their records to provide a strict and efficient control mechanism. At the same time these records must be securely stored complying with privacy laws and regulations of that country. This has made it all the more imperative to implement systems to alleviate all the above concerns.

The aim of this work is to implement a secure, decentralized, immutable seamless border control system to enable governments to easily and effectively log people exiting and entering their nations. This system also brings into sync every other port of entry into a unified decentralized system. It also aims to create separate data-paths for international transmission of departure records. The system will also include methods to securely store biometric information to validate/verify passengers automatically.

Rest of the paper is organized as follows: Sect. 2 discusses motivation based on Existing systems, strength of Blockchain Technology and enlists the Security Vulnerabilities in Existing Systems. Section 3 explains fundamentals and basics of Hyperledger framework. Section 4 discusses the proposed workflow for maintaining arrival and exit records of the passengers. Section 5 discusses the implementation architecture on Hyperledger. Section 6 discusses Mitigating privacy and legal concerns over biometrics on the blockchain with Conclusions and References at the end.

2 Motivation and Background

2.1 Existing Border Control Systems

In a typical workflow for a passenger leaving a country via a port of exit he/she would be required to swipe passport at a passport scanner which records the details and then an immigration officer validates and stamps the exit out of the country. The passenger now proceeds towards the boarding gates and takes the flight to the destination.

The next stage is the submission of the Advanced Passenger Information System (APIS) data to the destination country by the flight carrier. The APIS was introduced by the US Customs and Border Protection and is a required criterion for many nations [1]. In India, each flight vessel is obligated to send the APIS data to the destination airport within 15 min of take-off from the origination point [2]. The UK government also has similar rules [3].

Similar systems are there for entry into a country. Common system operational in many countries including the United States are fast-tracked and quick-entry systems like the Global Entry Program. Global Entry allows rigorous background checks verified passport holders to skip lines and an immigration desk and walk to a Global Entry Kiosk to generate an exit pass. The kiosk scans the passports and collects fingerprints to verify authenticity of the passport holder [4].

Mobile passports have also made headway into certain nations as an easy process to clear immigration and skip the lines [5].

All of these systems have clearly contributed to ease of air-travel especially for citizens and have helped alleviate extensive screening and congestions at airports. Similar systems exist for entry through rail/sea. The downside being each of these systems have exposed us to new points of failures and security breaches.

The centralized structure, and the difficulty involved in keeping records securely synchronized across entry points are key issues the proposed system tackles. It also paves the way for nations to receive passenger information seamlessly without trust on a third-party. The proposed system allows nations to leverage biometric identities of their citizens to validate their entry. In addition the system can be integrated with existing automated kiosks making adoption easier.

2.2 Blockchain

Blockchains are distributed digital ledgers of cryptographically signed transactions that are grouped into blocks. Each block is linked to the previous one after validation and consensus of all participating nodes. As new blocks are added, older blocks become more difficult to modify. New blocks are replicated across all copies of the ledger in the network, and any conflicts are resolved automatically using established rules [6].

At their most basic level, blockchain enables a community of users to record transactions in a ledger that is public to that community, such that no transaction can be changed once published. A block is an individual unit of a blockchain, composed of a collection of transactions and a block header. A block header keeps a collection of metadata about the block that contains a hash-value of its parent in the blockchain, and a hash of the aforementioned metadata and the data of the block itself [7].

In a public or permissionless blockchain anyone can participate without a specific identity. Public blockchains typically involve a native cryptocurrency and often use consensus based on "proof of work" (PoW) and economic incentives. Permissioned blockchains, on the other hand, run a blockchain among a set of known, identified participants. A permissioned blockchain provides a way to secure the inter-actions among a group of entities that have a common goal but which do not fully trust each other, such as businesses that exchange funds, goods, or information. By relying on the identities of the peers, a permissioned blockchain can use traditional Byzantine-fault tolerant (BFT) consensus [8].

2.3 Security Vulnerabilities and Weak Points in the Current System

The major associated problem with the current method of entry/exit is the centralized nature of data, making it an easy target of attacks and attempts at manipulation may cause complete or partial data loss.

With multitude of kiosks validating data and entering data into this centralized database, it could be disastrous if security flaws are found in the system.

The next point of failure might be the trust on the Airline carrier to report advance information of passengers' arrival. This inherent trust might be misused along with existing automated passport control kiosks.

Lastly the entry/exit records are always prone to modification either by malicious third parties or due to internal political pressures etc. This data must be immutable.

The intercommunication between multiple port-of-entries of a country with a centralized database is also a serious cause of concern in terms of security and scalability.

Apart from security issues, a simple lapse in stamping/recording of information might render a person as an invalid entrant into a country with no entry record and with no way of leaving the nation. This is especially true since many countries rely on a stamp on the passport to validate arrival, sometimes vis-a-vis maintaining a centralized record [9]. The adoption of this system prevents possibilities of such incidences especially for countries not maintaining centralized records.

3 Hyperledger Fabric

Fabric is a distributed operating system for permissioned blockchains that executes distributed applications written in general-purpose programming languages (e.g., Go, Java, Node.js) [8].

A distributed application using Fabric consists of two parts:

- A smart contract, called chaincode, which is program code that implements the application logic and runs during the execution phase. Special chaincodes are for managing the blockchain system and maintaining parameters, collectively called system chaincodes [8].
- An endorsement policy that is evaluated in the validation phase. An endorsement policy acts as a static library for transaction validation in Fabric, which can merely be parameterized by the chaincode. Only designated administrators may run system management functions and have the right to modify the endorsement policy [8] (Fig. 1).

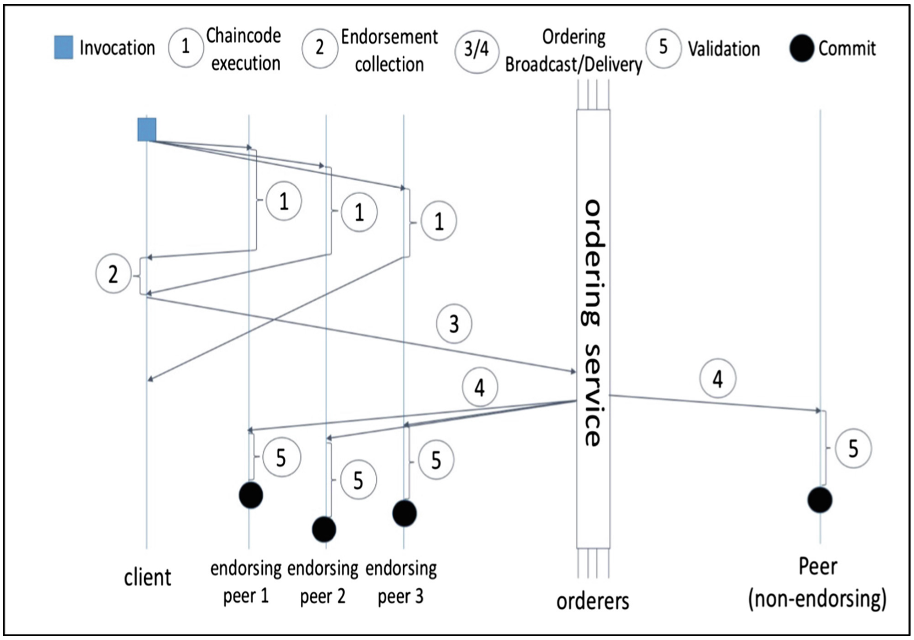


Fig. 1. Hyperledger Fabric transaction workflow [8]

As Fabric is permissioned, all nodes that participate in the network have an identity. Nodes in a Fabric network take up one of three roles:

- 'Clients' submit transaction proposals for execution, help orchestrate the execution phase, and, finally, broadcast transactions for ordering [8].
- 'Peers' execute transaction proposals, validate transactions and maintain the blockchain. Only the 'Endorsing peers' execute all transactions while all peers maintain the blockchain ledger [8].

- 'Orderering Service Nodes' (OSN) (or, simply, orderers) are the nodes that collectively form the ordering service. The ordering service establishes the total order of all transactions in Fabric, where each transaction contains state updates and dependencies computed during the execution phase, along with cryptographic signatures of the endorsing peers that computed them [8].

4 Proposed Workflow of Border Control System on Blockchain

4.1 Maintaining Entry/Exit Records on the Blockchain

The proposed workflow is similar to the current workflow with an essential difference that the immigration officer now marks his immigration decision which is recorded onto the blockchain. Once all details are recorded the passenger is allowed to pass through.

- In case of a departure workflow, the system automatically finds the corresponding arrival record into the country for a foreign citizen and validates this departure.
- In case of arrival, the system automatically finds the corresponding departure record of the citizen and validates his entry.

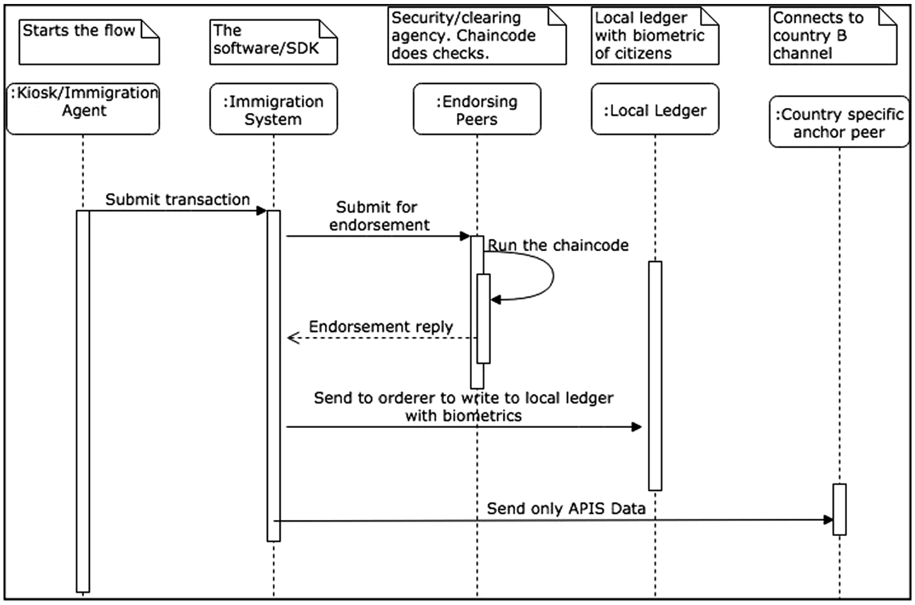

Starts the flow
The software/SDK
Security/clearing agency. Chaincode does checks.
Local ledger with biometric of citizens
Connects to country B channel
:Kiosk/Immigration Agent
:Immigration System
:Endorsing Peers
:Local Ledger
:Country specific anchor peer
Submit transaction
Submit for endorsement
Run the chaincode
Endorsement reply
Send to orderer to write to local ledger with biometrics
Send only APIS Data

Fig. 2. Departure sequence diagram

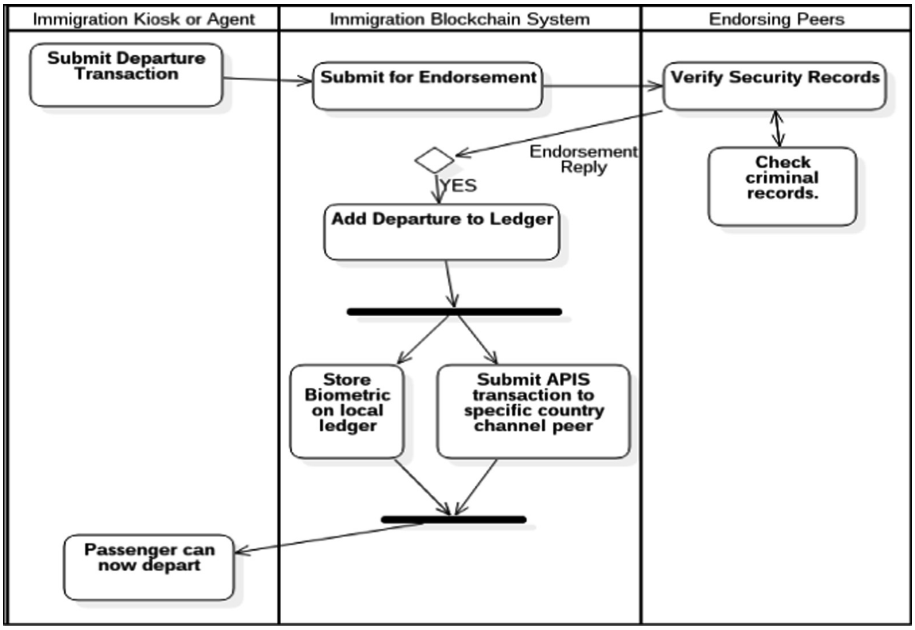

Immigration Kiosk or Agent
Immigration Blockchain System
Endorsing Peers
Submit Departure Transaction
Submit for Endorsement
Verify Security Records
Check criminal records.
Endorsement Reply
YES
Add Departure to Ledger
Store Biometric on local ledger
Submit APIS transaction to specific country channel peer
Passenger can now depart

Fig. 3. Departure activity diagram

4.2 Detailed Example Workflow During Departure

Referring to Fig. 2 we see that the immigration officer submits a transaction for departure to the system. Here, a transaction refers to the departure of the passenger.

The transaction is then sent to endorsement to the corresponding endorsement peers. The endorsement peers could be from local security agencies to a central no-flier database etc. Once the endorsements are successful, the entry is added into the departure ledger and biometric information stored in the local-ledger if the departing passenger is a citizen of the given country. In case of a rejection, it is added to a separate log with rejection comments by the immigration officer. The departure sequence diagram shows as various stages through with the system passes and its interaction with the ledger.

The two critical phases involve capturing and storage of biometrics securely to validate a passenger on return and to post the APIS data on to the specific country anchor peer so that the destination country is aware of the passenger. The robustness of the system lies in the fact that since there is no central machinery involved all ledgers in the airport work in a distributed fashion holding everyone's records. Additional all other ports in the nation are also now aware of this departure and biometric record which can be validated in case of a citizen on his return. Thus the system lacks central failure points but at the same time maintains a copy accessible with everyone.

A similar system is followed during the arrival of a passenger as noted in Fig. 4. The passenger admittance is subject to endorsement from the endorsing peers. This includes security agencies and validating departure record for a citizen. For an

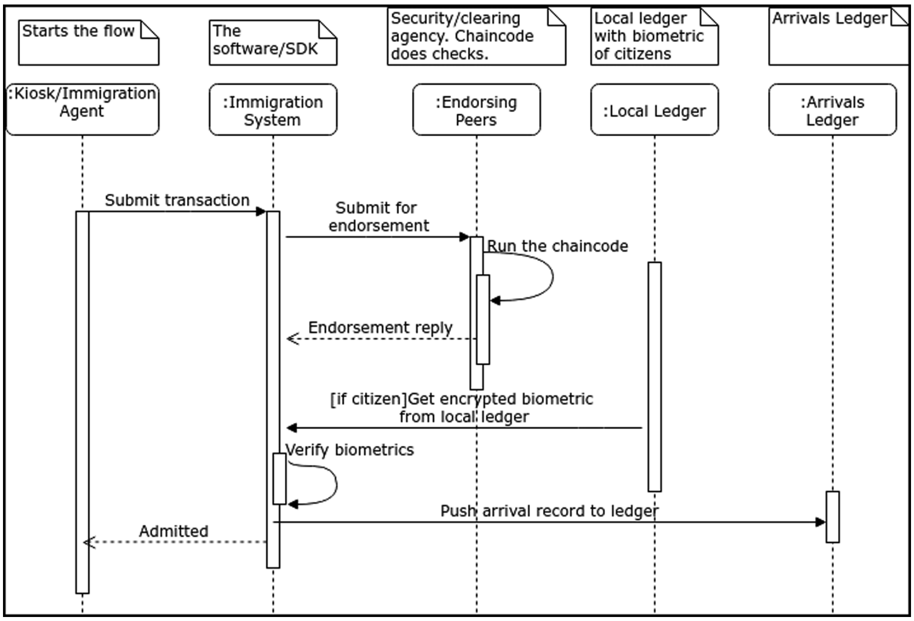


Fig. 4. Arrival sequence diagram

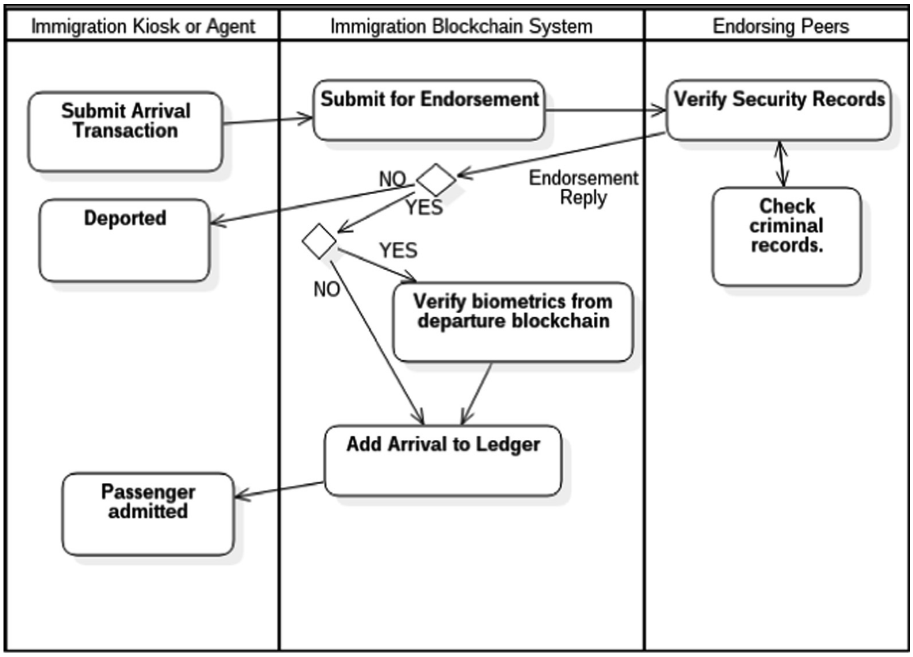


Fig. 5. Arrival activity diagram

incoming citizen, additionally biometrics shall be compared before final admission. This automates the procedure for citizens and removes involvement of immigration officers who now will have a supervisory role.

The following points shall be noted from the swim lane diagrams:-

- The check for corresponding arrival/departure records happens at an endorsement peer in the defined chaincode.
- Each security agency can define chaincodes, which can search databases and/or perform complex actions to complete security procedures and provide with an endorsement.
- The immigration officer can chose not to submit the transaction and directly reject it. Rejections are recorded in a separate peer node.
- The system can integrate with existing passport-control kiosks, with minor reconfiguration. The kiosks act as the client triggering the transactions.

5 Implementation Architecture

Hyperledger is the preferred choice for this use case as it provides a fast and scalable system with features complementing the specific needs and deployment of permissioned blockchains.

Hyperledger does not require mining and instead uses Endorsement for the Ordering Services. The Hyperledger Certificate Authority (CA) allows developers to enroll peers using existing public key infrastructure.

The endorsement policy ensures that the Immigration officer might not be the only endorser for approving a passenger (in this case, giving a vote as to whether the person gets in or not), additional security agencies can be made part of this endorsement process. Since endorsement chaincode can be written in non-deterministic languages like Go/Java, a quick Banned Flier list data lookup can be performed as an additional endorsing peer.

Each port maintains a minimum of one peer node (Immigration officer endorsing peer) although multiple peers can be maintained.

Channels provide a way to replace the APIS system with the destination port's receiving peer which can now easily be notified of arrival of passengers with all APIS data encoded on to it.

The following Fig. 6 shows the deployment architecture of the system. Here, symbols are derived from Hyperledger's default set of symbols for showing interactions. The oval represents an organization. Peers in an organization are always connected to each other.

- In this deployment, we see Port-1, Port-2 and Endorsing Organizations on a common channel.
- A separate organization called Airports Authority is maintained as the peer to send APIS information to specific destination countries. The chaincode in this peer shall be invoked by the system after successful departure as shown in Fig. 4.

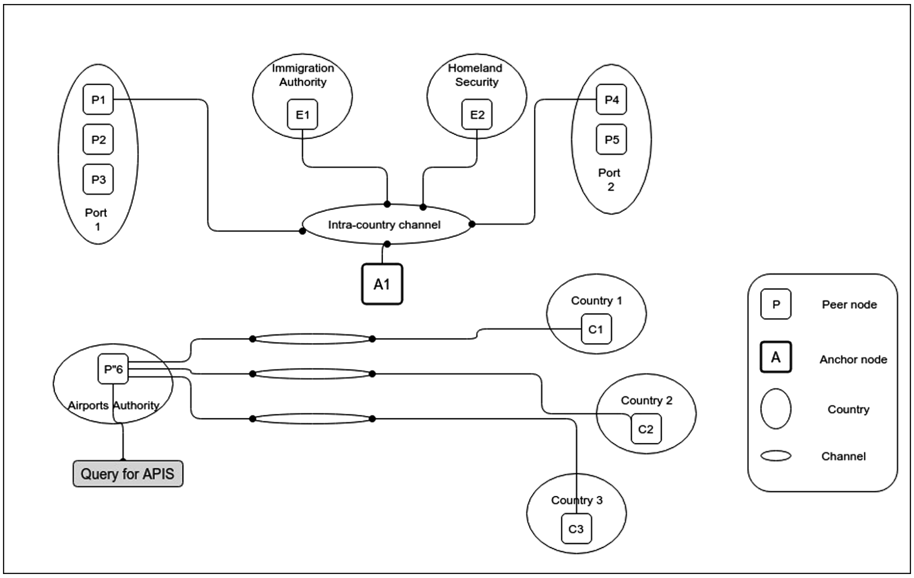


Fig. 6. Deployment architecture on Hyperledger

- For the sake of brevity, ordering services on the channel are omitted in the diagram but are assumed to exist.

The system will integrate with existing kiosks/immigration desks as they will act as clients interacting with the Immigration System APIs as shown in Figs. 3 and 5.

The data stored on the ledger shall be the basic details as recorded by APIS systems internationally. This include names, passport number, source and destination ports, along with carrier vessel identification and boarding pass details [2].

In addition the passenger's country would store biometric information to validate the passenger seamlessly on his return.

5.1 Pseudocode for the Immigration System

We provide the pseudocodes using the Hyperledger Composer CTO language containing the `model` and the `script` which forms the chaincode in Composer. Hyperledger Composer is a fast and rapid prototype deployment tool to be used with Hyperledger Fabric.

In Hyperledger Composer all interactions are performed using participants and assests. Assets are entities that change their state during the course of a transaction. The assets are recorded and saved on the blockchain as well as the transactions [10].

We model the system in terms of a passenger asset due to the constraints of Hyperledger CTO language. It must be noted that in a real installation we need not model it in a similar way but it is a good representation to store the details of a passenger and his passport. A passenger asset is identified by his `passportNo` as a unique key. In addition we show participants immigration-officer-peer node and the endorsing agency as assets. In a real deployment we might have multiple endorsing agencies. We also have defined our depart transaction here and its attributes. Similarly a transaction for arrival can be created.

Composer – Model (Fig. 7)

```
Pseudocode 1 Composer Model
1: namespace org.india.immigration
2: asset Passenger identified by passportNo:
       passportNo
       firstName
       lastName
       middleName
       passportExpiry
       visaNo
       destCountry
       depCountry
       depStatus
       endorsers
3: Port
       portNo
       portName
4: participant ImmigrationOfficerPeer identified by officerID
       officerId
       officerName
       port
5: participant EndorserAgency identified by agencyCode
       agencyCode
       agencyName
6: transaction Depart
       departId
       Passenger p
       ImmigrationOfficerPeer i
       officerRemark
```

Fig. 7. Hyperledger composer – model pseudo-code

Composer – Script (Fig. 8)

```
/**
 * A sample Immigration Script API to submit a passenger
departure transaction.
 * A dummy endorsement function is shown. This is just to
model it like a hyperledger endorsement returning from
 * a chaincode running at a hyperledger endorsement peer.
 */
```

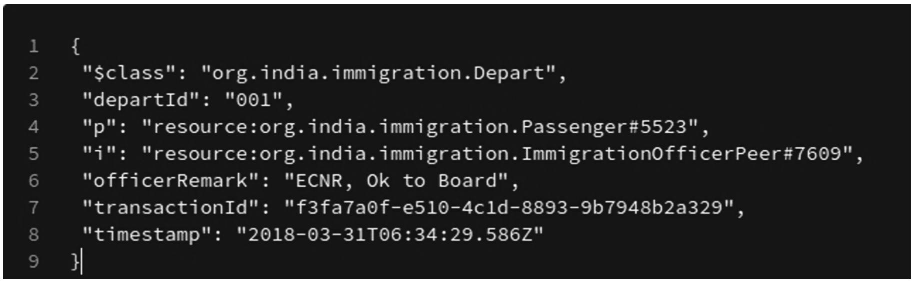
```
Pseudocode 2 Composer Script
    Input:
        tx: Transaction data
1: function GETENDORSEMENT( tx )
2:     if (tx and passenger data are valid) then
3:         return "Departure Granted"
4:     else
5:         return "Departure Not Granted"
6:     end if
7: function DEPART( tx )
8:     tx.p.departure_status ←GETENDORSEMENT(tx)
9:     tx.p.endorsers.push(endorserId)
10:    assetRegistry ←GETASSETREGISTRY('org.india.immigration.Passenger')
11:    return assetRegistry.update(tx.p)
```

Fig. 8. Hyperledger composer – script pseudo-code

The Script shown here is representative of our Immigration System deployment and in an actual use case the client machines shall interface with the API of our Immigration System to carry out the transaction i.e. letting a passenger depart and our arrive and collecting valid endorsements etc. The Client only needs to call the API to initiate the transaction and wait for results to come back. The system will do all the business logic processing and call appropriate chaincodes to execute the transaction.

Since Hyperledger Composer does not currently support custom Endorsement policies we have created a dummy endorsement function. In real life as soon as the transaction is submitted the endorsing peers would run their chaincodes to validate the transaction and such a dummy function is not needed. Here the function only serves to remind us about how the transaction would be endorsed.

The `Depart(transaction)` sets the departure status on the passenger asset and updates the asset.



```
{
 "$class": "org.india.immigration.Depart",
 "departId": "001",
 "p": "resource:org.india.immigration.Passenger#5523",
 "i": "resource:org.india.immigration.ImmigrationOfficerPeer#7609",
 "officerRemark": "ECNR, Ok to Board",
 "transactionId": "f3fa7a0f-e510-4c1d-8893-9b7948b2a329",
 "timestamp": "2018-03-31T06:34:29.586Z"
}
```

Fig. 9. Demo transaction

Figure 9 shows a sample executed transaction summary based on our initial model on the Composer Playground Web UI [11]. Here we see that the passenger departure is logged with an ID for easy query. Along with it the passenger asset and the immigration officer asset is also logged. The `officerRemark` along with timestamp and `transactionID` show a valid departure status.

6 Addressing Privacy and Legal Concerns Over Biometrics on the Blockchain

A very important question arises with regards to privacy of the passenger's biometric information and legal issues with its dissemination. Firstly, no biometric information shall be transmitted outside the passenger's country and shall be stored as per the laws of the country e.g. Aadhaar UID system in India.

Countries like Germany provides users an option to use biometric passwords and the data can be stored in double RSA hashing to implement the same. The above pseudocode provides a small example of using Double RSA encryption technique. The code can be easily written in Javascript, by utilizing the popular RSA library called JSEncrypt [12]. Here a public-private key pair based on passport information or other has to be generated for a passenger so that his private key is needed for reading his biometric data from the stored asset. The biometricHash in the stored asset corresponds to this encrypted biometric data (Fig. 10).

```
Pseudocode 3 Double RSA technique
    Input:
            biometricData: An object holding biometric data.
 1: procedure ENCRYPTION(biometricData)
 2:     crypt1 ←ENCRPYT()
 3:     crypt2 ←ENCRPYT()
 4:     crypt1.setPublicKey(PassengerPublicKey
 5:     crypt2.setPublicKey(ImmigrationAgencyPublicKey)
 6:     biometrics ←biometricData
 7:     enc1 ←crypt1.encrypt(biometrics)
 8:     encFinal ←crypt2.encrypt(enc1)                ▷ Double RSA encryption
 9:     return encFinal
    Input:
            encFinal: Holds cipher text after performing double RSA.
10: procedure DECRYPTION(cipherText)
11:     crypt1.setPrivateKey(PassengerPrivateKey)
12:     crypt2.setPrivateKey(ImmigrationAgencyPrivateKey)
13:     dec1 ←crypt2.decrypt(encFinal)
14:     biometricData ←crypt1.decrypt(dec1)
15:     return biometricData
```

Fig. 10. Double RSA for biometric data pseudo-code

The first round of hashing is performed by the passenger's private key while an additional hashing is performed by the Border control agencies private key. Passenger's private key can be generated from his own biometrics/passport, making it mandatory for the passenger to be present to enable decryption of his record. Thus although passenger's biometric identity might exist on the blockchain without the passenger such data cannot be accessed and utilized thus alleviating any concerns over privacy of this sensitive data.

7 Conclusions

We have looked at an interesting use case of global nature and implemented a secure, decentralized, immutable seamless border control system to enable governments to easily and effectively log people exiting and entering their nations.

In this paper we have addressed the security and reliability issues of current systems. Using blockchain on Hyperledger, this system brings into sync every other port of entry into a unified decentralized system. We have shown creation of separate data-paths for international transmission of departure records. A permissioned infrastructure is envisioned where the government security agencies act like gate-keepers (endorsers) automatically allowing entry/exit. We have also envisioned storage of biometrics in a local ledger allowing citizens to easily enter their own country while also keeping privacy concerns at bay. Currently the system is under active development. We look forward to further improve the system with simple, automated interfaces.

References

1. Advanced Passenger Information System Air Canada. https://www.aircanada.com/us/en/aco/home/plan/travel-requirements/advancepassenger-information.html. Accessed 20 Feb 2018
2. APIS, Bureaus of Immigration India. https://boi.gov.in/content/apis-advanced-passenger-information-system. Accessed 11 Mar 2018
3. Transfer e-Borders data: general aviation and maritime. https://www.gov.uk/government/publications/transfer-e-borders-data-general-aviation-and-maritime. Accessed 20 Feb 2018
4. Global Entry U.S. Customs and Border Protection homepage. https://www.cbp.gov/travel/trusted-traveler-programs/global-entry. Accessed 7 Mar 2018
5. Mobile Passport Control App U.S. Customs and Border Protection. https://www.cbp.gov/newsroom/national-media-release/new-mobile-passport-control-app-available. Accessed 7 Mar 2018
6. Yaga, D., Mell, P., Roby, N., Scarfone, K.: Blockchain technology overview. Draft NISTIR 8202, NIST, U.S. (2018)
7. Wurster, S., et al.: Specification on blockchain technology. ISO/TC 307, Tokyo (2017)
8. Cachin, C., Barger, A., Manevich, Y.: Hyperledger fabric: a distributed operating system for permissioned blockchains. https://arxiv.org/abs/1801.10228v1. Accessed 20 Mar 2018
9. No arrival stamp on Indian's passport at Mumbai airport. https://timesofindia.indiatimes.com/city/mumbai/No-arrival-stamp-on-Indians-passport-at-Mumbai-airport-he-cant-return-to-UAE/articleshow/47930826.cms. Accessed 15 Mar 2018

10. Hyperledger Composer. https://hyperledger.github.io/composer/latest/reference/reference-index. Accessed 28 Mar 2018
11. Hyperledger Composer Playground. https://composer-playground.mybluemix.net/. Accessed 31 Mar 2018
12. JSEncrypt. http://travistidwell.com/jsencrypt/. Accessed 28 Mar 2018

Application Track: Business Models and Analyses

RPchain: A Blockchain-Based Academic Social Networking Service for Credible Reputation Building

Dong Qin[1], Chenxu Wang[1,2](✉), and Yiming Jiang[2]

[1] School of Software Engineering, Xi'an Jiaotong University, Shanxi, Xi'an, China
[2] MoE Key Lab of Intelligent Networks and Network Security, Xi'an Jiaotong University, Xi'an, China
cxwang@mail.xjtu.edu.cn

Abstract. The development of the Web 2.0 technology has brought the prosperity of open science which is devoted to making scientific research, data and dissemination accessible to all levels of an inquiring society, amateur or professional. One of the challenges faced by open science is how to build the scholarly reputation of participants credibly and fairly. Existing academic social networking services use peer reviews to address this problem. However, there are still some drawbacks of these services. They are not friendly to the general public and lack effective incentive mechanisms to attract ordinary users. Moreover, these centralized services have the risks that review records might be tampered or lost. In this paper, we present a blockchain-based academic social network service model which has irrevocable peer review records, traceability of reputation building and appropriate incentives for content contribution. In order to achieve these goals, we propose a novel consensus algorithm named proof of reputation (PoRe) which utilizes the reputation of participants for consensus. Finally, we implement a prototype of the model and empirically validate its security and effectiveness.

Keywords: Blockchain · Academic social network · Reputation building
Proof of reputation

1 Introduction

With the development of the Web 2.0 technology, the academia has entered an era of collaboration and sharing. The "Open Science" proposal which advocates data sharing, open peer evaluation, citizen science, and so on, has gradually become the main-stream of research in the current digital age [1–4]. However, this also brings the problems of scholarly reputation building and digital copyright protection [1, 5].

There are several academic social networking services (ASNS) such as Research Gate [9] and Mendeley [10] to overcome these issues [6–8]. However, there are still some drawbacks [11–13]. First, these services are focused on serving academic scholars, which counters with the proposal of universal scientific research [11]. Second, these services lack an appropriate incentive mechanism to attract ordinary participants [12].

https://doi.org/10.1007/978-3-319-94478-4_13

Some scholars are reluctant to use ASNS due to the lack of trust in them [13]. Third, the public peer review mechanisms used by these services may harm the social relationships among users. In practice, most researchers prefer to review privately [14]. Forth, centralized ASNS such as Research Gate cannot avoid the security risks associated with traditional centralized network services. Attacks against these services may result in data loss or falsification which is fatal to fair reputation building. Therefore, it is desired for an ASNS that has a high degree of openness and appropriate incentives for participation.

The emergence of the blockchain technology provides us a feasible way to build such a service. The blockchain technology has several important properties such as decentralization, traceability, privacy, secure transfer of value and so on. Using blockchain to store peer-reviewed records ensures the maximum level of security, because it is almost impossible for attackers to destroy or tamper with the records. Furthermore, the cryptographic pseudonym provided by blockchain also maximizes the privacy of peer reviews.

However, it is non-trivial to employ the blockchain technology to build an adequate ASNS. First, we must design a suitable blockchain model to provide a platform which should fairly build a user's reputation based on other peers' reviews. Second, we have to develop a credible consensus algorithm to select an eligible peer for block maintenance. Third, we must design appreciate incentive policies to attract ordinary participants. Finally, we should consider the security problems in the implementation of the system.

In this paper, we present RPchain, a blockchain based academic social networking service model. In order to address the first issue, we propose three kinds of transactions, namely token transactions, content transactions, and vote transactions. Token transactions are used for value transference and exchange. The content transactions represent the content publicized by users. Content includes topics/issues posted by users, responses by peers, and smart contracts, etc. Vote transactions represent the votes that peers give to specific content (in favor of it). In order to ensure fair voting and systematic security, only users authorized by smart contracts or approved by consensus are eligible to vote. Votes of contents belonging to the same user are cumulated to evaluate the user's reputation. All transactions and contracts are stored in the blockchain, giving users credible copyright protection and traceability of reputation building. In addition, we divide the life cycle of RPchain into three phases: the initial phase, extension phase, and stable phase. Different consensus algorithms are adopted in different phases in order to facilitate the development and stability of the model.

The consensus algorithm is the soul of a blockchain system. However, existing computing-power-oriented consensus algorithms such as Proof of Work (PoW) used in Bitcoin is not consistent with the aims of our service-oriented model. In order to address this issue, we propose a novel consensus algorithm named proof of reputation (PoRe). PoRe uses the reputation gained by the participant's published content transactions to reach a consensus. When the content receives votes, the reputation of the content is assessed based on the weight of those votes (the more the votes, the greater the contributions). Then, when the conditions permit, users participate in the PoRe consensus using the reputation obtained by the content transaction. The higher the reputation value used for consensus, the lower the difficulty of the PoRe consensus, and the greater the probability that the user will get the next block. That is, the more contribution of a user to the system, the greater the chance for it to obtain the reward, which is exactly the virtuous circle required by ASNS.

To address the incentive problem, we introduce three different kinds of rewards, namely consensus rewards (block rewards), vote rewards, uncle block rewards. Consensus and vote rewards are used to incent users to participate in the maintenance of the blockchain and give reviews on content. Uncle block rewards are used to compensate users who fail in the consensus competition but contribute to the system significantly. We take appropriate approaches to the security issues in the design of RPchain. Finally, we implement a prototype of the proposed model and conduct experiments to evaluate its security and effectiveness. We evaluate the performance of the model. Experimental results show that our model is secure in the presence of malicious nodes and outperforms the general blockchain structure in terms of efficiency.

The main contributions of this paper are summarized as follows:

- We present RPchain, an academic social networking service model based on the blockchain technology. The model is capable of credible reputation building and academic copyright protection.
- We propose the proof of reputation (PoRE), a reputation-based consensus algorithm for the maintenance of the blockchain system. PoRe satisfies the requirements that the more contribution of a user, the greater the probability it gets the write right of a block and the reward.
- We considered the incentives mechanisms and security issues. Proper incentive policies are introduced to incent users to participate in the system while guaranteeing its security.
- We conduct experiments to evaluate the security and efficiency of RPchain. The experimental results show that RPchain can ensure the security of the system in the conditions that the percentage of malicious members is less than 50% in the initial phase and the proportion of the contents publicized by malicious nodes is no more than 40%. Moreover, our model outperforms the general blockchain structure in terms of efficiency.

The rest of this paper is organized as follows. Section 2 reviews the related work. In Sect. 3, we present the architecture of the model. In Sect. 4, we describe the PoRe consensus and the blockchain protocol. In Sect. 5, we show the incentive mechanisms. We conduct experiments to validate the security and efficiency of the system in Sect. 6. Finally, we conclude the paper in Sect. 7.

2 Related Work

2.1 Blockchain-Based Reputation Building and Social Networks

Sharples et al. [15] proposed to use blockchain to store education records which award scholars' reputation. However, they did not propose a specific model. Dennis et al. [16] attempted to establish a transaction-oriented blockchain reputation system. Schaub et al. [17] built a model of e-commerce reputation based on blockchain and focused on the privacy of evaluation. However, they did not optimize the consensus and structure of the blockchain. Konforty et al. [18] built a blockchain-based social network that focuses on the economy of attention and prioritizes information in a manner that is of interest. It is noteworthy that Steem employed a content-votes-based consensus method which is similar to ours [19]. However, the essential difference lies in who have the right to vote. Steem uses the mechanism of one ticket per token, indicating that everyone who owns a token has the right to vote. This poses a challenge to the stability of the system. For instance, an attacker who has enough wealth can easily influence the voting weight in the consensus. In addition, the system provides lucrative cash bonuses to content publishers and voters based on how many votes they cast, resulting in vicious competition. Actually, people are more concerned with the outcome of the vote rather than the content itself. This will make the community more and more far from its original intentions. In this paper, we are more concerned with the value of knowledge itself rather than the tokens.

2.2 Blockchain Consensus

Consensus algorithm is the core of a blockchain system. Because the blockchain is a peer-to-peer decentralized network, every node can enter and leave the network at any time. Therefore, it is desired for a reliable consensus algorithm to ensure the consistency of the blockchain and the resistance of "Sybil attacks". As one of the most successful applications of the blockchain, Bitcoin uses the PoW consensus algorithm, requiring nodes to meet certain computing power before it is probable to obtain block ownership.

Bitcoin's PoW algorithm uses the hash algorithm to calculate a random value in order to meet pre-set conditions. Every personal computer can do such a calculation. This coincides with the original intention of Bitcoin to allow everyone to participate in the system. However, with the increase of the Bitcoin value, the competition for hashing power has increased. ASIC mining machines which have a faster computing speed than CPU and GPU appear. This results in that individual nodes are difficult to profit only by their own resources. Subsequently, individuals are organized to form communities of interest-sharing, which are called mining pool. Such a result obviously departs from the original intention of Bitcoin. At the same time, this also lead to a series of security problems [20–22]. For example, selfish mining leads to the security assumption (that the attacker controls hashing computing power less than 50% of the total network) is threatened. The traditional proof of stack (PoS) algorithm uses the shares held by users in the system to express their "loyalty" to the system and obtain the opportunity to add blocks. Although the consumption of resources for hashing computing is reduced, the initial allocation of tokens has not been well addressed. This may result in the rich richer. Other consensus mechanisms, such as Ripple [23], maintain a list of trusted nodes and using voting methods to achieve faster consistency. The problem is that the voting mechanism does not have a transcendental trust basis.

In this paper, we present the proof of reputation (PoRe). The PoRe chooses system-friendly participants as the leader of a block and ensures that the behavior of malicious nodes is limited.

3 RPchain

3.1 RPchain Overview

RPchain is aimed to provide an academic exchange platform for scholars and serve those who like to contribute to scientific discussions. In the current system, user identities are represented by encrypted pseudonyms. This ensures private peer reviews. Each user can establish its own reputation through the corresponding academic activities. For example, users can post their own academic ideas, vote on other ideas and discuss with other users. The architecture of the model is shown in Fig. 1.

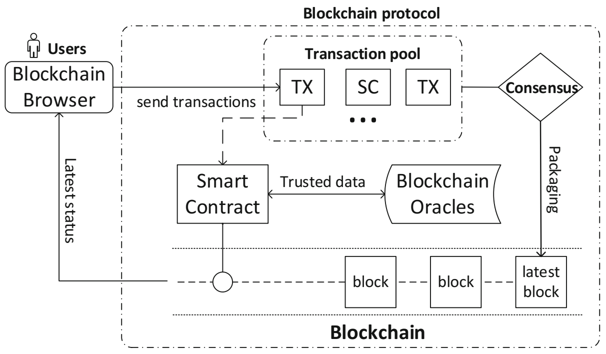


Fig. 1. The architecture of the RPchain model

In the RPchain model, users view the content and interact with other peers through the blockchain browser. If a user needs to post a new topic or ask a new question, he can send a content transaction to a new smart contract which will deal with all the replies and votes on the topic or question. Users can also send the content transaction to other similar smart contracts to participate in the discussion. If the user has the right to vote, he can vote to support the content. When new blocks are found by miners according to the consensus algorithm, the transactions and smart contracts in the transaction pool are packaged into the latest block. At the same time, the blockchain browser obtains the latest state of the blockchain and displays the content to users. All of the above operations are encapsulated into the blockchain browser, allowing users to browse RPchain just as if they were browsing a normal academic community, even without the delay of traditional servers.

3.2 Smart Contracts

A smart contract is essentially a piece of code stored on the blockchain. When some pre-set state is triggered, the code in the contract is executed automatically. Every node in the blockchain runs the code to ensure that the results of the execution are correct.

In RPchain, smart contracts are mainly the carriers of content transactions and vote transactions sent by users. At the same time, in order to facilitate scholars to use the service while guarantee the academic authority of the published content, smart contracts are also used to register the identity of academic researchers in the real world (RPchain allows organizations such as Research Gate to endorse users' identities and ensure their authenticity). The smart contracts used to process the transactions are pre-given in the model. If needed, smart contracts use the blockchain oracle to get trusted outside data, such as the current token price and so on. The adoption of smart contracts greatly improves the scalability of our model and provide users with diversified services.

3.3 Reputation Building

RPchain builds user reputation based on two kinds of data. One is users' academic contributions to the community, including posted topics and reviews on other content. The other is the external academic contribution in the real world which is certified by smart contracts. Only users who have a block or authenticated scholar identity in the blockchain have the right to vote. In the rest of this paper, we call these users as members. Members vote for content transactions and the weights of votes depend on the reputation of the members. Each member can only vote once for the same content transaction.

The more votes a user receives, the higher the user's reputation in the system. If the user is a certified scholar, the smart contract can update his/her reputation in RPchain based on the academic achievements obtained from the real world. Specifically, the initial weight of votes for all members is 1.0. The weight of the votes of certified scholars is related to his/her reputation in the real world, usually greater than 1.

3.4 RPchain Life Cycle

As shown in Fig. 2, the life cycle of RPchain is divided into three phases, namely initial phase, extension phase, and stable phase. Each phase has a specific consensus to meet the requirements of different developmental stages of the service.

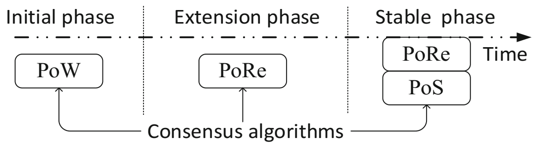


Fig. 2. The life cycle of RPchain

Initial Phase

In order to solve the cold-start problem, the main purpose of the initial phase is to increase the number of members and content transactions. If the number of members and content transactions is not sufficiently large, the voting results are not good enough to reflect the reputation of a user. Hence, in the initial phase, we adopt the PoW consensus. When the number of content transactions in the blockchain reaches a certain amount and the proportion of votes in the content transaction exceeds a certain threshold, the model enters the extension phase.

Extension Phase

In the expansion phase, RPchain uses the PoRe for consensus. In the extension phase, the aim of the model is to promote users who contribute to the system most as managers of the blockchain. For security reasons, we use a function *Mode*() to randomly output a set of content transactions with fixed sizes to participate in the consensus in each round. The hash of the latest block and a specific random value, as input to the *Mode*() function, determines which content transactions to participate in the consensus. RPchain adjusts the size of the content transaction set outputted by the *Mode*() function according to the speed of block generation.

Stable Phase

When the community enters into the stable phase, we allow both the PoRe and PoS consensuses. This satisfies the needs of more nodes and allows more nodes to participate in the maintenance of the blockchain.

3.5 Blockchain Structure

In order to optimize the scalability of the general blockchain structure, we employ the blockchain structure used by Bitcoin-NG [24]. Bitcoin-NG separates the block leader election process from the serialized transactions. In such a way the latency is limited only by the propagation delay of the network, and the bandwidth is limited only by the processing capacity of the individual nodes. As shown in Fig. 3, the blockchain contains two kinds of blocks, namely key blocks and mini blocks. The key blocks, like the blocks in the general blockchain structure, are held by winners in the consensus. They are also the leaders of mini blocks during each consensus round. In each consensus round, the key block holder is allowed to generate mini blocks at a set rate smaller than a predefined threshold. Transactions stored in mini blocks are signed by the key block holder. Such a structure reduces the impact of consensus delay and transaction confirmation time. When a fork occurs, we use the longest and heaviest principle of Bitcoin-NG to choose the most efficient chain. This promises the blockchain more security. For ease of description, the blocks mentioned below are referred to the key block.

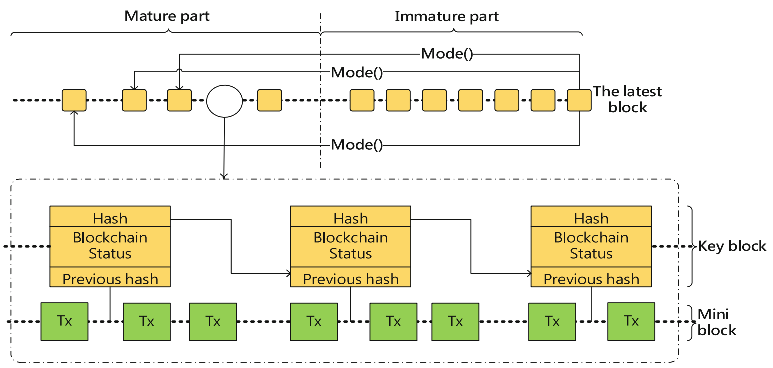


Fig. 3. The structure of the RPchain

Since members' online time is uncertain, new content transactions usually receive very few votes at the beginning. In order to make the voting results more credible, we divide the blockchain into two parts, namely immature part and mature part, as shown in Fig. 3. Content transactions in the immature part cannot be used to participate in the PoRe consensus. If the content transaction is in the mature part and is within the output set of the $Mode()$ function in the current round, it can participate in the consensus.

4 Blockchain Consensus and Protocol

In this section, we describe the PoRe consensus and the blockchain protocol. In order to explain it more clearly, we first give several relevant definitions.

4.1 Related Definitions

- Hash function $\boldsymbol{H}()$: A hash function $H()$ is used to verify the integrity of information. For example, a $H()$ can be SHA256.
- The state of a block $\boldsymbol{state}()$: The $state()$ function contains all the relevant information about a block, including the Merkel root of the transaction, time stamp, block height, a target value, the status of the current blockchain, and etc.
- PoW difficulty $\boldsymbol{D}$: The difficulty D is used to adjust the time interval that a new block is mined in order to maintain the stable growth of the blockchain, since the hashing computing power of the whole network is constantly changing.
- (Reputation value $\boldsymbol{R}$) R represents the current reputation value of the content transaction.
- $\boldsymbol{KeyGen}() \rightarrow (\boldsymbol{pk}, \boldsymbol{sk})$. The $KeyGen()$ function is used to generate a key pair (pk, sk), where the public key pk represents the address of the user (user's identity) in the system, and the private key sk is confidential and is used to sign transactions in the blockchain.
- $\boldsymbol{Tx}\left(\boldsymbol{pk_i}, \{\boldsymbol{pk_j}, \boldsymbol{Tx_{id}}\}, \boldsymbol{m}\right)$. The $Tx()$ function generates a transaction and m represents the type (tokens, contents, or votes) of the transaction. If it is a token transaction, the sender is pk_i, and the receiver is pk_j. In the case of voting transactions and content transactions, the recipient is the ID Tx_{id} of other content transactions, which means voting or reviewing the content transaction.
- $\boldsymbol{Sign}(\boldsymbol{sk}, \boldsymbol{m}) \rightarrow \boldsymbol{\sigma}$. The $Sign(sk, m)$ function is used to signature message m and outputs σ with a private key sk.
- $\boldsymbol{Verify}(\boldsymbol{\sigma}, \boldsymbol{pk}, \boldsymbol{m}) \rightarrow \{\boldsymbol{0}, \boldsymbol{1}\}$. The $Verify(\sigma, pk, m)$ function is used to verify that σ is the output of the user pk after signing the m. If the output of the function is 1, the validation is successful; otherwise, it is not successful.
- $\boldsymbol{weight}(\boldsymbol{pk_i}) \rightarrow \boldsymbol{w_i}$. The weight function outputs the weight w_i of pk_i. The weights of members without smart contract certification are 1.
- $\boldsymbol{Maxvaild}(\boldsymbol{C}) \rightarrow \boldsymbol{C_i}$. The $Maxvaild(C)$ is used to choose the best chain among multiple valid chains, where $C = \{C_1, C_2, \ldots, C_n\}$ is the set of valid chains in the network, and C_i is the most efficient chain of rules. The standard of choice is the heaviest and longest principle [24].

- $\boldsymbol{Mode(B_{hash}, P, I, rm) \rightarrow Tx\{Tx_1, Tx_2, Tx_3, \ldots, Tx_p\}}$. The $Mode(B_{hash}, P, I, rm)$ function is used to output a set of content transactions that allowed PoRe consensus in a round, where B_{hash} is the hash of the current latest block, P is the number of content transactions allowed for PoRe consensus this turn, rm is a random number that is generated by "iterated majority function" [25] to prevent stake grinding attacks and I is the width of the immature part.

4.2 Proof of Reputation

The essence of the blockchain consensus process is to maintain the same status of the ledger on different nodes. At each consensus round, an eligible node is selected to update the blockchain ledger. The problem is which one should be selected. In RPchain, the consensus principle is to choose a user who makes positive contributions to the system, rather than those who have rich physical resources. Moreover, the model represents the contribution of a user by its reputation. Therefore, we propose the proof of reputation (PoRe) for consensus. Particularly, PoRe employs the reputation gained by the content transactions sent by users in the system to reduce the difficulty of PoW. This avoids the needs of ASIC mining machines and the risk of centralization. We assume that a highly reputed user will honestly manage the blockchain.

PoRe-Based Proof of Work. The formal expression of PoRe consensus is:

$$H(state(), nonce) < F(R_i, D)$$

The output of the $F()$ function is the target value that needs to be satisfied, and it is determined by the current input reputation value R_i and the current PoW difficulty D. The hash function $H()$ hashes the header information and the nonce of the random number. Once the generated hash value is smaller than the current target value, it is considered as a successful PoRe certificate. The larger the reputation value of the content transaction used to prove, the larger the target value of $F()$. This means that the probability of getting the next block in the same physical condition increases versus the increase of the reputation. Although PoRe has similar advantages to the PoS consensus, it is essentially different from PoS. PoS determines the difficulty of calculation according to the number of tokens, whereas the difficulty of PoRe is determined by the reputation of the content transaction sent by the user.

4.3 Blockchain Protocol

The blockchain protocol determines how nodes reach consensus and update ledgers in the network. Each node creates its own cryptographic pseudonym in the blockchain network and uses it as the identity to receive and send messages in the blockchain network. The miners use the allowed consensus to take ownership of new blocks and packing transactions in the pool into mini blocks. The miner then broadcasts the latest blockchain status to the network. Other nodes listen to the network and update the blockchain to agree on the status. The specific protocol is shown in Table 1.

Table 1. The blockchain protocol

1. The node uses $KeyGen()$ to generate a key pair (pk, sk). Then the node can send $Tx()$ and sign it with $Sign()$.The miners' running protocol monitors the status of other nodes and updates the local blockchain to the latest chain.
2. The miners use the corresponding consensus algorithm to compete for the ownership of the next block. The protocol allows different consensus algorithms to be used in different phases.

a) PoW consensus. The simplified expression of PoW consensus is as follows:

$$H(state(), nonce) < 2^{256}/D$$

The $state()$ is the state snapshot of the current block, including the timestamp, the target value, the hash of the previous block, and so on. D is the current PoW difficulty value, with the increase of the total computing power in the network. And $nonce$ is a random number. The entire PoW is a process that uses hashing to continually find a $nonce$ that meets the above inequality.

b) PoRe consensus. Miner m_i uses the $Mode()$ function to get the current set of transactions for the PoRe consensus. With the same blockchain status, all miners using the $Mode()$ function in this round will get the same set of content transactions. If this set does not contain the content transaction signed by m_i, m_i cannot participate in the PoRe consensus this round. Conversely, if the set contains at least one content transaction signed by m_i, m_i can choose one of them to participate in the PoRe consensus.

$$H(state(), nonce) < F(R_i, pk_i, Tx_{id}, D)$$

Where R_i is the reputation value of the content transaction with Tx_{id} issued by pk_i, specifically expressed as follows:

$$R_i = \sum_{j=1}^{n} weight(pk_j)$$

n is the number of nodes voting on this content transaction, and the weight function $weight(pk_j)$ outputs the voting weight of the member pk_j. For function $F()$, the larger the value of R_i, the greater the chance of getting the next block.

c) PoS consensus. The more shares a node holds in the system and the longer it takes to hold the shares, the greater the probability to get the next block.

$$H(state(), nonce) < F(token, t, D)$$

In the above, $token$ is the number of tokens used for this consensus, t is the time lag from the begin of the node to hold these shares to the current time.

3. Once a miner successfully calculates a $nonce$ that meets the above inequality, it broadcasts a block and the latest blockchain to the network. Then it uses $Verify()$ to verify the transactions in the transaction pool. The verified transactions are packaged into the mini block. This process continues until the next block is found.
4. After receiving a block, the node verifies the validity of the block and the transactions contained in the mini blocks, add these blocks, and update the local chain.
5. If there are multiple valid chains in the current network, that is, a fork occurs, use $Maxvaild()$ to select the most efficient chain and the transactions in other chains will be replaced in the transaction pool.

5 Incentive Mechanism

Whenever a new block is mined in the blockchain, some new tokens are generated to reward the node that mines the block. Not only does this motivate nodes to actively participate in the maintenance of the blockchain, but also eliminates the need for third-party issued tokens. In the RP chain, tokens are used to pay for the consumption of smart contract complexity or as bonuses for some valuable questions. The block holder receives a certain amount of token rewards and transaction fees as a reward for maintaining the blockchain. In particular, the block reward using PoRe consensus is calculated by:

$$R = \frac{R_I}{N(Tx_{id})},$$

where R_I is the initial reward and $N(Tx_{id})$ is the number of times the content transaction Tx_{id} was used for successful consensus. This indicates that the reward of the new block R is inversely proportional to the number of times (n) the content transaction Tx_{id} is used for PoRe consensus. This prevents malicious nodes from reusing the same content transaction for PoRe consensus. The purpose of charging transaction fees is to motivate miners to package transactions into the block and limit Dos attacks.

In order to encourage members to vote, we introduce the voting rewards. If a node successfully uses a content transaction to find a new block with the PoRe consensus, members who participate in voting the content transaction will split a fixed voting reward according to the weight of the vote. One issue is that members may only vote for content transactions that already have many votes in order to increase their chances of getting a voting rewards. This reduces the quality of community content and affects the security. To address this issue, we calculated the voting rewards R_v got by member pk_i as:

$$R_v = \frac{R(t_i - t)weight(pk_i)}{\sum_{i=1}^{m}(t_i - t)weight(pk_i)},$$

where $weight(pk_i)$ is the voting weight of the member pk_i, R is the fixed voting award, m is the number of members voting on the content transaction, t_i is the time when pk_i votes for the content transaction and t is the release time of the content.

In RPchain, we allow miners to include uncle blocks in new blocks to solve the effects of wasted computing resources and security factors. Including uncle blocks in the block will get extra token rewards. The included uncle blocks will also get some token rewards to compensate for the consumption of computing power. The inclusion of uncle blocks in the block can reduce resource consumption, increase the security of the main chain, and further promote de-centralization of the chain.

6 Experiment

We implement a prototype of RPchain and assess its security and consensus efficiency. We build a testbed of 100 nodes using 10 physical computers, with each running 10 virtual machines. We do not consider the impact of propagation delay, and the transactions on each node are prepared in advance.

In the extension phase, the model uses the PoRe consensus and the security of the model may be impacted by two parameters: the proportion δ of malicious members and the proportion ε of content transactions generated by malicious nodes in the initial phase. Since our model employs the PoW consensus in the initial phase, the parameter δ to some extent represents the proportion of computation power of malicious nodes throughout the network. As the model uses the $Mode()$ function to randomly select a set of content transactions, the parameter ε represents the probability that a malicious node is selected to participate in the PoRe consensus. In order to evaluate the security of the model, we examine the proportion θ of malicious members in the extension phase under different values of δ and ε.

In the experiments, we employ the Chinese restaurant process (CRP) [26] to simulate the voting process. Imagine that a member reviews the content in the community and randomly vote on a content transaction. The voted content transactions are chosen according to the following random process:

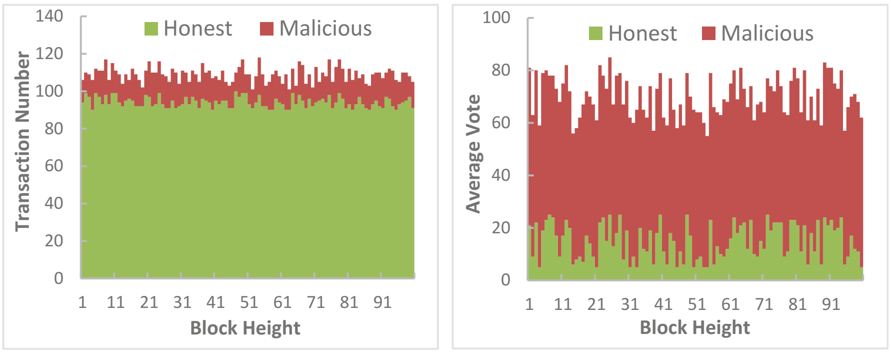
1. The first member always votes on the first content transaction.
2. The n-th member votes on an unvoted content transaction with probability $\frac{\alpha}{\alpha+n-1}$, and votes on a voted content transaction with probability $\frac{c}{\alpha+n-1}$, where c is the number of votes received by the content transaction.
3. If member votes on many content transactions, we view it as a different CRP process. However, the maximum number of votes received by a specific content transaction is limited by the number of members.

In the above, α is a scalar parameter of the process. The larger the α, the more likely that a member vote on an unvoted content transaction. The detailed experimental settings are described as follows:

1. The initial phase includes 120 blocks (The genesis block not included), and the immature part width is 20. This indicates that the $Mode()$ function will only get the content transactions in the first 100 blocks (the mature part) when the model firstly enters the extension phase. That is, the 121st block will be mined by nodes according to the PoRe consensus.
2. In the experiment, each block contains about 110 content transactions. We adjust the probability that a node sends content transactions in each round to limit the number of content transactions in each block. The set size output by $Mode()$ is set to 200. Currently, we do not consider certified scholars and all members have the same vote weight of 1.
3. Since CRP cannot simulate the behavior that malicious members are more likely to vote on malicious content transactions, we set a fixed probability p that a malicious member vote on a malicious content transaction. In the experiment, $p = 0.9$. We use CRP to simulate the voting process of honest members in the initial phase. If there

are N content transactions at the time of the simulation, we only use the first N values of a CRP result. In each simulation, we set $n \approx 120 \times 110 \times (1 - \delta)$, $\alpha = 10$.

Figure 4 shows the blockchain status in the initial phase with a parameter configuration of $\delta = 50\%, \varepsilon = 10\%$. Figure 4(a) presents the stacked bar plot of the content transactions publicized by honest and malicious nodes. It is shown that the content transactions publicized by malicious nodes occupy about 10% of the transactions in each block (recall that each node has a probability to send a content transaction). Figure 4(b) presents the stacked bar plot of the average number of votes received by content transactions publicized by honest and malicious nodes in each block. It is shown that content transactions publicized by malicious nodes receive much more votes than that publicized by honest nodes. However, when the model enters the extension phase, as illustrated by the solid line A in Fig. 5, the proportion θ of malicious members decreases stably with the growth of the blockchain. This is because more honest nodes are selected as members because their content tractions are more likely to be selected by the *Mode*() function. The results show that malicious nodes in the system are limited by both the computation power and the number of the content transactions. We also conduct experiments with different parameter configurations and obtain similar results. The situation will be even better if we take into account certified scholars.



(a) The number content transactions (b) The average number of votes

Fig. 4. The blockchain status in the initial phase ($\delta = 50\%, \varepsilon = 10\%$)

In this experiment, we compare the performance of the Bitcoin-NG structure with that of the ordinary blockchain structure. We evaluate the efficiency by measuring their consensus delays. The consensus delay is the best point-consensus-delay the system achieves for a certain fraction of the time, on average [24]. More formally, the (ρ, τ) consensus delay of a system is the ρ-percentile τ-point-consensus-delay. For example, if 90% of the time, 50% of the nodes agree on the state of the state machine 10 s ago (but not less than that), then the (50%, 90%)-consensus delay is 10 s. The consensus delay is affected by the size and generation time of a block. In this experiment, we take

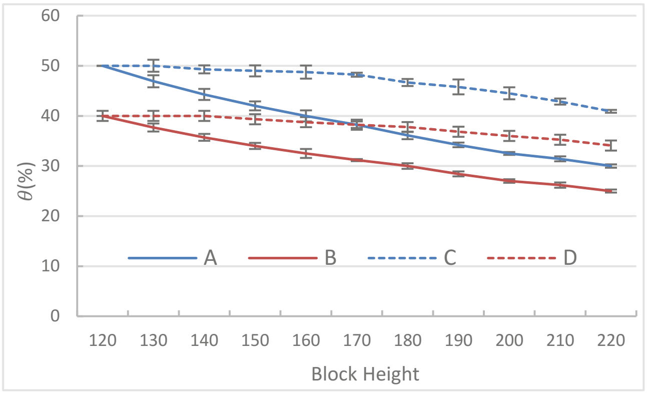


Fig. 5. The blockchain status in the extension phase (A: ε = 10%, δ = 50%; B: ε = 10%, δ = 40%; C: ε = 40%, δ = 50%; D: ε = 40%, δ = 40%;)

the (90%, 90%)-consensus delay based on block generation times. Then we set the block average generation time to 1 s, which means that in the model using the Bitcoin-NG's structure, an average of 1 s to generate a key block. We adjust the generation rate of the mini-blocks in order to keep the throughputs of the two models with different structures the same. The experimental results are shown in Fig. 6. It is shown that the consensus delay increases with the increase of block size. However, the consensus delay of the model using Bitcoin-NG's structure is lower than that using the general blockchain structure. This indicates that our RPchain is more efficient than the common blockchain model in terms of consensus efficiency.

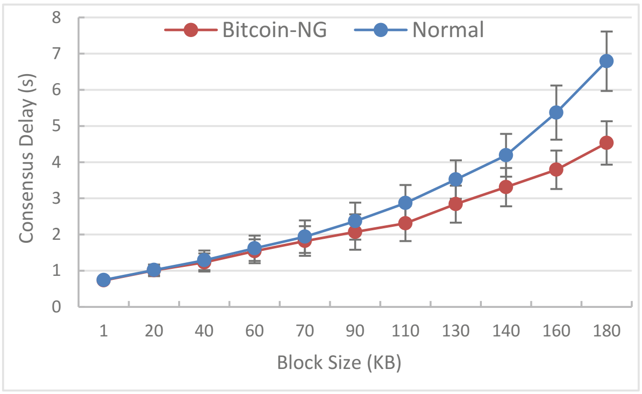


Fig. 6. Evaluation of consensus delay

7 Conclusion

In this paper, we present RPchain, a blockchain based academic social networking service model. Three kinds of transactions are designed to build user reputation based on other users' reviews. In order to maintain the virtuous cycle of RPchain, we propose a novel service-oriented consensus algorithm which exploits users' reputation to consensus. We also design appropriate rewards to incent the contributions of users to the community. Finally, we implement a prototype of the model can conduct experiments to evaluate the security and efficiency of RPchain. The experimental results show that our system is secure as long as the proportion of malicious members is less than 50% and the proportion of content transactions generated by malicious members is less than 40%. In addition, our model is more efficient than the general blockchain structure.

Acknowledgement. The research presented in this paper is supported in part National Natural Science Foundation of China (No.61602370, 61672026, 61772411), Postdoctoral Foundation (No.201659M2806) of China, Fundamental Research Funds for the Central Universities (No.1191320006).

References

1. Nicholas, D., Herman, E., Jamali, H.R.: Emerging Reputation Mechanisms for Scholars. European Commission, Institute for Prospective Technological Studies (2015)
2. Nosek, B.A., et al.: Promoting an open research culture. Science **348**, 1422–1425 (2015)
3. Pearce, N., et al.: Digital scholarship considered: How new technologies could transform academic work. Education **16**(1) (2012)
4. Greenhow, C., Robelia, B., Hughes, J.E.: Learning, teaching, and scholarship in a digital age: Web 2.0 and classroom research: what path should we take now? Educ. Res. **38**(4), 246–259 (2009)
5. Jamali, H.R., Nicholas, D., Herman, E.: Scholarly reputation in the digital age and the role of emerging platforms and mechanisms. Res. Eval. **25**, 37–49 (2015)
6. Egghe, L., Bornmann, L.: Fallout and miss in journal peer review. J. Doc. **69**(3), 411–416 (2013)
7. Fitzpatrick, K.: Peer to peer review and the future of scholarly authority. Soc. Epistemol. **24** (3), 161–179 (2010)
8. Zuckerman, H., Merton, R.K.: Patterns of evaluation in science: institutionalisation, structure and functions of the referee system. Minerva **9**(1), 66–100 (1971)
9. ResearchGate. https://www.researchgate.net. Last accessed 1 Mar 2018
10. Mendeley. https://www.mendeley.com. Last accessed 1 Mar 2018
11. Nicholas, D., Herman, E., Clark, D.: Scholarly reputation building: how does researchgate fare. IJKCDT **6**(2), 67–92 (2016)
12. Jeng, W., He, D., Jiang, J.: User participation in an academic social networking service: a survey of open group users on Mendeley. JAIST **66**(5), 890–904 (2015)
13. Kieslinger, B.: Academic peer pressure in social media: experiences from the heavy, the targeted and the restricted user. First Monday **20**(6) (2015)
14. Ford, E.: Defining and characterizing open peer review: a review of the literature. J. Sch. Publishing **44**(4), 311–326 (2013)

15. Sharples, M., Domingue, J.: The Blockchain and Kudos: a distributed system for educational record, reputation and reward. In: Verbert, K., Sharples, M., Klobučar, T. (eds.) EC-TEL 2016. LNCS, vol. 9891, pp. 490–496. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45153-4_48
16. Dennis, R., Owenson, G.: Rep on the roll: a peer to peer reputation system based on a rolling blockchain. Int. J. Dig. Soc. (IJDS) **7**(1), 1123–1134 (2016)
17. Schaub, A., Bazin, R., Hasan, O., Brunie, L.: A trustless privacy-preserving reputation system. In: Hoepman, J.-H., Katzenbeisser, S. (eds.) SEC 2016. IAICT, vol. 471, pp. 398–411. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-33630-5_27
18. Konforty, D., et al.: Synereo: The Decentralized and Distributed Social Network (2015). Self-published
19. Steem. https://steem.io. Last accessed 1 Mar 2018
20. Eyal, I., Sirer, E.G.: Majority is not enough: Bitcoin mining is vulnerable. In: International Conference on Financial Cryptography and Data Security, pp. 436–454. Springer, Heidelberg (2014)
21. Kwon, Y., et al.: Be selfish and avoid dilemmas: Fork After Withholding (FAW) attacks on Bitcoin. In: CCS. ACM (2017)
22. Rosenfeld, M.: Analysis of bitcoin pooled mining reward systems. arXiv preprint arXiv: 1112.4980 (2011)
23. Ripple. https://ripple.com. Last accessed 1 Mar 2018
24. Eyal, I., et al.: Bitcoin-NG: a scalable blockchain protocol. In: NSDI (2016)
25. Bentov, I., Gabizon, A., Mizrahi, A.: Cryptocurrencies without proof of work. In: International Conference on Financial Cryptography and Data Security, pp. 142–157. Springer, Heidelberg (2016)
26. Blei, D.: COS 597C: Bayesian nonparametrics. Lecture Notes in Princeton University. https://www.cs.princeton.edu/courses/archive/fall07/cos597C/scribe/20070921.pdf

IPFS-Blockchain-Based Authenticity of Online Publications

Nishara Nizamuddin, Haya R. Hasan, and Khaled Salah(✉)

Department of Electrical and Computer Engineering,
Khalifa University of Science, Technology and Research, Abu Dhabi, UAE
{nishara.nizamuddin, haya.hasan,
khaled.salah}@kustar.ac.ae

Abstract. In this paper, we propose a solution to provide originality and authenticity of published and posted freely online digital content such as books, music, and movies. Our solution utilizes a blend of newly emerging technologies that primary include (InterPlanetary File System) IPFS and blockchain smart contracts. IPFS is used to store digital content with a high integrity and global accessibility to all, and Ethereum smart contract is used to govern, manage, and provide traceability and visibility into the history of digital content from its origin to the latest version, in a manner that is decentralized and globally accessed with high integrity, resiliency, and transparency. In the paper, our solution is focused on online book publication, but the solution can be a framework that can be easily extendible and adoptable for, to other digital and multimedia content. The full code of our smart contract is provided, with discussion on implementation and testing of its key functionalities.

Keywords: Ownership authenticity · Originality · Online publishing
Blockchain · Ethereum · Smart contracts

1 Introduction

The Internet and the digital era have unleashed the unique access to information. With the increased ease in information access and sharing, the authenticity of freely posted and published digital materials is always questionable. The authenticity of digital content is a major challenge for today's online book publishing industry, and digital content in general, as those of multimedia, movies, music, etc. Digital content, available on the internet, during its lifetime can be modified, copied, reproduced, translated into different languages, re-published, and reformatted. There is an immense need for an appropriate authenticity with the ability to trace and track the publication history of posted online material to the original author, writer, or artist, with high degree of trust, credibility, and integrity.

In practice, a hardcopy manuscript of a book or newspaper article can be printed, scanned, digitized, and translated into different languages—resulting in multiple versions of the original manuscripts which were published by different publishing entities or individuals. That is, digital content available across various resources such as online journals, e-books, and websites can indeed be subjected to illegitimate alteration that ultimately leads to tainted information access. Also, there is a lack of a strict audit to ensure that the digital book is verifiable, complete and accurate. While e-book is collated and printed from diverse sources, the authenticity and integrity of digital asset is at stake.

Problem Statement. *To date, there exists a lack of authenticity and integrity of digital content made available online. Freely posted and published online digital contents are not tampered-proof and their publication history cannot be easily tracked in a credible, trusted, open, and decentralized manner.* As shown in Fig. 1, a book can be published, and re-published by different publishing entities, and thus producing multiple versions of the original book. The book is originally written by the author which can be made available to public users via various resources such as handwritten manuscript, physical print media, e-books and Internet sources. The credibility of a digital document cannot be checked at any point as the publishers are not accountable for the content published. Furthermore, the readers are totally unaware about the accuracy and authenticity of the available e-book. Typical readers usually accept the online versions of digital manuscripts despite being tampered with.

Figure 1 shows the traditional way of public readers accessing digital assets of e-books from available resources. The original work of the author undergoes various stages of publication process before it reaches the end user i.e., the readers. Typically, author writes the book and chooses a main publisher to submit his original work. The main publisher is granted the publishing permission from the author. Figure 1 illustrates the scenario where secondary publishers (P_1, $P_2 \ldots P_n$) request for publishing permission in different versions from the main publisher. The main publisher P grants permission to the requesting publishers upon agreeing to the terms and conditions are accepted by both the parties. The same book can have many editions and versions with different versions being translated in many languages. In today's book publishing industry, a certain online book version cannot be traced back to an original author, as information is usually defragmented and not available to all readers to verify and examine the authenticity and originality of published content.

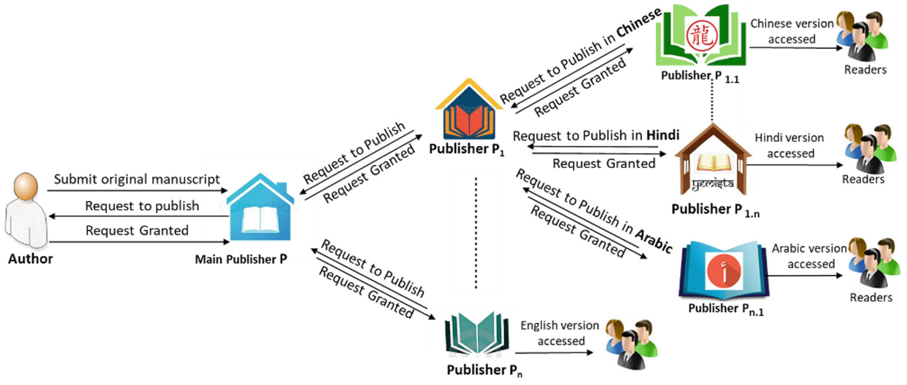


Fig. 1. Production of different versions of an online book through various publishers

Blockchain is a newly emerging and disruptive technology that can be key in providing a solution to authenticity of digital materials. Blockchain is the underlying technology of the cryptocurrency bitcoin, but now is seen as a distributed ledger that can be accessed globally by anyone to verify stored data and content, with high integrity, resiliency, credibility, and above all traceability [1]. All of this is done in de-centralized manner and without intermediaries. Later, Ethereum smart contracts provided the ability to upload and execute code that carries out business logic to the blockchain [2, 3]. The smart contract code resides on a blockchain as multiple functions with unique addresses that can be called by any user of the blockchain.

Blockchain, however is an expensive medium for data storage, especially for large data and digital content. For efficient storage of large data and content, we propose using IPFS file system. IPFS stands for Inter-Planetary File System [4], which is a distributed, decentralized file system and a platform to store data and files with high integrity and resiliency. Fundamentally, IPFS is a peer-to-peer, open source, content addressable globally distributed file system that can be used for storing and sharing large volume of files with high throughput. Our proposed solution makes use of both blockchain smart contracts and IPFS, whereby the digital contents are stored on the IPFS and the IPFS hashes are stored into the blockchain smart contracts to provide traceability and authenticity. Specifically, the hash generated on storing the documents to IPFS, can be stored in the smart contracts effectively and documents can be accessed using the hash. If there is any change in the content of the digital document, the hash changes, to show that the original content was modified and altered.

In this paper, we propose a combined IPFS-blockchain-based solution to solve the authenticity and originality of digital content posted freely on the Internet. In the paper we show how this problem can be solved for online published books, but our solution can be extended and adopted for, to other digital and multimedia content. We show how our solution has the ability to trace and track the digital content, with its different published versions, back to the certified true copy created by the original author. The main contributions of this paper can be summarized as follows:

- We propose an IPFS-blockchain-based solution and framework for providing authenticity of published online books and digital content. Our solution provides decentralized storage and governance with different versions of the original book being stored, tracked, and traced with high integrity, and resiliency.
- We present and discuss the system design and architecture, along with sequence diagrams to illustrate the interactions among participants which include author, main publishers, secondary publishers, and readers.
- We provide the full code of the smart contract, and discuss key implementation and testing details to demonstrate the proper operations and functionality of the overall system.

The remainder of this paper is organized as follows. Section 2 summarizes the related work. Section 3 presents our proposed solution for improving authenticity of online books. Section 4 describes key aspects of the implementation and testing of the smart contract. Section 5 concludes the paper.

2 Related Work

In this section, we provide a brief background on the existing approaches found in the literature related to authenticity and originality of books in the publishing industry using blockchain technology. Ericsson [5] proposes a blockchain-based system for tracking the origin of digital assets. It is by converting the digital content and books into a binary file and to store the hash on blockchain. This hash is stored generally with an identifier for the owner. The paper focusses on the idea that ownership can be verified by checking the integrity of digital assets at any point of time. This is achieved through the verification by a centralized unit for security of digital documents such as Security Operations Center (SOC) to estimate the legitimacy of digital assets. But the system itself deviates from the decentralized concept as it operates in the presence of a centralized security unit and has a precarious state for breach of integrity. Moreover the file storage is done on a centralized server which can be a single point of failure, corruption, hacking or compromise.

Gaetani *et* al. [6] propose a verification ID which can be used for blockchain-based authenticity of digital assets. This ID is inherently a digital block on the blockchain that can be used for verification of the e-document. That is, whenever an ID is added to the blockchain, an identification certifying service combines the public key with the owner's ID and transfers the ownership of the private key to the user. A blockchain centered handshake mechanism is employed to ensure authenticity of the e-document. But this method suffers detrimental effect as it is purely based on the trust factor on the requesting entity. As the identity of the requesting body is not always reliable in the digital world, the system is not stable enough to provide a secure storage and access of documents online.

The author in [7] proposes a blockchain-based model for publishing online books and for providing integrity of the digital document. The author achieves authorship by storing the book/file hash and the owner's name in pairs. The author argues that by storing the hash of the file and the block timestamp as pairs, integrity of the document/file can be proved. If the content of the file was modified, then its hash will change, and the smart contract won't be able to access the file, therefore proving that the file content was modified. Sun *et* al. [8] describes a framework for evaluating the trust issues when storing online documents in decentralized networks. In this paper, authors present a framework to quantitatively measure trust, model trust propagation, and defend trust evaluation systems against malicious attacks. This system was used to secure adhoc routing and support to unmask malicious node in a decentralized environment but is not yet implemented as a real-world application.

The authors in [9] propose a blockchain-based personal data management system to ensure that document owners have complete authority over their asset. This model features a blockchain-based automated online document access control system thereby eliminating trust in a third party. Blockchain and off–blockchain storage is combined to construct a management platform which precedes to trust based computing. But the work does not describe about the feasibility of storing larger files. Morgan [10] presents the idea of using blockchain technology to prove the existence of a document using the timestamping concept. The author discusses a method where the document is presented to a site which in turn converts the document into a cryptographic hash. The hash generated represents the content of the document. If the original document is presented, same hash will be generated, notifying that the document is authentic. However, if there is any modification of content, the newly generated hash will not match with the previous hash. The legitimacy of the document can be verified, but this system is not focused about the authority of the owner on his/her document.

Acronis Notary system described in [11] is a blockchain-based notary service which aims at providing a solution for timestamping digital documents. As blockchain is a very expensive storage medium for storing large documents, the proposed approach is to send file hashes to the Notary service. This service calculates hash value, based on the received file hashes and saves the new hash obtained, on the Ethereum network. A verification certificate is provided specifying the technical details of the document. Whenever the document is reflected in user interface, it is shown as 'notarized' or 'certified' by Acronis. By doing so, the system gives an assurance to the user that the online document is identical to the original version, on a bit-by-bit basis. This method supports providing notarization for document authenticity and a certification that an e-book existed at some point of time in the chain. But it doesn't clearly state about the author's rights to claim the ownership of digital book in the decentralized environment.

3 Proposed Solution

Our solution is based on using IPFS and smart contracts of Ethereum blockchain. IPFS is used to store the digital content (of the e-book or multimedia files) in a decentralized, distributed manner that is publicly and globally accessible by all through the use of IPFS hashes. This IPFS hash is used by the smart contract of Ethereum blockchain to ensure integrity, originality, and authenticity. The hash value remains the same if the content of the document or e-book remains intact. If there is an alteration of content during the publication stages, the IPFS hash for the book changes, and would then not match the hash stored within the smart contract. Therefore, each participating entity can track back, and verify the accuracy and history of e-books being stored in the file system and be assured that the book accessed is a legitimate copy of the author's work.

3.1 System Architecture and Design

Figure 2 illustrates the overall system architecture and design for automating the online books authenticity, originality, and integrity using IPFS and Ethereum smart contracts. The proposed solution uses smart contracts to trigger events that are logged to notify the participating parties to keep track of events and transaction details. The figure highlights the interactions of the smart contract with main participants that include author, main publisher P, secondary and other publishers, and readers. The participants of the smart contract can be summarized as follows:

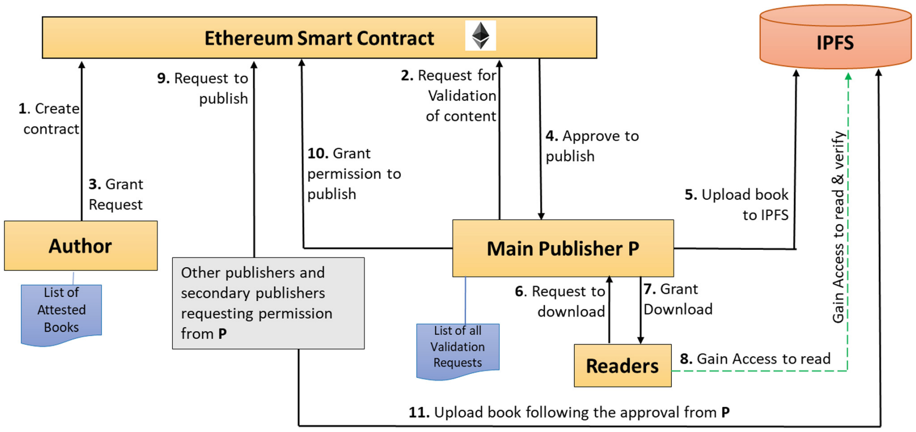

Ethereum Smart Contract
IPFS
1. Create contract
3. Grant Request
9. Request to publish
10. Grant permission to publish
2. Request for Validation of content
4. Approve to publish
5. Upload book to IPFS
Gain Access to read & verify
Author
List of Attested Books
Other publishers and secondary publishers requesting permission from P
Main Publisher P
List of all Validation Requests
6. Request to download
7. Grant Download
Readers
8. Gain Access to read
11. Upload book following the approval from P

Fig. 2. An overview of the system architecture for automating the online books authenticity using IPFS and Ethereum smart contracts.

- **Author**: The author or artist is the person who owns the IP and original work of the book or digital content. The author creates the smart contract and provides permission for one or more publishers to publish his digital content. The main publisher is also involved in validating and notarizing the book content presented to it by the publishers. The publisher gets to upload the content on IPFS, only if the notarization by author is successful. The author also maintains the original hash of the book.
- **Publisher**: The publisher is the entity which obtains permission from the author to publish the manuscript content in various sources such as web pages, printed books and e-books. The main responsibility of a publisher is to maintain the digital content intact as presented by author. Also, a publisher uploads the digital content to IPFS and the IPFS hash is stored in the smart contract with the author's validation and verification.
- **Other and Secondary Publishers**: A book can be translated into different languages and can have different versions or editions produced by different publishers who produced the book in agreement with the original author or with the main publisher.
- **Readers**: Readers are those who request access for legitimate digital books. The smart contract provides the readers with trace back functionality to verify the originality of the book. The readers can access the history of notarizations regarding the originality, authenticity, and integrity of the book.

In our proposed solution, the author initially writes the book, creates the contract which includes key attributes about the book to include book title, original hash of the book and author details. Offline, a main publisher seeks publishing permission from the author after finalizing publishing terms. The publisher, then requests for an approval from the original author before uploading the book to IPFS, which returns a hash. The author then examines the stored IPFS file (with the given IPFS hash) to the original content submitted to the main publisher. The author concludes that the digital manuscript is uncorrupted and attests the publisher copy. The author has a list of books he attested, and also the publishers have a list of book versions attested by them.

Readers or other secondary publishers can also request for content validation history, to know about authenticity of digital manuscript. Once the book is made available on the IPFS, a reader or a secondary publisher can request for history of validation proof. This can be needed to make sure that the document has the notarization of the main publisher from whom publishing permission was granted and the attestation of the original author who validated for the main publisher to publish the content. The smart contract has the ability to display the list of attestations for that the e-book along with the address of the requesting entity, address of the publisher who provided access for the document and the address of the original author. The validation history helps to track and trace back the history of attestation provided for the digital documents.

4 Implementation and Testing

Our smart contract was implemented and tested using Remix IDE http://remix.ethereum.org. In this section, we provide the implementation details and focus primarily on testing the correct interaction and functionality among system participants. Remix IDE offers rich features that make it possible to test and debug smart contracts prior to deploying them.

4.1 Implementation Details

The code was written in Solidity using the web browser-based IDE, Remix. There are three entities participating in the contract, author, publisher, readers/secondary publishers. Each of the entities has an Ethereum address and can participate by calling functions within the smart contract at certain time stamps. Figure 3, illustrates the message sequence diagram for granting publishing permission with a successful attestation and a failed attestation by author. It shows the interaction between publisher, author and smart contract.

Figure 4 represents the message sequence diagram for successful and failed trace back of validation history between the requesting readers or secondary publishers, main publisher and the author. Figure 4 illustrates the flow of `traceBackHistory()` function and events `ValidationHistorySuccess` and `FailedValidationHistory` on successful and failed trace back history of attestations respectively.

Next, we show the important code snippets of our smart contract. The smart contract code is available at: https://github.com/SmartContract1/DataAuthenticityForOnlinePublications. To track the state of publishers and their approval results we use mapping, which represents a key-value pair. We also maintain a mapping to record a list of approvals by author and the hash provided by publishers during attestation process. Figure 5 shows mapping where every Ethereum address points to the address of publishers who submitted request for approval and hashes provided by publishers. Figure 5 also shows a mapping which consists of a list of books approved by authors and a mapping consisting of state of publishers.

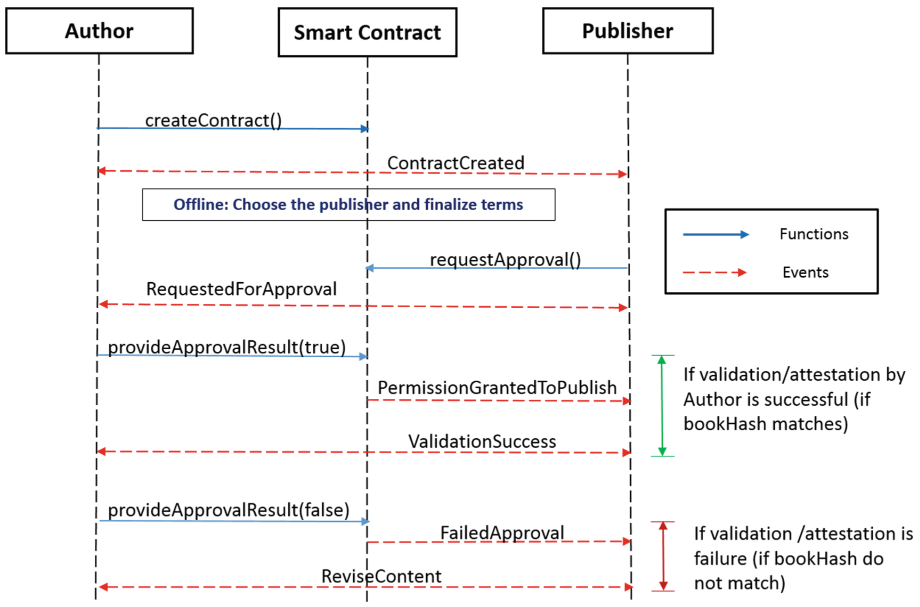

Author
Smart Contract
Publisher
createContract()
ContractCreated
Offline: Choose the publisher and finalize terms
Functions
Events
requestApproval()
RequestedForApproval
provideApprovalResult(true)
PermissionGrantedToPublish
If validation/attestation by Author is successful (if bookHash matches)
ValidationSuccess
provideApprovalResult(false)
FailedApproval
If validation /attestation is failure (if bookHash do not match)
ReviseContent

Fig. 3. Message sequence diagram showing scenarios of successful validation and a failed validation.

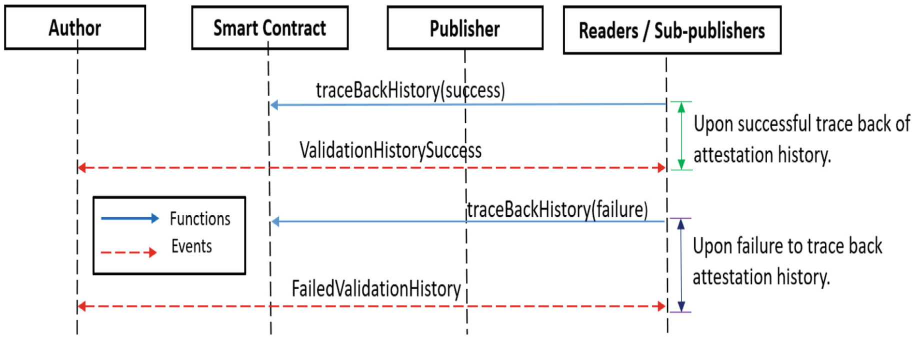
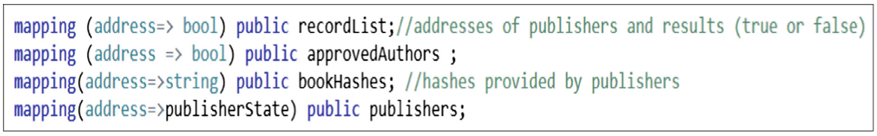

Author
Smart Contract
Publisher
Readers / Sub-publishers
traceBackHistory(success)
Upon successful trace back of attestation history.
ValidationHistorySuccess
traceBackHistory(failure)
Functions
Events
Upon failure to trace back attestation history.
FailedValidationHistory

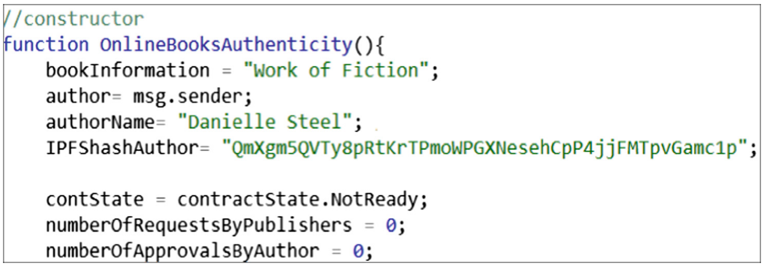
Fig. 4. Message sequence diagram showing scenarios of successful validation trace back and a history of failed validation trace back.



```
mapping (address=> bool) public recordList;//addresses of publishers and results (true or false)
mapping (address => bool) public approvedAuthors ;
mapping(address=>string) public bookHashes; //hashes provided by publishers
mapping(address=>publisherState) public publishers;
```

Fig. 5. Mapping of publishers and books attested by author (See footnote 1).

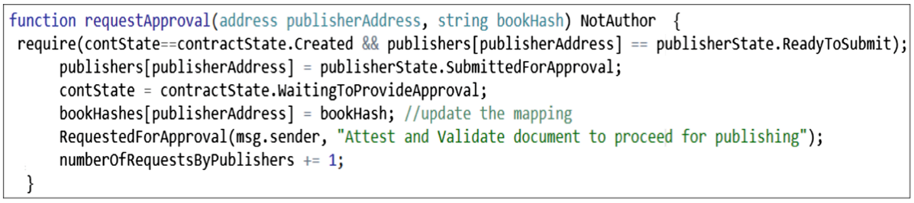
Figure 6 represents constructor `OnlineBooksAuthenticity()` which shows that author is the owner of contract. It comprises of initialization such as the `authorName`, `bookInformation` and most importantly consists of the original book hash. The state of contract initially is `NotReady`. The author creates the contract and executes the `CreateContract()`[1] function and invokes `ContractCreated` event.



```
//constructor
function OnlineBooksAuthenticity(){
    bookInformation = "Work of Fiction";
    author= msg.sender;
    authorName= "Danielle Steel";
    IPFShashAuthor= "QmXgm5QVTy8pRtKrTPmoWPGXNesehCpP4jjFMTpvGamc1p";

    contState = contractState.NotReady;
    numberOfRequestsByPublishers = 0;
    numberOfApprovalsByAuthor = 0;
```

Fig. 6. Constructor of the smart contract `OnlineBooksAuthenticity()` code (See footnote 1)

Figure 7 demonstrates the `requestApproval()` function for publisher requesting attestation from the author. At this stage, the state of the contract is `Created` and the publisher state will be `ReadyToSubmit`. The publisher state changes to `SubmittedForApproval` and the contract state changes to `WaitingToProvideApproval`. The event `RequestedForApproval` is triggered and the publisher waits for the approval results (True/False) from the author.



```
function requestApproval(address publisherAddress, string bookHash) NotAuthor  {
 require(contState==contractState.Created && publishers[publisherAddress] == publisherState.ReadyToSubmit);
      publishers[publisherAddress] = publisherState.SubmittedForApproval;
      contState = contractState.WaitingToProvideApproval;
      bookHashes[publisherAddress] = bookHash; //update the mapping
      RequestedForApproval(msg.sender, "Attest and Validate document to proceed for publishing");
      numberOfRequestsByPublishers += 1;
  }
```

Fig. 7. Smart contract function for publisher requesting attestation from author (See footnote 1)

In this paper, we have considered two scenarios based on the approval results provided by the author for every publisher requesting an attestation or validation before uploading the book to IPFS. We have considered two other scenarios based on the attestation history results requested by either readers/secondary publisher. The four scenarios are as follows:

[1] The code is available at: https://github.com/SmartContract1/DataAuthenticityForOnlinePublications.

1. If the book validation results i.e. `provideApprovalResult()` is "True", the transaction is successful, and the smart contract directs the publisher to upload the authentic book to the IPFS and the book is now freely available for readers to access.
2. If `provideApprovalResult()` yields "False", the author does not approve the online book content as the hash submitted by publisher while requesting for attestation does not match the original hash of the book which is stored in smart contract. The change in hash clearly indicates, that the original content was changed/modified. The author triggers an event soliciting the publisher to amend the content as per original document and to resubmit for attestation.
3. If the attestation history trace back i.e. `traceBackHistory()` is successful, the list of attestation made for the book is provided to requesting entity.
4. When `traceBackHistory()` is a failure, it indicates that the book accessed is not validated and hence the requesting entity is provided with the detail that the book is not an authentic work of author.

4.2 Testing and Validation

In this section, we test the correctness of the interaction among participants, and the correct functionality of the overall system. Testing of the smart contract ensured that the flow of the contract followed the correct sequence, validations and refusals of e-book submitted were tested correctly and the attestation history tack back functionality executed correctly. In this paper, the smart contract is tested for 4 different scenarios. Figure 8 shows the various testing scenarios carried out. The first test group is on the approval from author to publish the book. It is further classified for testing two scenarios – Validation success and Failed Validation. The second test group represents readers or secondary publishers requesting for attestation history details. Test case 2 has two scenarios: Successful traceability, and Failed traceability. All the test cases are discussed in detail in the next subsection.

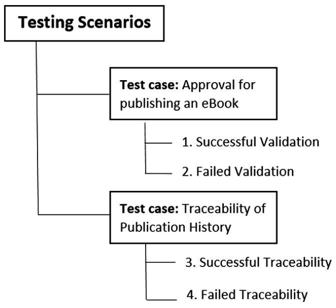
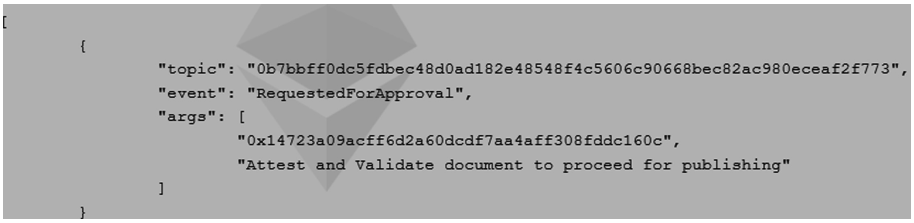

Testing Scenarios
Test case: Approval for publishing an eBook
1. Successful Validation
2. Failed Validation
Test case: Traceability of Publication History
3. Successful Traceability
4. Failed Traceability

Fig. 8. Smart contract testing scenarios



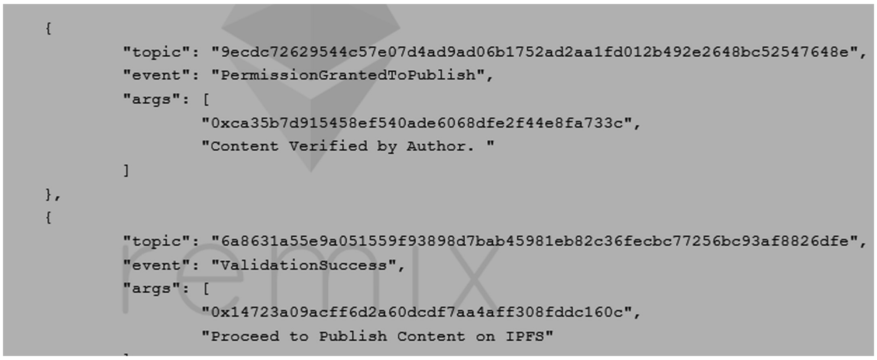
```
[
        {
                "topic": "0b7bbff0dc5fdbec48d0ad182e48548f4c5606c90668bec82ac980eceaf2f773",
                "event": "RequestedForApproval",
                "args": [
                        "0x14723a09acff6d2a60dcdf7aa4aff308fddc160c",
                        "Attest and Validate document to proceed for publishing"
                ]
        }
```

Fig. 9. Logs showing event `RequestedForApproval` triggered (See footnote 1)

4.2.1 Test Case 1

Firstly, we test the smart contract for a successful validation scenario. For testing purpose, we consider the Ethereum address of the author to be `0xca35b7d915458ef540ade6068dfe2f44e8fa733c`, the addresses of publishers who submit the request for attestation or validation to be `0x14723a09acff6d2a60dcd-f7aa4aff308fddc160c` for Publisher1 and `0x4b0897b0513fdc7c541b6d9d7e929c4e5364d2db` for Publisher2. We tested, as shown in Fig. 9, the case when the publisher request for a validation to publish from the author. The event `RequestedForApproval` is triggered and Publisher1 waits for the author to attest its content.

The original hash of the book is "`QmXgm5QVTy8pRtKrTPmoWPGXNesehCpP4jjFMTpvGamc1p`". Firstly, we test for the scenario of successful attestation/validation of book by the author with Publisher1. Upon successful attestation, events `PermissionGrantedToPublish` and `ValidationSuccess` are triggered and Publisher1 is granted the permission to upload the book on IPFS. Figure 10 shows the scenario of successful validation provided to Publisher1 by author.

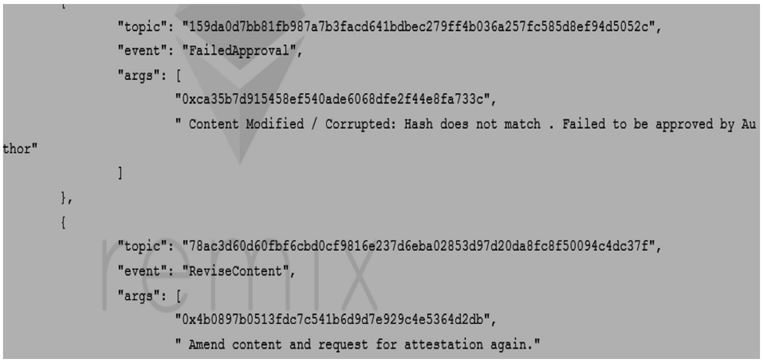


```
    {
            "topic": "9ecdc72629544c57e07d4ad9ad06b1752ad2aa1fd012b492e2648bc52547648e",
            "event": "PermissionGrantedToPublish",
            "args": [
                    "0xca35b7d915458ef540ade6068dfe2f44e8fa733c",
                    "Content Verified by Author. "
            ]
    },
    {
            "topic": "6a8631a55e9a051559f93898d7bab45981eb82c36fecbc77256bc93af8826dfe",
            "event": "ValidationSuccess",
            "args": [
                    "0x14723a09acff6d2a60dcdf7aa4aff308fddc160c",
                    "Proceed to Publish Content on IPFS"
```

Fig. 10. Logs showing a successful attestation (See footnote 1)

4.2.2 Test Case 2

Secondly, we test the smart contract for the scenario of a failed validation with Publisher2. The validation fails because of change in hash provided by Publisher2 while requesting attestation to publish. The author refuses approval and asks Publisher2 to resubmit the document after making the necessary amendment. We tested a case, where Publisher2 submitted a different hash value - "`QmYh1A5qcUXXAxEPaVmY7Frg-w6rGwyUqE5uc71ThtuoAUM`". Figure 11 shows that the approval failed, and the author recommends for amendment of content before resubmitting for attestation again. Events `FailedApproval` and `ReviseContent` are invoked to notify Publisher2 to amend the book content.

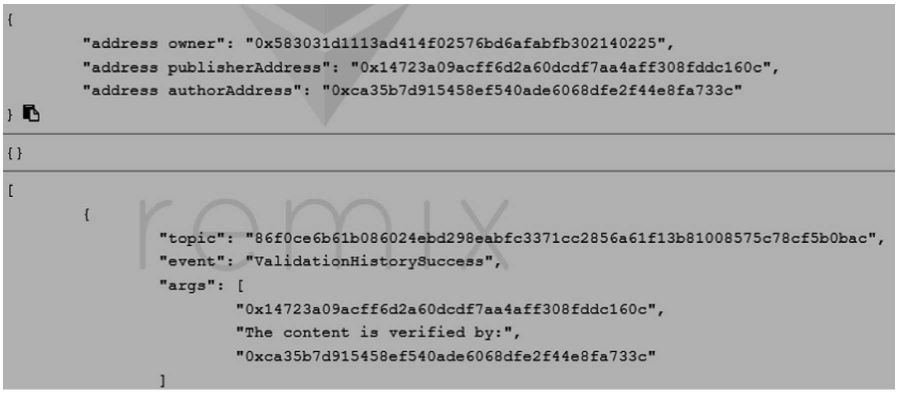


```
            "topic": "159da0d7bb81fb987a7b3facd641bdbec279ff4b036a257fc585d8ef94d5052c",
            "event": "FailedApproval",
            "args": [
                    "0xca35b7d915458ef540ade6068dfe2f44e8fa733c",
                    " Content Modified / Corrupted: Hash does not match . Failed to be approved by Au
thor"
            ]
        },
        {
            "topic": "78ac3d60d60fbf6cbd0cf9816e237d6eba02853d97d20da8fc8f50094c4dc37f",
            "event": "ReviseContent",
            "args": [
                    "0x4b0897b0513fdc7c541b6d9d7e929c4e5364d2db",
                    " Amend content and request for attestation again."
```

Fig. 11. Logs showing a validation failure with events `FailedApproval` & `ReviseContent` triggered (See footnote 1)

4.2.3 Test Case 3

Next, we tested for successful traceability of the attestation history of the document with a reader whose Ethereum address is `0x583031d1113ad414f02576b-d6afabfb302140225`. The reader requests to prove that the document is the original work of the author and has a notarization proof. Figure 12 represents the



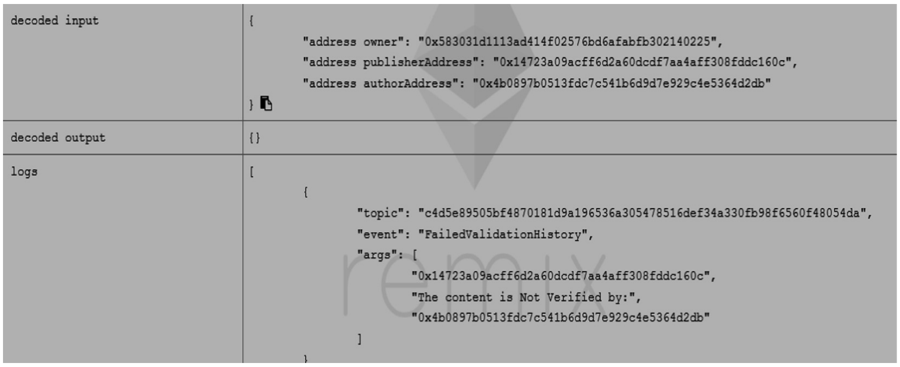
```
{
        "address owner": "0x583031d1113ad414f02576bd6afabfb302140225",
        "address publisherAddress": "0x14723a09acff6d2a60dcdf7aa4aff308fddc160c",
        "address authorAddress": "0xca35b7d915458ef540ade6068dfe2f44e8fa733c"
}
{}
[
        {
                "topic": "86f0ce6b61b086024ebd298eabfc3371cc2856a61f13b81008575c78cf5b0bac",
                "event": "ValidationHistorySuccess",
                "args": [
                        "0x14723a09acff6d2a60dcdf7aa4aff308fddc160c",
                        "The content is verified by:",
                        "0xca35b7d915458ef540ade6068dfe2f44e8fa733c"
                ]
```

Fig. 12. Logs showing event `ValidationHistorySuccess` triggered to show the content validation history of the book (See footnote 1)

scenario where the validation history of the document is obtained successfully displaying the address of Publisher1 from whom the reader obtained book access, the address of the original author. Event `ValidationHistorySuccess` is triggered and the smart contract assures the reader that the received copy is a legitimate work. The logs also show that the book was initially attested by author and the readers can confirm that the received book is authentic.

4.2.4 Test Case 4

Lastly, we test for the scenario of attestation trace back failure with `0xdd870-fa1b7c4700f2bd7f44238821c26f7392148`, the address of the secondary publisher who requests for attestation history trace back from another publisher `0x583031d1113ad414f02576bd6afabfb302140225`. Figure 13 shows the testing results. Event `FailedValidationHistory` is invoked as the attestation is not done by the actual author. This event indicates that the online book requested by the secondary publisher is not attested by the Publisher2 to whom the request for validation history was made.



```
decoded input     {
                          "address owner": "0x583031d1113ad414f02576bd6afabfb302140225",
                          "address publisherAddress": "0x14723a09acff6d2a60dcdf7aa4aff308fddc160c",
                          "address authorAddress": "0x4b0897b0513fdc7c541b6d9d7e929c4e5364d2db"
                  }
decoded output    {}
logs              [
                          {
                                  "topic": "c4d5e89505bf4870181d9a196536a305478516def34a330fb98f6560f48054da",
                                  "event": "FailedValidationHistory",
                                  "args": [
                                          "0x14723a09acff6d2a60dcdf7aa4aff308fddc160c",
                                          "The content is Not Verified by:",
                                          "0x4b0897b0513fdc7c541b6d9d7e929c4e5364d2db"
                                  ]
                          }
```

Fig. 13. Logs showing event `FailedValidationHistory` invoked (See footnote 1)

5 Conclusion

In this paper, we have proposed a solution to solve the originality and authorship authenticity of freely published and posted online books and documents using IPFS and Ethereum smart contracts. Our solution is focused on authenticity of online books, but the solution in terms of architecture, design, sequence diagram, logics, smart contract code, and overall aspects are generic enough to be easily extended and used to provide the originality and authenticity, as well as integrity, to all other forms of digital assets. We implemented and verified the functionalities of the smart contract code using Remix IDE. As a future work, we plan to deploy the smart contracts on the real IPFS and Ethereum network and develop frontend Decentralized Applications (DApps) with different views to authors, main publishers, secondary publishers, and readers.

References

1. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system. White paper
2. Toyoda, K., et al.: A novel blockchain-based Product Ownership Management System (POMS) for anti-counterfeits in the post supply chain. IEEE Access **PP**(99), 1 (2017)
3. 'Ethereum'. http://www.ethdocs.org/en/latest/introduction/what-is-ethereum.html/. Accessed 20 Dec 2017
4. Pors, M.: Understanding the IPFS White Paper part 2. https://decentralized.blog/understanding-the-ipfs-white-paper-part-2.html. Accessed 18 Mar 2018
5. Ericsson Home Page, Ericsson White Papers – "Industrial Blockchain and Data Integrity". https://www.ericsson.com/hyperscale/cloud-infrastructure/data-centric-security/data-integrity-assurance. Accessed 16 Feb 2018
6. Gaetani, E., Aniello, L., Baldoni, R., Lombardi, F., Margheri, A., Sassone, V.: Blockchain-based database to ensure data integrity in cloud computing environments. In: 1st Italian Conference on Cyber Security, ITASEC17, Venice, Italy, pp. 146–155 (2017)
7. Prusty, N.: Building Blockchain Projects – Develop Real-Time Practical DApps Using Ethereum and JavaScript, 1st edn. Packt Publishing Ltd., Birmingham (2017)
8. Sun, Y., Han, Z., Yu, W., Liu, K.J.R.: A trust evaluation framework in distributed networks: vulnerability analysis and defense against attacks. In: INFOCOM 2006. Proceedings of the 25th IEEE International Conference on Computer Communications, Spain, pp. 1–13 (2006)
9. Zyskind, G., Nathan, O., Pentland, A.: Decentralizing privacy: using blockchain to protect personal data. In: 2015 IEEE Conference Proceedings of the Security and Privacy Workshops (SPW), San Jose, CA, USA, pp. 180–184 (2015)
10. Morgan, P.: Using blockchain technology to prove existence of a document. https://bravenewcoin.com/news/using-blockchain-technology-to-prove-existence-of-a-document/. Accessed 20 Feb 2018
11. Acronis Inc.: Acronis Notary: a new way to prove data authenticity via blockchain. https://www.acronis.com/en-us/articles/data-protection. Accessed 21 Feb 2018

Blockchain Framework for Textile Supply Chain Management

Improving Transparency, Traceability, and Quality

Magdi ElMessiry[1] and Adel ElMessiry[2](✉)

[1] Textile Engineering Department, Faculty of Engineering, Alexandria University, Alexandria, Egypt
mmessiry@yahoo.com
[2] Garment Chain, Nashville, TN, USA
ammessir@ncsu.edu

Abstract. Modern textile supply chain systems are both large and complicated, with global sources and suppliers feeding into production lines that can span continents. A substantial amount of defects can't be directly traced back to defective batches that entered the supply chain along the way, causing waste and frustration downstream. Traceability is almost impossible due to the number of stages the product goes through and the size of data involved. No single system is globally utilized to record and trace the product throughout the supply chain. By the time the root cause of the issue is discovered, no recourse is possible except to discard the end product, resulting in losses that could reach 40% of the end product value. Communicating quality issues cross-stream is virtually nonexistent due to the challenges in identifying the source and recognizing that the other systems can deal with utilizing it. While traceability is an obvious problem in textile supply chain, transparency is a more impactful issue that is not well addressed. Cross supply chain and lack of transparency exacerbates the problems facing each participant and forces each entity to work locally using the localized information. This approach is fundamentally flawed as it deals with a global problem from a localized point of view. Not all industries are ripe for taking advantage of blockchain technology. Blockchain requires an industry with a complicated and widely distributed supply chain, containing an increased number of middle stages. This cannot apply more than in one of the world's oldest industries, textile.

In this paper, we propose a complete blockchain-based framework for textile quality improvement that enables in near real time, cross chain information sharing with guaranteed authenticity and accuracy allowing quality defective batches to be identified in all systems as soon as they are detected in any few.

Keywords: Blockchain · Supply chain · Textile · Garment chain

1 Introduction

To have an appreciation of the problem we are addressing, we need first to visit the current status in the textile industry and then provide an understanding of what blockchain technology is.

1.1 Textile Quality

In the textile industry, the low-cost players have forced many manufacturers to compromise on quality to be more cost competitive in the face of global competition. Currently, due to the inefficient quality monitoring systems, the cost of poor quality is at an average of 14% of sales for textile and apparel industries versus to other industries (6.5%) [1]. Several papers [4,6] attained the quality of various products flow in the textile production chain. The evaluation of the quality control principles through the application of statistical quality control, statistical processes control, total quality control, total quality management and Six Sigma are static systems. The dynamic quality control system was suggested but not fully developed.

Traditionally, high variability has been known to result in a substantial loss due to the existence of several nonconforming units in a highly variable process [22]. The On Line Quality Control System comprises with the raw material quality control and process. Control is the target for controlling the level of the quality in the production line [11,17]. Despite the advancements in technology and production-monitoring systems, these quality requirements seem to be a distant task as on-line systems for monitoring the quality of material at different stages of processing have not yet received due importance. One of the key issues for failure that has been identified is the incongruent application of current static control systems in the complex textile production environment [9,18,21,22].

The quality of the final product is a function of the qualities of the sub product that accommodate the complex, dynamic and interactive nature of the textile production environment. Consequently, the single stage control algorithms usually lead to loss of production, material and profit. It was revealed that the total quality of a product is a dynamical function and depends on the transfer function of the sequential process involved in its production [8,21]. To solve this problem, several attempts were successfully made to measure the quality parameters on line, such as trash %, number of neps, sliver evenness, yarn evenness, and real-time fabric inspection [3,4]. Thus, there is a need and requirement for development of new methods for modeling and automated monitoring of key parameters in the textile processing industry to optimize quality of the product, which consequently will improve profits.

1.2 Textile Supply Chain

We can view the textile supply chain in a simple view consisting of two main activities:

Primary Activities. Inbound Logistics - involve relationships with the suppliers and include all the activities required to receive, store, and disseminate inputs. Operations - are all the activities required to transform inputs into outputs (products and services). Outbound Logistics - consist of all the activities required to collect, store, and distribute the output. Marketing and Sales - activities inform buyers about products and services, induce buyers to purchase them, and facilitate their purchase. Service - includes all the activities required to keep the product or service working effectively for the buyer after it is sold and delivered.

Secondary Activities. Procurement - is the acquisition of inputs, or resources, for the firm. Human Resource Management - consists of all the activities involved in recruiting, hiring, training, developing, compensating and (if necessary) dismissing or laying off personnel. Technological Development - pertains to the equipment, hardware, software, procedures and technical knowledge brought to bear in the firm's transformation of inputs into outputs. Infrastructure - serves the company's needs and ties its various parts together. It consists of functions or departments, such as accounting, legal, finance, planning, public affairs, government relations, quality assurance and general management.

Raw Material. The ability to have full transparency into the raw material is critical and can lead to huge impact such as in the case with false advertisement. A case point is that in the week since Indian textile maker Welspun was called out by Target for mislabeling sheets and pillow cases as premium Egyptian cotton products, the company's shares have collapsed. While Target has severed all ties and Walmart and J.C. Penney have announced their own reviews, Bed Bath & Beyond has appointed an independent auditor. A year ago, Welspun, one of the world's largest manufacturers of home textiles, was boasting of a durable competitive advantage over Chinese rivals. The latter, it claimed, had higher labor costs and were dependent on stockpiled Chinese fiber, which tended to lint.

Manufacturing. Communication and coordination are vital in streamlining goals and ensuring timelines are met. Effective coordination will result into successful and smooth order processing. Generally, it has been seen that shipment delay is only the result of poor coordination of merchandisers with another department. Factory buyers largely depend on factory merchandisers. By increasing knowledge sharing, communicating best practices and developing functional coordination platforms, merchandisers can place themselves in a better position to create plans and execute the same as per requirements.

Garments. The dictionary definition of garment is "an article of clothing", however, from a textiles supply chain point of view, garments are the end product of a very long and complicated process. The quality of a single finished garment depends on several quilts of subproducts and materials. This made the second quality and waste in each sector depend on the previous processes [1,11]. Huge efforts and cost are spent on the inspection of the quality of the product in the textile chain, which started with the raw material and is finished by the final garment product. For example, the garment industry inspected all the fabric imported and number of defects which are visually inspected before the purchase. Defect assessment is conducted on between 10%–50% of the produced products where visible flaws such as stains, stitching, incorrect color variations, patterns, sizes, and poor alignment, etc., are tested.

The American Apparel Manufacturers (AAMA) point-grading system for determining fabric quality is the most recommended fabric inspection [13]. The work presented in [14] provides a good example of the garment value chain structure as shown in Fig. 1, demonstrating complicated relations between the different elements of supply chain with each of them having their own quality control system on their product which reacts on the final quality of the final garment [12]. The cost of the quality control in the garment industry varies from 11% up to 39%.

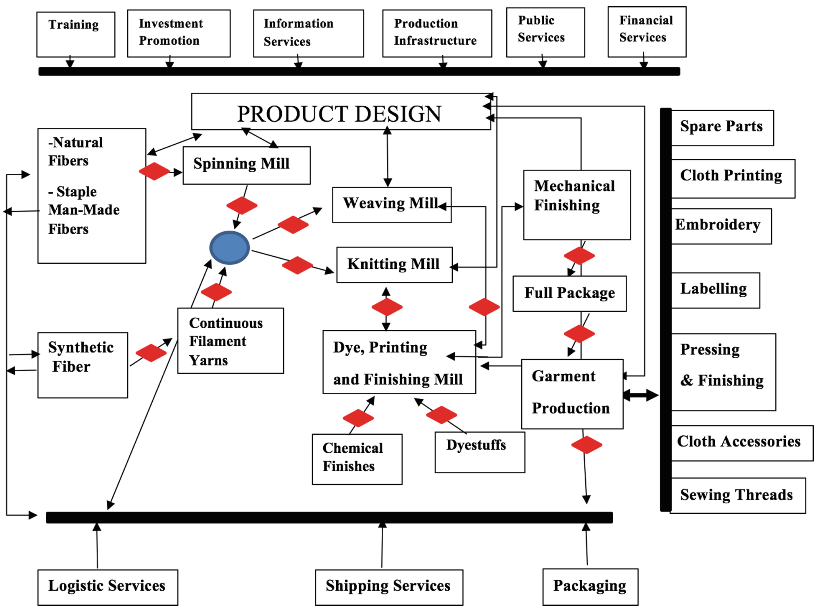


Fig. 1. The garment chain structure.

Fashion. Fighting fake is not the only benefit that blockchain technology can offer the fashion world, it also gives consumers and brands the opportunity to track and display supply chain information [10]. Making this possible is a company called Provenance. They have used blockchain to build a traceability data system that will securely store information that is inherently auditable, unchangeable and open. Their objective is to work towards an open traceability protocol that allows them to tell every product's story using blockchain technology in a way that enables secure traceability of certifications and other information in the supply chains. It answers the question, where does my product come from? As one of the new emerging technologies, blockchain technology is empowering brands to take steps towards greater transparency. Should you ever want to trace the origins, footprint and histories of that cute dress you have your eye on, then blockchain technology will be able to tell you the product's story which leaves us wondering, is fashion ready to be that honest and transparent? They do this by making sure that every physical product comes with a "digital passport" that not only proves the product's authenticity, but also creates an auditable record of the journey the product took. The quality data hub in various points of the quality chain has several types of data about the sub product.

1.3 Blockchain

Blockchain technology is a form of an encrypted distributed ledger, essentially a distributed database of records, or public ledger of all transactions that are shared among participating parties [5,24].

Consensus of a majority of the participants in the system is the main mechanism by which each transaction in the public ledger is verified. Once the transaction is deemed verified, it is then admitted to all the records and can never be erased. The verified transactions are put in a queue to be committed to the next block. The data is secured using a hash function which is any function that can be used to map data of arbitrary size to data of fixed size, more formally defined by Eq. 1, where H is the hash and n is number of bits returned by the hashing function [15].

$$H : K \times M \rightarrow \{0, 1\}^n \tag{1}$$

A block consists of the following main parts:

Payload, which contains the actual data to be committed to the blockchain.
Previous Block Hash, the digital fingerprint of the previous block.
Current Block Hash, the current digital fingerprint of the current block payload and the previous block hash.

The main concept of the blockchain can be illustrated in Fig. 2.

The transactions can be traced back to the original first block, commonly called the genesis block. The genesis block is the only block that does not reference an actual previous block hash. Blockchain contains a certain and verifiable record of every single transaction ever made. The first example of a widely used blockchain application is Bitcoin, the decentralized peer-to-peer digital currency. The central hypothesis is that the blockchain provides a system of a distributed consensus in the digital universe, removing the need for trust and transferring it

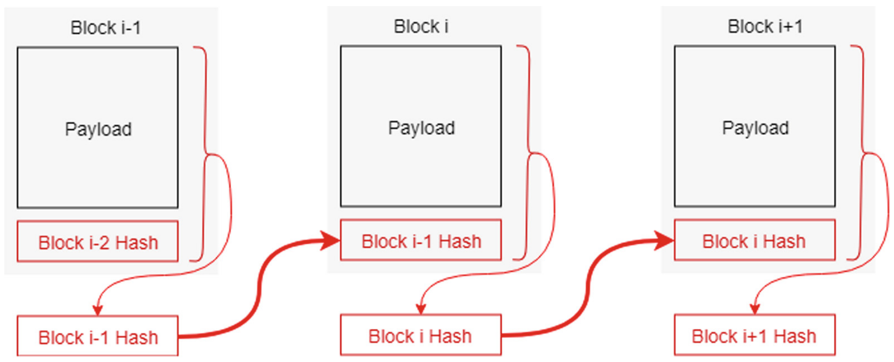


Fig. 2. Conceptual illustration of the blockchain.

to a binding contract, which assures the users that a digital event occurred by creating an irrefutable record in a public ledger [23].

The blockchain can be viewed as a global computing machine with near 100% uptime due to the fact that the contents of the database and ledger are copied across thousands of computers. Thus in case of 99% of the computers running it were taken offline, the records would remain accessible and the network could rebuild itself. The distributed nature of the blockchain also means that a local copy can exist at or near the user. This is a very important practical consideration as many of the textile facilities are located in developing countries with very limited bandwidth. Having a local copy that auto updates reduces potential failure due to Internet bandwidth.

The central feature of the blockchain technology is an immutable ledger [19]. Immutable means that the contents of the payload of each block cannot be changed after it is committed to the chain. This is due to the fact that each block hash is computed based on the payload of the block and the hash of the previous block as shown in Fig. 2. If we want to tamper with block i, we will need to recompute the hash of block i. That will require us to recompute the hash of every and all subsequent blocks as changing one hash will invalidate all subsequent hashes. Now, because the ledger is distributed, we will need to gain control and change the hashes of at least 51% of the entire network. The sheer amount of required effort renders it practically impossible. One model of understanding blockchain is through comparing it to the new application layer for Internet protocols because blockchain can enable both immediate and long-term economic transactions, and more complicated financial contracts. It can be a layer for transactions of different types of assets, currency or financial contracts. Moreover, a registry and inventory system for recording, tracking, monitoring, and transacting of all assets could be managed with blockchain. Consequently, blockchain can be used for any form of asset, including every area of finance, economics, and money [16].

1.4 Blockchain in Textiles

There are many benefits of blockchain. Yet, there is little to no adoption of blockchain technology in the textiles supply chain domain. Blockchain technology presents many features and characteristics that can be useful in textile industry aspects such as: compliance, transparency, tracking, tracing, error reduction, payment processing, and many others [20]. IBM has revealed its intention to lead an "industry-wide collaboration" to create a supply chain and trading ecosystem built on IBM blockchain technology. It will use the Hyperledger Fabric, which provides a foundation for developing blockchain solutions with a modular architecture, pluggable implementations and so-called container technology. There are numerous organizations, processes, systems, and transactions involved from field to fabric. Located at the intersection of agriculture, finance and technology, the Seam with the help of IBM, is in a unique position to introduce blockchain technology to cotton-affiliated companies around the world. In conjunction with IBM, the Seam wishes to create a supply chain and trading ecosystem built on IBM blockchain using the hyperledger fabric. This new technology will be transformational for the cotton industry. There are numerous organizations, processes, systems and transactions involved from field to fabric. The Seam and IBM launched the first cotton industry blockchain consortium.

2 Problem Statement

The problem we address in this work is as wide as it is deep. It deals with an industry that has been in existence for as long as humans started coexisting in large groups. In the following sections, we will go over the main aspects of the problem in the textiles supply chain.

Transparency. The textiles and clothing sectors are a supply chain consisting of many discrete activities as shown in Fig. 3. It is being organized as an integrated production network where the production is sliced into specialized activities and each activity is located where it can contribute the most to the value of the product.

When the location decision of each activity is being made, costs, quality, reliability of delivery, access to quality inputs and transport and transaction costs are important variables. Walmart insisted that suppliers implement information technologies for exchange of sales data and adopt standards for product labeling and methods of material handling. This ensures quick replenishment of apparel, which in turn allows the retailer to offer a broad variety of fashion clothes without holding a large inventory. This approach has spread throughout the industry in the United States as well as elsewhere (and to other industries), shifting the competitive advantage of suppliers from being mainly a question of production costs to becoming a question of costs in combination with lead time and flexibility and quality.

The number of the suppliers in any industrial product increased almost exponentially throughout the chain, which means that the reality of an organization being able to control all aspects and impacts of its supply chain is a goal rather than a certainty. Figure 3 shows that the reality of tracking and influencing suppliers was clearly articulated. This required high transparency throughout the supply chain. Consequently, the success of the enterprise depends on the transparency and the trust between the suppliers within the enterprise.

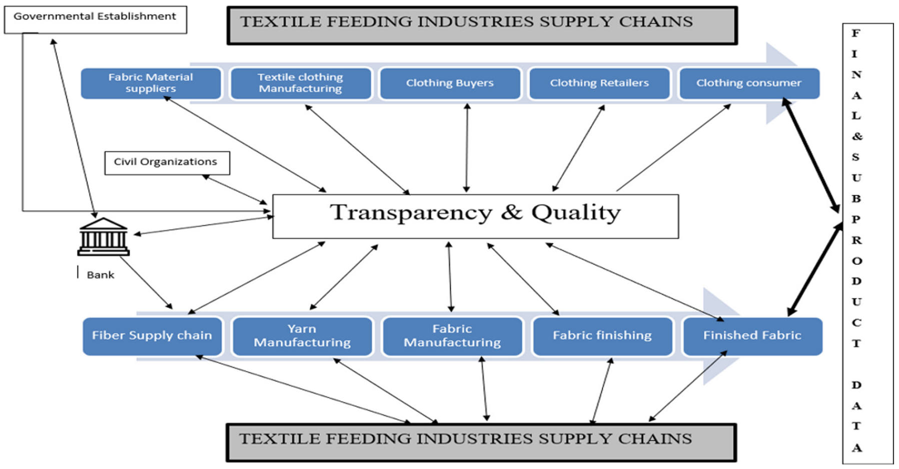


Fig. 3. Conceptual illustration of the textile supply chain.

The results of these long chains, the time required for quick response of the market demands and the cost increased tremendously with instability of the product quality.

We know surprisingly little about most of the products we use every day. Even before reaching the end consumer, goods travel through an often-vast network of retailers, distributors, transporters, storage facilities, and suppliers that participate in design, production, delivery, and sales, yet in almost every case, these journeys remain an unseen dimension of our possessions. The supply chain sector represents billions of dollars in enterprise revenue but is fraught with losses and inefficiencies resulting from risk, fraud or anachronistic manual paperwork delays. The main challenge is setting up technology for farmers, field workers and others to collect data and insert it onto a blockchain. Innovative data entry tools running on ubiquitous smartphones, with backends in the cloud, are expected to allow field workers to input relevant data to a blockchain ledger that tracks all data, making it accessible "in minutes, rather than days", thereby improving supply chain efficiency, identifying bottlenecks and reducing waste.

The bales of cotton arriving at the port and being scanned automatically trigger the smart contract to execute the terms, which would involve transferring the ownership of goods and authorizing payment. This happens because there is a single document agreed on by all parties and that is only completed once a certain action has taken place.

Traceability. Traceability has recently gained considerable attention in the textile industry. Traceability stands for information sharing about a product including the history, specification, and location. With the involvement of globally dispersed actors in the textile supply chain, ensuring appropriate product quality with timely supplies is crucial for surviving in this industry with ever increasing competition.

Quality. Quality is one of the main cost factors in the textile industry. As manufacturers face an increasingly competitive global business environment, they seek opportunities to reduce production costs without negatively affecting product yield or quality. Rising quality requirements are driving up costs and decreasing value added at the plant. The supply chain of the textile product is completed with involvement of worldwide distributer supplier chains as shown in Fig. 6 textile supply chain. The high percentage of quality control cost is related to lack of data about the subproduct and material and processing defects either in the previous operations or in processes defects. All the applied quality control management systems suffer from: late detection of defects, higher cost of defective work, poor access to previous production supervision, second quality being higher, more manpower to maintain quality, and higher manufacturing costs.

One of the early works that related information impacted on the product quality is outlined in [7], where the authors show that in the textile field, the quality of the final product is normally affected by different material and production parameters. They propose a matrix representation of each production process input and required information as well as the importance of the information for the production process. In the case of garments, it is established that the garment quality is determined by different fabric characteristics such as fabric defects, yarn hairiness, color shades variations, sewing defect, etc. Consider n quality characteristics of a certain process, defined as $Y_i(i = 1\, to\, n)$, each Y_i value may be affected by m data value of the previous process parameters defined as $X_j, (j = 1\, to\, m)$. The introduction of the entire set of values of the previous process parameters to predict the entire set of garment characteristics is very complex. Therefore, the identification of the main data affecting the investigated set of Y_i value characteristics would be very useful. To perform this task, a matrix $A(mxn)$ is formed with the variables X_j which are expected to affect properties Y_i. Each coefficient a_i, j will be equal either to 1 when the corresponding parameter X_j is expected to have an effect on the corresponding garment characteristic Y_i or to zero when no effect is to be expected. Matrix $B(n * 1)$ defines the set of yarn characteristics to be predicted. The multiplication of the two matrices (AxB) gives a new matrix C. For the values $c_j > 0$ the corresponding data X_j should be taken as an input data required for this particular process.

The larger the positive value of c_j is, the more significant the effect of the corresponding value X_j is expected to be. The matrices are used to predict the impact on quality of the different required data that defines the quality. In more complex systems, the impact of each information would be adjusted using a learned model [2] (Fig. 4).

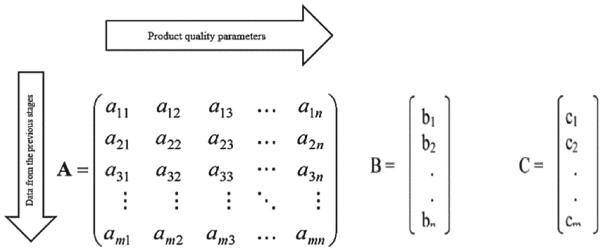

Product quality parameters
Data from the previous stages
$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{pmatrix}$
$B = \begin{bmatrix} b_1 \\ b_2 \\ . \\ . \\ b_n \end{bmatrix}$
$C = \begin{bmatrix} c_1 \\ c_2 \\ . \\ . \\ c_m \end{bmatrix}$

Fig. 4. Matrices used for the presentation and selection of input parameters from one stage to the next.

3 Proposed Framework

Our proposed framework works by adding textile materials information to the blockchain at each point along the supply chain journey. Each atomic unit of raw material will be uniquely identified, as illustrated in Fig. 5. The process starts at the farm with identifying the seeds used. The resulting crop bales will be given a unique number as the first component in the manufacturing process. As raw materials are used in each manufacturing process, the resultant product will be recorded as a transaction on the blockchain. The transactions can identify the processes and machinery used and be traced to the previous manufacturing stage. Each machine or manufacturing unit can be connected directly to the blockchain and is capable of reading previously recorded data pertaining to the material which the manufacturer has acquired. The machinery can update new information regarding only to the material that has been acquired by the manufacturer. Before the machine starts processing the material, the machine would check the reported history by confirmed owners of the same material batch. After processing the material, the machine would automatically post the results to the blockchain. A consensus mechanism is used to evaluate when to flag a batch with a defect.

The framework allows the machine to utilize the information shared by others to automatically stop processing the material and require human override. The proposed framework work flow is illustrated in Fig. 6.

3.1 Machine Chain Interface

One critical change required to enable the framework is to upgrade the current machinery so that it can interface with the blockchain. Most modern textile

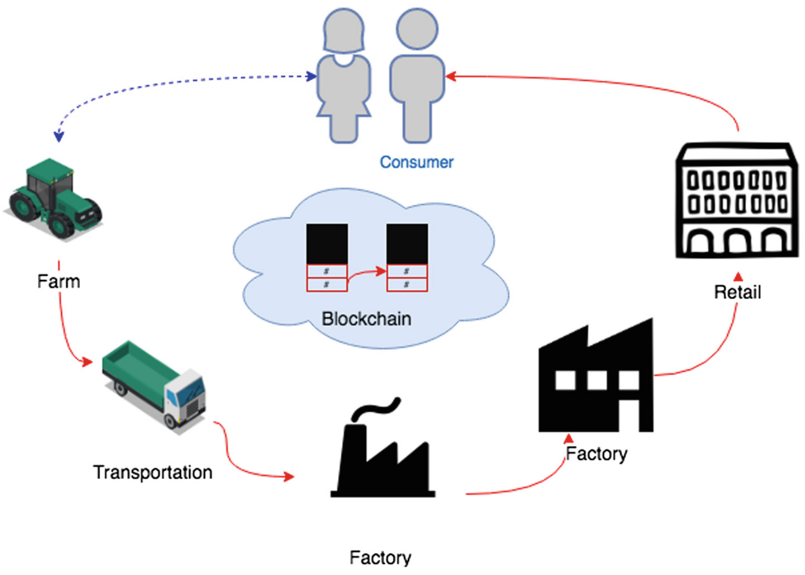


Fig. 5. Matrices used for the presentation and selection of input parameters from one stage to the next.

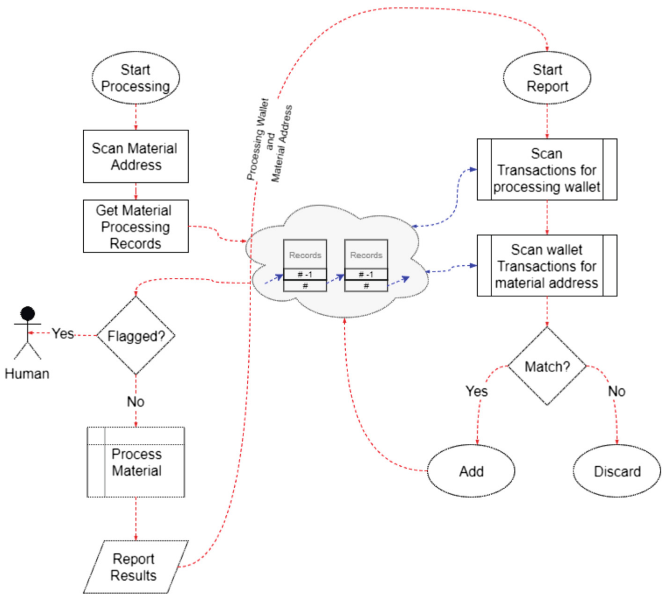

Start Processing
Scan Material Address
Get Material Processing Records
Processing Wallet and Material Address
Records
Records
Flagged?
Yes
Human
No
Process Material
Report Results
Start Report
Scan Transactions for processing wallet
Scan wallet Transactions for material address
Match?
Yes
No
Add
Discard

Fig. 6. The proposed framework work flow.

machines are equipped with sensors that can communicate measurement back and allow remote control. The fundamental difference in our model is that each machine will be identified by a wallet in addition to the existing identifier. Using a wallet will enable the chain to track incoming material and the processed output. In addition, due to the ledger architecture, the material will have to be traced back to the initial creation wallet.

3.2 Material Creation

The first step in the framework is to create the material. Only certain actors such as farms and fiber manufacturers would be able to create new materials and record them on the chain. Once the material is created, all the properties are committed to the chain. This way, the material is traceable back to this genesis wallet.

3.3 Manufacturing

The process of manufacturing is mapped to the chain by adding a manufacturer role. A manufacturer can record the purchase of a certain batch of material in conjunction with the material creator. This is represented by a transaction of the material portion. The transaction can be of the entire or partial batch. Once the material is transferred on the chain from the creator to the manufacturer, the manufacturer is now able to record the material processing results. This is a fundamental step in the framework to prevent bad actors from destroying the credibility of competitors. The network will only accept reports from those that have been recorded on the chain to own the material.

3.4 Reporting and Consensus

Any network participant has full access to the entire network. The information is publicly accessible. However, only material owners can report on the processing results of the material. A consensus mechanism requiring certain criteria to be fulfilled before the material is flagged on the chain. The consensus is a critical component to remove errors and outliers from the chain. An example of consensus criteria can be:

- Same material batch
- Similar manufacturing process
- Minimum number of reports
- Defined time in which the reports are recorded.

3.5 Automated Protection

Usually, the action to correct the fault in the textile chain takes place after it occurs. Through the introduction of the concept of automated protection, if the equipment queries the chain to find that the batch about to process is flagged as a bad batch for this process, the equipment will halt the process and notify the attendant to make a decision based on the provided chain data.

4 Evaluation

It is clear that from a traceability and transparency point of view, our proposed workframe provides capabilities that do not exist in the current status quo. To evaluate our proposed workframe from the quality point of view, we need to take into consideration the potential cost waste due to the degradation in product quality as a function of information quality.

We build on the approach proposed in [7] by extending it to different stages in the supply chain as illustrated in Fig. 7. The impact at each stage is computed and then combined with the following stage as shown in Eq. 2.

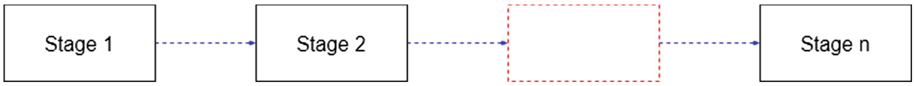


Fig. 7. Abstract representation of textiles supply chain stages.

$$\text{Quality Impact} = 1 - \prod_{i=1}^{i=n} \left(\frac{Si_i}{Ai_i} \right) \tag{2}$$

where i is the current stage, n the total number of supply chain stages, Si_i the shared information at stage i, while Ai_i is the available information at stage i. Figure 8.

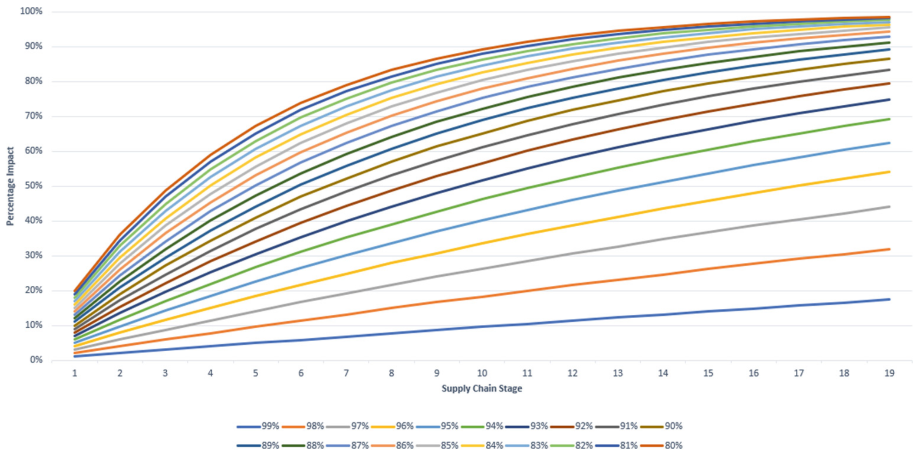


Fig. 8. Potential impact of supply chain stage information on quality.

At every stage, the missing information affects not only the current stage quality, but also the subsequent stages down stream. Our simulation shows that missing information can greatly affect the overall quality of the entire process. Just missing consistently on 20% throughout the supply chain can lead to more than 98% quality degradation by the 10^{th} stage. In contrast, we mapped the information flow in a local garment production line, then assured that all required fabric information is provided in every stage. We observed that second quality products dropped by 26.5%.

5 Conclusion

Most agroindustry corporations believe new technologies of blockchain will be transformational for the cotton industry by making it easier and more secure to trade the commodity. In this work, we proposed a novel approach to solve a huge problem in the textiles supply chain. We presented how this approach will interact with the current status quo and how it will lead to massive improvement in the traceability, transparency and quality of the textile production and end product. Our work combines one of the oldest technologies known to man with the latest cutting edge one. In our further research, we will explore exact parameters for this approach to be adopted as well as the impact on the end user, consumer, and how it can change the social sensibilities of the consumer.

References

1. Bheda, R.: Do you know the cost of poor quality in your organization? (2010). http://rajeshbheda.com/pdf/rbc-article.pdf
2. Burke, H., Rosen, D., Goodman, P.: Comparing artificial neural networks to other statistical methods for medical outcome prediction. In: Proceedings of the IEEE International Conference on Neural Networks, Orlando, vol. 4, pp. 2213–2216, June 1994
3. Çelik, H.I., Topalbekitoglu, M., Dülger, L.: Real-time denim fabric inspection using image analysis. Fibres Text. Eastern Eur. (2015)
4. Çelik, H., Dülger, L., Topalbekiroğlu, M.: Development of a machine vision system: real-time fabric defect detection and classification with neural networks. J. Text. Inst. **105**(6), 575–585 (2014)
5. Crosby, M., Pattanayak, P., Verma, S., Kalyanaraman, V.: Blockchain technology: beyond bitcoin. Appl. Innov. **2**, 6–10 (2016)
6. Das, P., Roy, S., Antony, J.: An application of six sigma methodology to reduce lot-to-lot shade variation of linen fabrics. J. Ind. Text. **36**(3), 227–251 (2007)
7. El Messiry, M., Abd-Ellatif, S.A.M.: Artificial Neural Networks For Solving Textile Problems
8. Gandino, F., Montrucchio, B., Rebaudengo, M., Sanchez, E.: Analysis of an RFID-based information system for tracking and tracing in an agri-food chain. In: RFID Eurasia, 2007 1st annual, pp. 1–6. IEEE (2007)
9. Gunay, M.: Dynamic Textile Process and Quality Control Systems
10. Kapfunde, M.: Is blockchain technology set to revolutionize the fashion industry? (2016). http://fashnerd.com/2016/10/is-blockchain-technology-set-to-revolutionize-the-fashion-industry/

11. Mogahzy, Y.E.E.: Using off-line quality engineering in textile processing: Part I: concepts and theories. Text. Res. J. **62**(5), 266–274 (1992)
12. Purnomo, M.R.A., Dewi, I.H.S.: A manufacturing quality assessment model based-on two stages interval type-2 fuzzy logic. In: IOP Conference Series: Materials Science and Engineering, vol. 105, p. 012044. IOP Publishing (2016)
13. Rana, N.: Fabric inspection systems for apparel industry. Indian Text. J. (2012)
14. Richero, R., Ferrigno, S.: A background analysis on transparency and traceability in the garment value chain. In: The European International Cooperation And Development Commission
15. Rogaway, P., Shrimpton, T.: Cryptographic hash-function basics: definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 371–388. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-25937-4_24
16. Sadouskaya, K., et al.: Adoption of Blockchain Technology in Supply Chain and Logistics (2017)
17. Sharif, E.E.: Study on quality management system and technical solution of defects in lingerie unit. Daffodil International University, Technical report (2014)
18. Suh, M.W., Gunay, M., Vangala, R.: Dynamic Textile Process and Quality Control Systems. NC State University, NTC Annual Report (2007)
19. Tapscott, D., Tapscott, A.: Blockchain Revolution: How the Technology Behind Bitcoin is Changing Money, Business, and the World. Penguin (2016)
20. Tapscott, D., Tapscott, A.: How blockchain will change organizations. MIT Sloan Manage. Rev. **58**(2), 10 (2017)
21. Vangala, R.N.: Design of a Dynamic Quality Control System for Textile Processes (2008)
22. Vangala, R.N.: Dynamic Process/Quality Control System using Structural EquationsApplications in Ring Spinning and Continuous Dyeing. North Carolina State University (2013)
23. Wang, J., Li, L., He, Q., Yu, X., Liu, Z.: Research on the application of block chain in supply chain finance. In: DEStech Transactions on Computer Science and Engineering (ICEITI) (2017)
24. Wright, A., De Filippi, P.: Decentralized Blockchain Technology and the Rise of Lex Cryptographia (2015)

Research on the Pricing Strategy of the CryptoCurrency Miner's Market

Liping Deng[1,2](✉), Jin Che[1,2], Huan Chen[1,2], and Liang-Jie Zhang[1,2]

[1] National Engineering Research Center for Supporting Software of Enterprise Internet Services, Shenzhen, China
[2] Kingdee Research, Kingdee International Software Group Company Limited, Shenzhen, China
liping_deng@kingdee.com

Abstract. Although the attitudes of governments and the general public on virtual currencies vary greatly, the prices of virtual currencies have grown at an extremely exaggerated rate since 2016, attracting more and more investors and media attention. Not only that, while virtual currency prices have increased, mining has turned into a profitable business, and many people have purchased miners to invest in the mining industry. This article examines the stability of the two time series of bitcoin's price and miner's hashrate from 2016 to the present. Research shows that price changes are the Granger cause of changes in hashrate. By establishing a distributed lag model, the quantitative relationship between hashrate and price is analyzed. Combined with the follow-up investigation of the miner's market, it discovered the pricing strategy of the miner's market, that is, the current miner's price is determined by the price of the previous cryptocurrency, and the lag period is calculated.

Keywords: Cryptocurrency · Miner · Pricing strategy · Granger causal Distributed lag model

1 Introduction

With the development and extensive application of blockchain technology, cryptocurrencies represented by bitcoin are also gradually known. Figure 1 is a trend graph of the frequency with which bitcoin was retrieved as a keyword in Google [1]. Bitcoin's attention has been increasing since 2016. Although people and government have different attitudes toward cryptocurrency, and different kinds of virtual currency prices fluctuate greatly, some investors' enthusiasm for investing in cryptocurrency remains unabated.

Different from the traditional currency issuance strategy, the addition of virtual currency requires the miners to contribute computing resources to compete for additional currency per unit of time. This process is called mining, and professional mining tools are called miners. Mining maintains the issuance and circulation of the entire virtual currency. As the virtual currency prices have risen, mining has turned from the small profits of the past to a business that can be turned over, and more and more

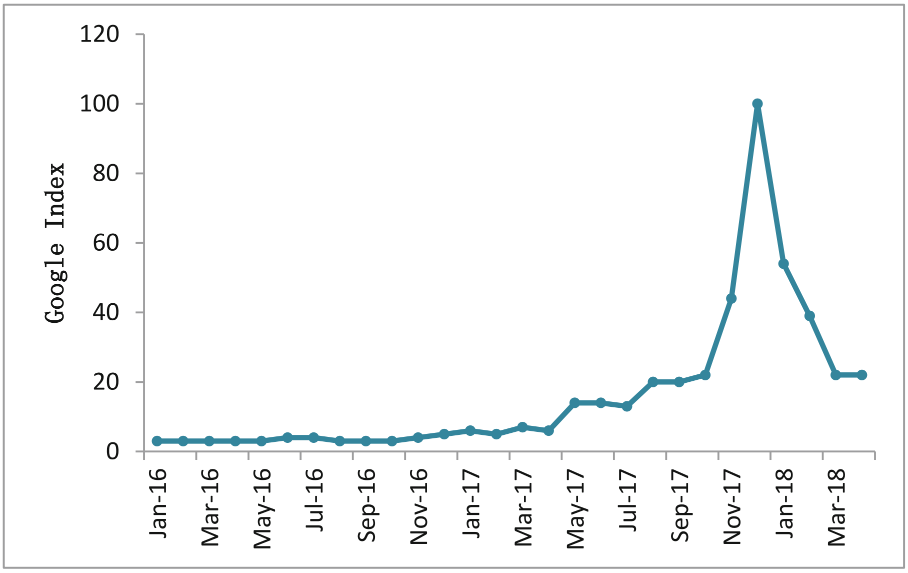


Fig. 1. Google trends of Bitcoin

people are starting to purchase large quantities of miners for professional mining. Naturally, the miner's price and even the price of the video card have also skyrocketed.

Mining [2] is the process of a node that participating in maintaining the cryptocurrency network obtains a certain amount of new cryptocurrencies through assisting in generating new blocks. Mining is also a very hot profitable industry. From ordinary CPUs, GPUs, and FPGAs, to ASIC miners, and to mining pools formed by numerous miners. In just a few years, the technology of miners has gone through the evolution of integrated circuit technology over the past decades. At present, mining is mainly concentrated in China. Driven by huge profits, the Chinese miner's market has developed rapidly and is favored by investors. This paper, through the follow-up investigation of the miner's market, the pricing strategy of the miner's market was discovered. That is, the current miner's price is determined by the price of the previous cryptocurrency. The results of this study have important guiding significance for investors to purchase miners and avoid risks.

2 Research Background

On October 31, 2008, Satoshi Nakamoto first proposed the concept of bitcoin in Bitcoin: *A Peer-to-Peer Electronic Cash System* [3]. Bitcoin is a cryptocurrency and worldwide payment system. It is the first decentralized digital currency, as the system works without a central bank or single administrator. An alternative, the numerous among of other cryptocurrencies have been created. In cryptocurrency, miner is a computer or group of computers searching for cryptocurrency. They constantly verify transactions and as an incentive, they get rewarded with cryptocurrency.

2.1 Bitcoin

Researchers have conducted in-depth research on whether or not bitcoin has monetary functions and obtained many valuable results. In 2015, Kristoufek [4] used the continuous wavelet framework to study the evolution of bitcoin in both time and frequency domains, showing that bitcoin formed a unique asset with standard financial assets and speculative assets. In 2016, Bouri [5] used the dynamic conditional correlation model to analyze the relationship between bitcoin and major world equity indices, bonds, oil, gold, general commodity indices and US dollars. From the perspective of research assets, bitcoin can be seen as a digital currency. In 2017, Indera [6] proposed a Multi-Layer Perceptron based Non-Linear Autoregressive model. Use bitcoin's historical price trend to predict the future price of bitcoin and achieve better results.

2.2 Cryptocurrency

A cryptocurrency is a digital asset designed to work as a medium of commutation that uses cryptanalytics to secure its transactions, to control the introduction of additional unit of measurement, and to verify the transfer of assets. Cryptocurrency is based on cryptography and network P2P technology, generated by computer programs, and distributed and circulated on the Internet. The cryptocurrency [7] has received increasing attention from the media, the public, scholars and the government. In addition to research from computer science, cryptography, etc., more and more scholars have begun to pay attention to the economic analysis of cryptocurrency, its currency or asset characteristic, and the innovation of cryptocurrency to traditional currency theory and payment methods.

In 2013, Ahamad [8] criticized the legal currency issued by the government, gave a detailed introduction to various virtual currencies, and proposed a feasible scheme to replace the legal currency with virtual currency. Recently, the perspective of the central bank, the design of cryptocurrency should not only consider protecting people's privacy, but also need to pay attention to social security and social order. In short, cryptocurrency is still in its early stages of development, and many new ideas have emerged. However, there are obvious argument that guarantee deep research.

2.3 Miner

Mining is the process of confirming the transactions that occur in the bitcoin system over a period of time and recording the formation of new blocks on the blockchain. Bitcoin may be obtained through mining, with objective benefits. In 2016, Bouiyour [9] made a quantitative analysis of the sharp declination in price of bitcoin from 2015 to mid-2016 using Asymmetric-power GARCH. Researchers showed that the bitcoin market was very immature and had an impact on price fluctuations in the miner's market. In 2015, Hayes [10] analyzed 66 virtual currencies and conducted regression analysis from the three dimensions: the difficulty in mining for coins, the rate of unit production and the cryptographic algorithm. Researches showed that bitcoin production was similar to the highly competitive commodity market. Theoretically, miners will always produce until their marginal cost is equal to their marginal turnout.

Anything that helps to reducing bitcoin's production costs will have a negative impact on its price, such as increasing the energy efficiency of mining hardware or lowering global electricity prices. At the same time, the increased computing power of the global mining network will increase the difficulty of mining and will have a positive impact on prices.

Hayes's researchers showed that bitcoin's price was related to miner's hashrate. On this basis, this paper used Granger causality relation test and distributed lag model, quantitatively expounded the impact of bitcoin price changes on the changes of hashrate.

3 The Relationship Between Hashrate and Price

This section examines the relationship between hash rate and price. The information of bitcoin from 2016.01.01 to 2018.03.31 is obtained from bitinfocharts.com, and the average price and hash rate are calculated monthly. Figure 2 shows that prices and hash rates have risen in synchrony. So, is there a stable relationship between them? Is there a causal relationship with "who causes changes"? The following applies the principles of econometrics to answer this question.

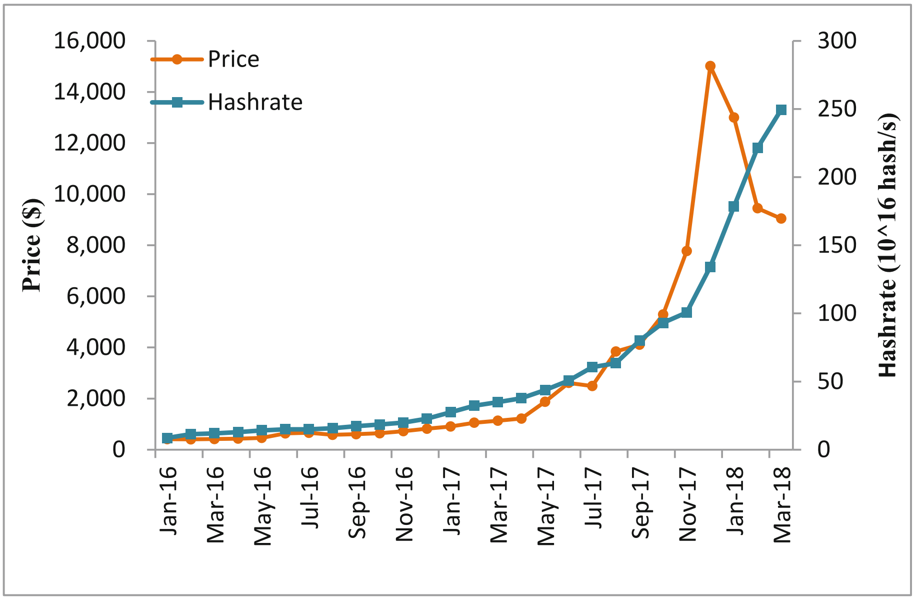
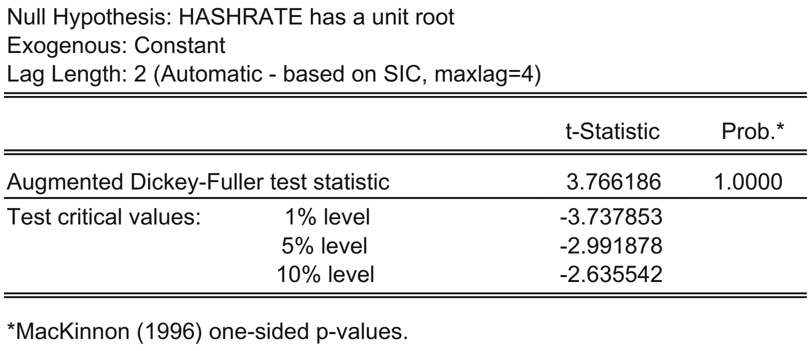


Fig. 2. The trend of Bitcoin's price and hashrate

3.1 Unit Root Test

In econometric theory, finding the relationship between variables requires a stability test. If it is a stationary time series given that a classical model such as a regression model can be constructed; if it is not a stationary time series, a stationary series is constructed by difference, and then a cointegration relationship between variables is established. Unit root test [11] is often used to verify the smoothness of the variables.

Using Eviews 9.0 [12], the unit root test results for the hash rate series is shown in Fig. 3. Prob = 1.0000 > 0.05 accepts the null hypothesis. That is, the hash rate series has a given unit root and is an unstable series.



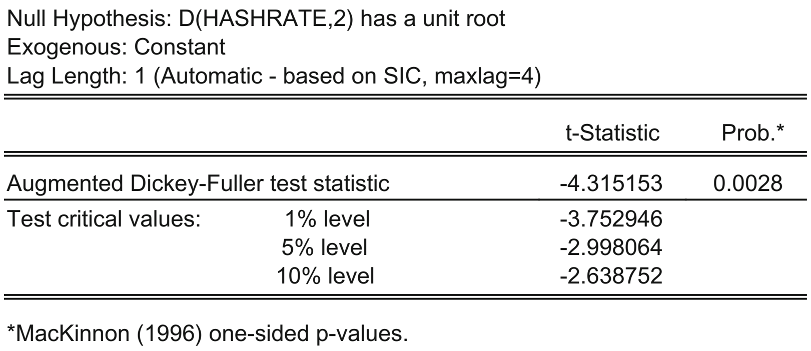
Null Hypothesis: HASHRATE has a unit root
Exogenous: Constant
Lag Length: 2 (Automatic - based on SIC, maxlag=4)

		t-Statistic	Prob.*
Augmented Dickey-Fuller test statistic		3.766186	1.0000
Test critical values:	1% level	-3.737853	
	5% level	-2.991878	
	10% level	-2.635542	

*MacKinnon (1996) one-sided p-values.

Fig. 3. The results of unit root test for hashrate

If a time series is smoothed by a difference, the original series is said to be integrated of order 1 and denoted as I(1). The differential operation and unit root verification are performed on the sequence of hashrate. Studies have shown that the hashrate series is not integrated of order 1 but integrated of order 2. Prob = 0.0028 < 0.05, rejecting the null hypothesis that the second-order difference of the hashrate series is a stationary sequence (Fig. 4).



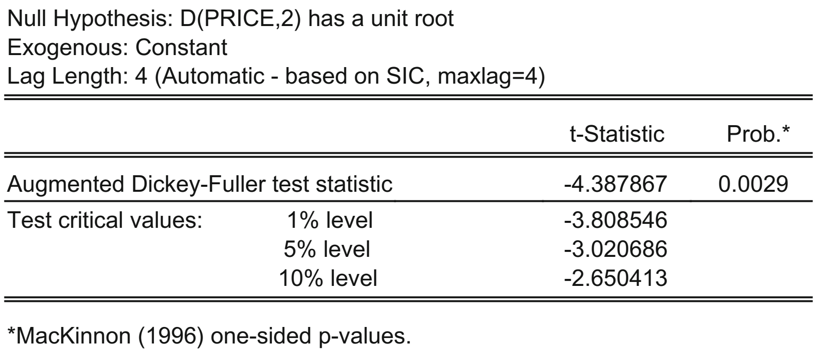
Null Hypothesis: D(HASHRATE,2) has a unit root
Exogenous: Constant
Lag Length: 1 (Automatic - based on SIC, maxlag=4)

		t-Statistic	Prob.*
Augmented Dickey-Fuller test statistic		-4.315153	0.0028
Test critical values:	1% level	-3.752946	
	5% level	-2.998064	
	10% level	-2.638752	

*MacKinnon (1996) one-sided p-values.

Fig. 4. The results of unit root test for hashrate I(2)

A unit root test of the price series makes it easy to know that the price series is not a sequence of I(0) and I(1). In Fig. 5, with Prob = 0.0029 < 0.05, the price series is integrated of order 2.



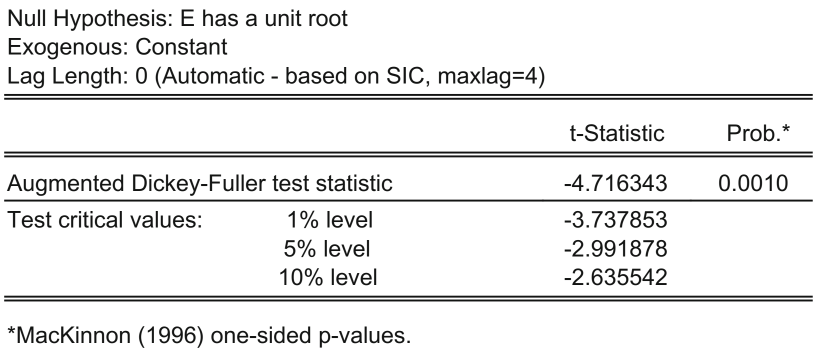
```
Null Hypothesis: D(PRICE,2) has a unit root
Exogenous: Constant
Lag Length: 4 (Automatic - based on SIC, maxlag=4)

                                             t-Statistic    Prob.*

Augmented Dickey-Fuller test statistic        -4.387867     0.0029
Test critical values:        1% level         -3.808546
                             5% level         -3.020686
                             10% level        -2.650413

*MacKinnon (1996) one-sided p-values.
```

Fig. 5. The results of unit root test for price

3.2 Cointegration Test

The cointegration test is a causality test for non-stationary sequences. If the linear combination of non-stationary sequences is stationary, this combination reflects the long-term stable proportional relationship between variables.

Both the hash rate series and the price series are integrated of order 2 and denoted as I(2), and the two-variable Engle-Granger test is used to study the assistance relationships between variables. The OLS (Ordinary Least Square) regression was performed on the second-order differential of the hash rate and price to obtain the residual sequence denoted as e. The ADF (Augmented Dickey-Full) test result is shown in Fig. 6.



```
Null Hypothesis: E has a unit root
Exogenous: Constant
Lag Length: 0 (Automatic - based on SIC, maxlag=4)

                                             t-Statistic    Prob.*

Augmented Dickey-Fuller test statistic        -4.716343     0.0010
Test critical values:        1% level         -3.737853
                             5% level         -2.991878
                             10% level        -2.635542

*MacKinnon (1996) one-sided p-values.
```

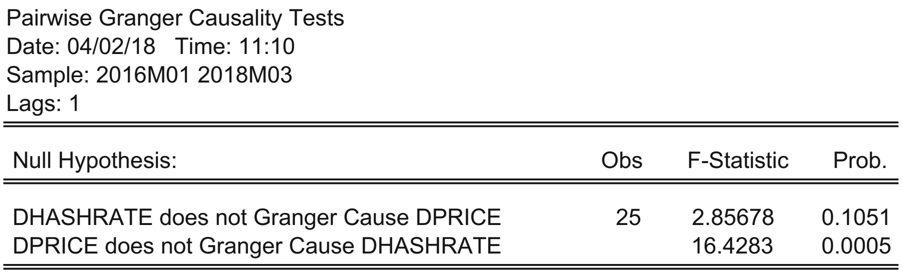
Fig. 6. The results of ADF test for residual sequence

The t-statistic of the residual sequence is −4.716343, Prob = 0.0010 < 0.05, and the residual sequence of the regression model is a stationary sequence, indicating that there is a cointegration relationship between the hash rate and price.

3.3 Granger Causal Relation Test

Clive W. J. Granger, winner of the 2003 Nobel Prize in Economics, proposed Granger's causality relation test. This method is suitable for econometric variables prediction. Granger causality relation refers to the previous changes in variable X that helps to explain the future changes of variable Y.

Cointegration results indicate that there is a long-term equilibrium relationship among variables. In the end, it is "who causes changes"? Granular causality validation can be used for integrated of the same order.



```
Pairwise Granger Causality Tests
Date: 04/02/18   Time: 11:10
Sample: 2016M01 2018M03
Lags: 1

Null Hypothesis:                               Obs    F-Statistic    Prob.

DHASHRATE does not Granger Cause DPRICE         25      2.85678     0.1051
DPRICE does not Granger Cause DHASHRATE                 16.4283     0.0005
```

Fig. 7. The results of Granger Causality Tests

In Fig. 7, Prob = 0.105 > 0.05, receiving the null hypothesis. In this Case, the change in hash rate does not Granger cause for price change. Prob = 0.0005 < 0.05, rejecting the null hypothesis. It shows that the change of the price in the previous period helps to explain the change of hashrate in the future. That is to say, the change in price is the Granger cause of change in hashrate.

In the next section, we use the price as an independent variable and the hashrate as a dependent variable to establish a distributed lag model to reveal the Granger causality between the hashrate and the price.

4 Distributed Lag Model

4.1 Model Introduction

In real life, there is widespread time lag effect. A certain variable is not only affected by various factors in current period, but also affected by various factors in the past period or even its own past value. We call this kind of hysteretic variable in the past period a lagged variable. Models with lagged variables are called lagged variable models. The general form of the model [13] is:

$$\mathrm{Y}_t = \alpha + \beta_0 X_t + \beta_1 X_{t-1} + \cdots + \beta_p X_{t-p} + u_t \quad (t = p+1, p+2, \cdots, n) \tag{4.1}$$

Equation (4.1) is called the distributed lag model, which shows the influence of the explanatory variable X over time on the explained variable Y. Where α is a constant term and u_t is the random interference. The coefficient β_i reflects the different degree of influence of the current value of the explanatory variable and the lag value of each period on the explained variable, also known as the multiplier.

p is the lag time interval and is called the lag period. If the length of the lag period is limited, the model is called a finite distributed lag model; if the lag period is infinite, the model is called an infinite distributed lag model.

4.2 Koyck Method

For the infinite distributed lag model [13],

$$\mathrm{Y}_t = \alpha + \sum\nolimits_{i=0}^{\infty} \beta_i X_{t-i} + u_t \tag{4.2}$$

The degree of influence of the lag explanatory variable on the explained variable is gradually weakened with the time interval. The Koyck method assumes that the decay is in decreasing geometric order, that is, the coefficient of the lag explanatory variable β_i satisfies (4.3):

$$\beta_i = \beta\gamma^i,\ 0 < \gamma < 1 \quad (i = 0, 1, 2, \cdots) \tag{4.3}$$

γ is called distributed lag decay rate and $1 - \gamma$ is called speed of adjustment. Substituting (4.3) into (4.2) gives:

$$\mathrm{Y}_t = \alpha + \beta\left(X_t + \gamma X_{t-1} + \cdots + \gamma^p X_{t-1-p} + \cdots\right) + u_t \tag{4.4}$$

Equation (4.4) is called the Koyck distributed lag model.

$$\mathrm{Y}_{t-1} = \alpha + \beta\left(X_{t-1} + \gamma X_{t-2} + \cdots + \gamma^p X_{t-1-p} + \cdots\right) + u_{t-1} \tag{4.5}$$

(4.4) minus the result of γ multiply by (4.5):

$$\mathrm{Y}_t - \gamma \mathrm{Y}_{t-1} = \alpha(1-\gamma) + \beta X_t + (u_t - \gamma u_{t-1}) \tag{4.6}$$

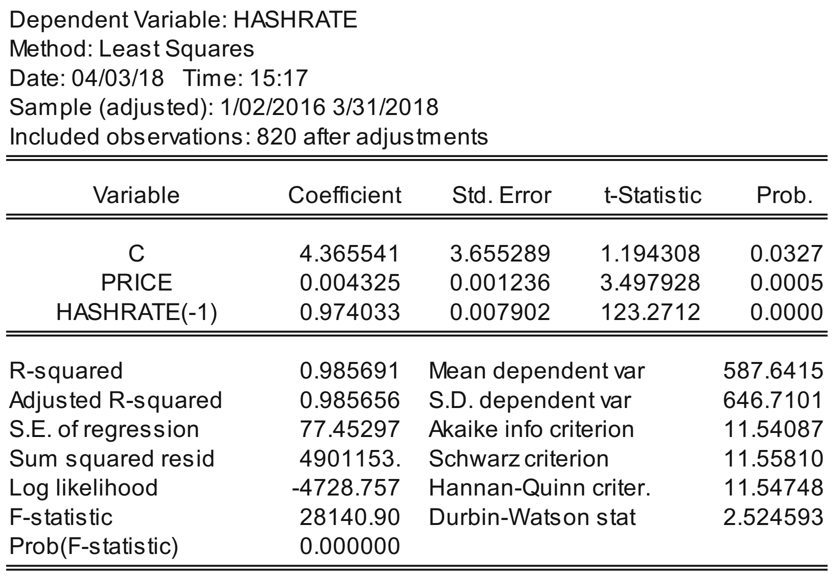
After finishing:

$$\mathrm{Y}_t = \alpha(1-\gamma) + \beta X_t + \gamma \mathrm{Y}_{t-1} + v_t, \quad v_t = (u_t - \gamma u_{t-1}) \tag{4.7}$$

From Eq. (4.7), an infinitely distributed lag model can be transformed into an autoregressive model using the Koyck transform. The larger the γ value, the more prolonged effect of lagged variables.

4.3 Model Solving

With 9.0, the Eq. (4.7) is solved by using the OLS method. The time series is constructed from the daily price and hash rate of bitcoin from 2016.01.01 to 2018.03.31. The price unit is \$ and the unit of hash rate is 10^{16} hash/s.



```
Dependent Variable: HASHRATE
Method: Least Squares
Date: 04/03/18   Time: 15:17
Sample (adjusted): 1/02/2016 3/31/2018
Included observations: 820 after adjustments
==============================================================
        Variable        Coefficient   Std. Error   t-Statistic   Prob.
==============================================================
           C             4.365541     3.655289     1.194308     0.0327
         PRICE           0.004325     0.001236     3.497928     0.0005
      HASHRATE(-1)       0.974033     0.007902     123.2712     0.0000
==============================================================
R-squared                0.985691   Mean dependent var      587.6415
Adjusted R-squared       0.985656   S.D. dependent var      646.7101
S.E. of regression       77.45297   Akaike info criterion   11.54087
Sum squared resid        4901153.   Schwarz criterion       11.55810
Log likelihood          -4728.757   Hannan-Quinn criter.    11.54748
F-statistic              28140.90   Durbin-Watson stat      2.524593
Prob(F-statistic)        0.000000
==============================================================
```

Fig. 8. The results of Least Squares

Figure 8 shows that Adjusted R-squared = 0.985656, DW = 2.5245, p-values for both t-test and F-test are less than 0.05, and given that the model is significant. Substituting the coefficient into (4.7) gives:

$$Y_t = 4.3655 + 0.0043X_t + 0.9740Y_{t-1} \tag{4.8}$$

Then, $\beta = 0.0043$, $\gamma = 0.9740$, $\alpha = 167.9038$. Substituting the values of β and γ into (4.3) obtained:

$$Y_t = 167.9038 + 0.0043X_t + 0.0042X_{t-1} + 0.0041X_{t-2} + 0.0040X_{t-3} + \cdots \tag{4.9}$$

4.4 Research Results

In Eq. (4.1), $\sum_{i=0}^{p} \beta_i$ is called the long-term or equilibrium multiplier. It indicates that when X changes a unit, the total influence of Y on average is caused by lagging effect.

In Eq. (4.9), $\sum_{i=0}^{\infty} \beta_i = \sum_{i=0}^{\infty} \beta\gamma^i = \beta/(1-\gamma) < \infty$. So what should be the lag period? The distributed lag model gives the definition of the average lag period s:

$$s = \left(\sum_{i=0}^{\infty} (i+1)\beta_i\right) / \left(\sum_{i=0}^{\infty} \beta_i\right) \tag{4.10}$$

The average lag period is defined as the weighted average of all lags. According to the Granger causal analysis, the lag period of early price change affects the future hashrate change is one month. $S = 13.92$ is obtained by replacing the value of β_i into Eq. (4.10). Which is given that the price modification is the Granger cause of the hash rate change, and the norm lag period of time is about fourteen days.

5 The Pricing Strategy

Bitmain [14] is a Chinese IC design company established in early 2013. It specializes in research, development, and sales of bitcoin-specific mining chips and miner. The AntMiner S9 [15] is a highly available and low power consumption mining equipment developed by Bitmain. The BM1387 chip used by S9 is bitcoin's fifth-generation bitcoin miner chip independently developed by Bitmain. It is by far the lowest power chip in the history of bitcoin chips. For a given type of miner, before they leave the factory, its rated hash rate and power are fixed. The AntMiner S9 has a rated hashrate of 11.85 TH/s (1TH/s equals 10^{12} hash/s) and the power consumption at the wall is 1172 W.

As we derived, since the modification in bitcoin price is a Granger cause of change in hashrate, the average lag period is 14 days. At the same time, the rated hashrate of each mining machine is fixed. Therefore, the price change of the mining machine is caused by the price change of bitcoin. The same lag period is 14 days. In other words, the price of bitcoin 14 days ago is in one-to-one correspondence with the price of the current miner.

Compared with bitcoin prices, miner prices are lagging behind. The main selling of the miner market is the miner futures. In other words, the future miner's price is determined by the price of bit coin in the previous period. The market phenomenon is consistent with our findings.

5.1 Survey and Forecast

On February 28, 2018, the survey results of the Shenzhen Huaqiangbei miner's market revealed that the AntMiner S9 had a futures price of 3018.03 dollars. Bitcoin prices and tracking surveys of miner's prices are shown in the Table 1 below:

Table 1. The price of bitcoin and miner futures

Date	Bitcoin's price ($)	Price of miner futures ($)
2018/2/28	10607.00	3018.03
2018/3/10	9252.00	2382.66
2018/3/20	8641.00	2223.81

On April 1, 2018, the price of bitcoin was 6,826 dollars. Using the linear regression model, the forecasted miner's futures price is 1432.77 dollars.

Figure 9 is the scatter plot of bitcoin's price and price of miner futures, and the bitcoin's price is expressed by abscissa, and the price of miner futures is expressed by ordinate. The straight line in Fig. 9 is linear regression line. When the bitcoin's price is 6826 dollars, the price of the miner futures will be 1432.77 dollars.

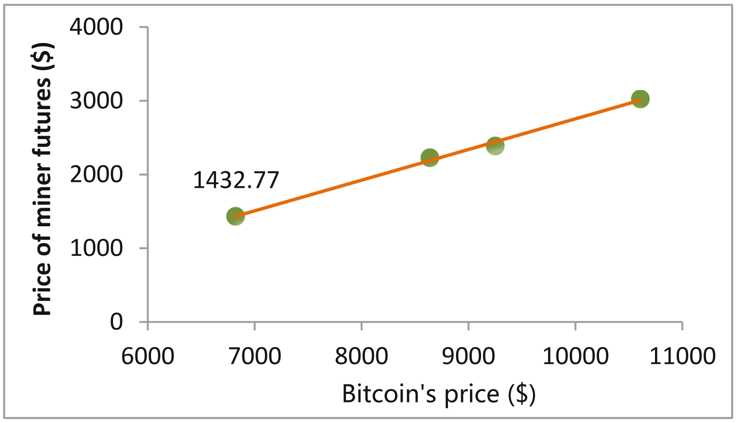


Fig. 9. The linear regression model between bitcoin's price and the price of miner futures

5.2 Verification of Results

According to the inquiry from the official website of Bitmain, the latest futures price of the AntMiner S9 is 1509.01 dollars. In Fig. 10, the broken line represents the prediction price of the miner futures, and the diamond square dot represents the actual price of the current miner futures.

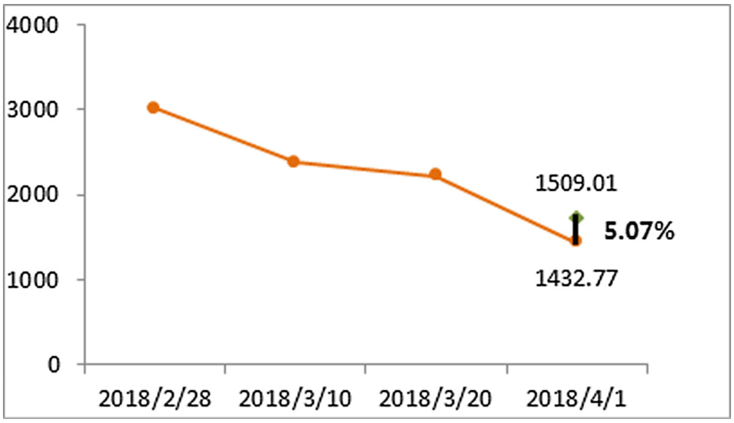


Fig. 10. The price trend chart of miner futures

$$\text{the error rate} = \frac{|\text{predicted value} - \text{actual value}\,|}{\text{actual value}} \times 100\% \tag{4.11}$$

In April 1st, the prediction price of AntMiner S9 is 1432.77 dollars, and the actual price is 1509.01 dollars. In the form of (4.11), the error rate is 5.07%.

The error rate is small, which indicates that our distributed lag model is scientific and reasonable. We can predict the price of future miners based on the early price of cryptocurrency.

From the perspective of economics, the cryptocurrency miner can be used to mine and bring huge gains, which is the value of the cryptocurrency miner. In the cryptocurrency miner market, value determines prices, prices are affected by supply and demand, and they fluctuate around value. When supply is constant and demand increases, prices will rise; when demand does not change and supply increases, prices will fall.

For miner's suppliers, the goal of pricing strategy is to obtain profits and to raise the market share. Profit is the most important reference of funds for suppliers. When suppliers have a dominant position in the product market place, they will object to maximize net gains. This pricing strategy is based on a comprehensive consideration of various factors within a certain period of time. Based on the maximum difference that equals the total revenue minus the total cost, the price of a unified product is determined to obtain the maximum profit. The maximum profit is the maximum total profit that the supplier may be prepared to realize in a certain period of time, not the highest price of the unit product. The maximum price may not be able to obtain the maximum profit.

For a given miner, its profit and rate of return are determined by the price of the cryptocurrency. In essence, the price of miners is based on the price of the previous cryptocurrency, and the lag period is 14 days. This is the pricing strategy for the miner's market.

For investors, they are the consumers of the miner's market. Investors are also buying miners for maximum profit. The profits made by investors are affected by both the price of the cryptocurrency and miner. The higher the price of the cryptocurrency, the shorter the return period of the investment, the higher the profit and the lower risk; the higher the price of the miner, the longer the return period of the investment, the lower the profit and the greater risk. This paper studies the law of cryptocurrency price that determines the price of miners. Based on this strategy, investors can more accurately determine the future price trend of the miner's market, so as to effectively avoid investment risks, and make the investment's profits tend to maximize.

5.3 Conclusion

The current cryptocurrency price determines the future price of miners, and the lag period is 14 days, which is a common pricing strategy for the cryptocurrency miner's market.

Acknowledgement. This work is partially supported by the technical projects No. 2017YFB08 02703, No. 2016YFB1000803, No. 2017YFB1400604, No. 2012FU125Q09, No. 2015B0101 31008 and No. JSGG20160331101809920.

References

1. https://trends.google.com/trends/explore?q=bitcoin. Accessed 03 Apr 2018
2. Yang, B., Chen, C.: The Principle, Design and Application Of Blockchain, pp. 80–82. China Machine Press (2018)
3. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. Consulted (2008)
4. Kristoufek, L.: What are the main drivers of the Bitcoin price evidence from wavelet coherence analysis. PLoS ONE **10**(4), e0123923 (2015)
5. Bouri, E., Molnár, P., Azzi, G., et al.: On the hedge and safe haven properties of Bitcoin: is it really more than a diversifier. Finance Res. Lett. **20**, 192–198 (2016)
6. Indera, N.I., Yassin, I.M., Zabidi, A., et al.: Non-linear autoregressive with exogeneous input (NARX) Bitcoin price prediction model using PSO-optimized parameters and moving average technical indicators. J. Fundam. Appl. Sci. **9**(4S), 791–808 (2017)
7. Xie, P., Shi, W.: A literature review of cryptocurrency. J. Financ. Res. 1–15 (2015)
8. Ahamad, S., Nair, M., Varghese, B.: A survey on crypto currencies. In: International conference on Advances in Civil Engineering AETACE (2013)
9. Bouoiyour, J., Selmi, R.: Bitcoin: a beginning of a new phase. Econ. Bull. **36**(4), 1430–1440 (2016)
10. Hayes, A.: Cryptocurrency value formation: an empirical analysis leading to a cost of production model for valuing Bitcoin. In: Mediterranean Conference on Information Systems (2015)
11. Cao, Y., Mao, J., Li, X.: Applied Econometrics, pp. 250–265. China Social Sciences Press, Beijing (2016)
12. Zhao, G., Fan, H.: Eviews/Stat Econometrics Primer, pp. 119–131. China Renmin University Press (2014)
13. Wang, B.: Econometric Model and Application of R Language, pp. 191–196. Peking University Press (2015)
14. https://www.bitmain.com. Accessed 03 Apr 2018
15. https://shop.bitmain.com/product/main?locale=zh. Accessed 03 Apr 2018

Short Paper Track: Fundamental Research

FBaaS: Functional Blockchain as a Service

Huan Chen[1,2](✉) and Liang-Jie Zhang[1,2]

[1] National Engineering Research Center for Supporting Software of Enterprise Internet Services, Shenzhen, China
[2] Kingdee Research, Kingdee International Software Group, Shenzhen, China
huan_chen@kingdee.com

Abstract. Serverless architecture has been gaining popularity in the last three years. Function as a Service (FaaS) is a concrete realization of the Serverless architecture and has several advantages and features. This paper proposes a new service model which is based on FaaS model, named FBaaS – Functional Blockchain as a Service. Compared with Blockchain as a Service (BaaS), FBaaS has a lighter implementation of top-level business logics, which brings a number of advantages. Firstly, it could improve the operation speed of a blockchain. Secondly, the continuous advances in high robustness, high availability of the underlying FaaS network can be naturally adapted to the FBaaS because of its hierarchical architecture. Thirdly, FaaS implements higher level of abstraction of the logics that is much succinct. Moreover, this paper proposes an abstraction method in the realization of a business logic of consortium blockchain that could further improve the performance. In this paper, we also unfold the details of a concrete example network, which is the conference blockchain network for Services Conference Federation (SCF) 2018.

1 Introduction

Serverless architecture [6,16] has been gaining popularity in the last three years. Function as a Service (FaaS) [10,12] is a concrete realization of the Serverless architecture and has several advantages and features. This paper proposes a new service model which is based on FaaS model, named FBaaS – Functional Blockchain as a Service. Compared with conventional Blockchain as a Service (BaaS) [8,9,11,13], FBaaS has a lighter implementation of top-level business logic, which brings a number of straightforward advantages. Firstly, it could improve the operating speed of a blockchain. Secondly, the advances in high robustness, high availability of the underlying FaaS network can be naturally adapted to the FBaaS because of its hierarchical architecture. Thirdly, FaaS implements higher level of abstraction of the logics that is much succinct [12]. This paper proposes an abstraction method in the realization of business logic of consortium blockchain that could further improve the overall performance. In this paper, we also unfold the details of a concrete example network, which is the conference blockchain network for Services Conference Federation (SCF)

2018 [14]. In the future, the proposed technologies could be partially applied to public chains, such as the [13].

This paper is organized as follows: Sect. 2 introduces the concept of serverless architecture. Section 3 briefly introduces the BaaS. Section 4 illustrates the proposed ideas of FBaaS. Section 5 concludes the paper and outlines some future research directions.

2 Serverless

DevOps (Development and Operations) and Serverless architecture have been gaining popularity in the last three years [6,10,12]. In particular, microservices architecture significantly boosts the development of DevOps. Microservices architecture helps to split a monolithic application into several smaller services, which are self-governed and can be easily developed and maintained by smaller development team. Serverless architecture [6,12] produces a more granular division of services than microservices architecture that gives four-fold advantages to applications (Fig. 1):

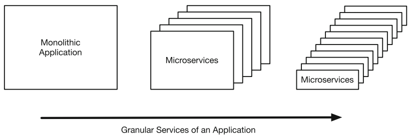


Fig. 1. Monolithic application, microservice application and serverless application

– It reduces costs. Serverless produces smaller functional blocks than monolithic application and microservices, which reduces overhead in larger functional blocks in a way that serverless only requires on-demand resources and gets thereby allocated.
– It accelerates the execution. Because of service is divided into smaller functions, only the needed functions are called at the run time. Those unnecessary functions can be fully stopped instead of standbying that indeed occupies resources.
– It allows better resilience. Smaller blocks somewhat mean shorter start time that gives faster response to the callers. In addition, because less overhead exists, resource allocation can be much more flexible and efficient.
– It enables event-driven applications. Serverless is actually an IPO (Input-Processing-Output) model, which suits the event-driven applications, for instance IoT (Internet of Things) applications. A functional block of serverless architecture is only triggered when it is really required. It would be destroyed immediately once the caller does not ask for its capability. In other words, the life cycle of the function is shorter than the ones of the monolithic and microservices architectures. As a result, event-driven application can leverage lighter framework, such as the serverless, for much more agile execution.

3 Blockchain as a Service

Block Chain as a Service (BaaS) [8,9,11,13] offers the services for blockchain manipulations and building, shipping and running the business logics over blockchain networks. The well-known solutions include but not limited to IBM blockchain [8], Ethereum Blockchain over Azure [13], Microsoft Azure Blockchain [11] and R3 Corda [9]. Although many vendors provides different BaaS, the existing solutions are mainly using the traditional 3-layer cloud architecture: IaaS, PaaS and SaaS. This paper proposes the FBaaS that has significant difference when compared with the conventional BaaS offerings.

4 Functional Blockchain as a Service

This section proposes the FBaaS – Functional Blockchain as a Service in details, by unfolding its architecture, realization process, and a concrete example network. Figure 2 shows the detailed FBaaS architecture.

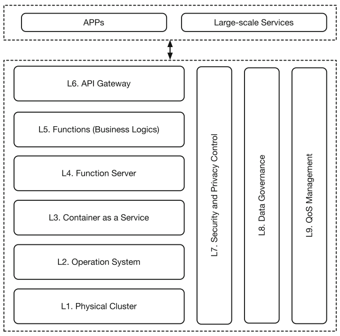


Fig. 2. The architecture of Functional Blockchain as a Service

4.1 Architecture

The underlying architecture of FBaaS partially follows the ideas of CCOA [17], but without the ESB [15]. More concretely, the system is implemented in accordance with the Big Data Open Architecture (BDOA) [7]. The layers of BDOA that have been developed and deployed in the realization of FBaaS are:

- Layer one: Infrastructure layer, realized FBaaS layers 1 and 2. The system is built onto AWS [1], without using any Amazon Lambda functions, but only the normal components such EC2 are used. In other words, we do not use the serverless framework nor the function as a service given by AWS.
- Layer two: Components layer, realized as FBaaS layers 3. The basic functions are implemented inside this layer. For example, authentication functions and authorization functions are developed here to be frequently reused in the upper layer.

- Layer three: Services layer, realized as FBaaS layers 4 and 5. This layer implements the majority of functions of BaaS. We will unfold the details later.
- Layer four: Business logic layer, realized as APPs and Large-scale Services in Fig. 2. By grouping the functions of layer three, we are able to easily establish a blockchain and the complicated large-scale services.

4.2 Functions in the Services Layer

Function 1: Object Storage. Object storage is with strong atomic, which means the entire object storage process is either successful or failed and there is no intermediate/uncertain status during the storage process. At the same time, object uploaded is complete and no breakpoint point resuming uploading is allowed. The strong consistency of such an object storage scheme brings much convenience to the users. Users are no need to worry about the problems of eventual consistency existed in the distributed blockchain networks (Fig. 3).

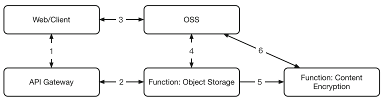


Fig. 3. The workflow of object storage and encryption

Function: New Transaction. This function adds the content of a transaction into a new block, which is then to be encrypted and mined. There is no limitation for the content of a transaction, however, for the sake of storage, it is good to have a limit. Moreover, it is better to make abstractions for the content that we will unfold the details in the remainder of this paper.

Function: Mine. Mining a block is actually a process of simple enumeration and hashing. To accelerate the mining process, one can apply CUDA [2] of NVidia GPUs. In the context of FaaS, it is somewhat difficult to directly apply. Instead of the traditional approach, we introduced the messaging system Kafka [3] for distributing the hashing requirement in an extremely speedy way. Normally, the performance reaches as high as 40x compared with the CPU hashing approach.

Function: Chain Info. Blockchain information can be fetched by this function. Typical data include the number of blocks and the length of the blockchain. It is important to emphasize that here the length represents the only certified chain among many chains that has not yet been mined.

Function: Resolve. Resolving function tries to resolve conflicts by running consistency algorithms, aiming to ensure that all of the nodes have the unique correct chain.

Function: Add Node. Adding a node is complicated with the conventional solutions. However, with the help of FaaS, we are able to easily add nodes to an existing network. First, we retain the current state into the persistent storage. Then, we replicate one node by Container as a Service (CaaS) replication functions. Afterward, the configuration of the new node is the established by adjusting the configuration files of the replica node. Finally, we restore the state from persistent storage and continue executing the network. The performance of adding the node depends on the size of the existing network. Network manipulation, such as replicating a node, does not take too much and is sometimes trivial. Most of the time is consumed in the sub procedure of storing and restoring the states of the system.

4.3 Realization

Function Server. The system is implemented in Python and Golang. Python is used for the top layer framework and those logics inside functions, which are then to be called frequently and simultaneously. The layers *Function Server* and *Container as a Service* are implemented in Golang. We developed system over the OpenFaaS [12]'s modulars *API Gateway* and *Function Watchdog*, shown as Fig. 4, where the RBAC (Role-Based Access Control) server is used for authentication and authorization of functions. Implementing the RBAC inside the layer *Function Server* increases the complexity of the system and decreases the performance since every function call procedure has to be checked. However, this scheme allows system administrators to limit the abnormal violate calls from inner systems. A typical scenario is the short message service. If the RBAC is only existed in the API gateway, once the function is hacked, the message delivery cannot be controlled. In contrast, RBAC implemented in the *Function Server* can stop only the user whose application server has those hacked functions, and will allow the functions with the same functionalities but different authorization to continue working.

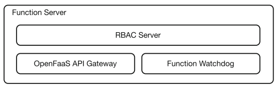


Fig. 4. The inner components of Function Server

Finer FaaS. A microservice provides less functionality than a monolithic system. In other ways, it provides the higher level of abstraction for the service they provide. Decomposing a microservice into several functions can be similar, where FaaS is commonly seen as the one with ultimate level abstraction. On top of FaaS, can we implement a new layer with finer granularity? The certain answer is to freeze some components of FaaS into constant. Formally, we can define the function λ as the abstraction of part of microservice μ:

$$\lambda = abs(\mu(\alpha, \delta)) \tag{1}$$

α denotes the variable part where δ denotes the constant part. When we freeze the δ, the function becomes simpler and its abstraction degree can be further increased. In the realization process, we can introduce some pre-defined function to imitate these constant parts. In Fig. 5, FaaS is in gray and it is further decomposed by introducing the IFTTT [4] and Zapier [5]. This way of decomposing produces finer granularity and increases abstract degree of FaaS.

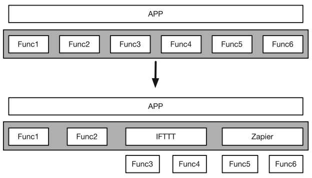


Fig. 5. The finer FaaS by introducing the IFTTT and Zapier

Improving the Performance by Logic Abstraction. In the case of simple rules, if complex transactions are submitted to a node, and in the case of high concurrency, the speed of the consortium blockchain (such as Hyperledger Fabric) will be very poor, which will affect the transaction recording speed and block generation speed. If we reduce the transaction submission frequency by assigning complex rules, we can effectively increase the transaction speed to some extent. For example, we can change the time slot of generating the block and the size of the content of a block. Formally, we abstract the complex things (transactions) in the business system into simple logic through a logical abstraction method, thereby improving the system performance. Such an increase is essentially achieved by reducing the amount of recorded content, where the content is now logic abstraction of actual content. It is important to note that the real state space and abstract state space first satisfy Galois connected:

$$\begin{aligned} \eta(\theta(abs)) &= abs \\ \theta(\eta(real)) &\supset real \end{aligned} \tag{2}$$

This means that if we extract an element from the abstract space, materialize it, and abstract the concrete version, the result is equal to the original. On the other hand, if you select an element from the real space and abstract out a specific abstract version, the final result is the original superset. The specific abstract process is shown in Fig. 6. The original logic, if not abstracted, runs according to the original logic (dashed line) flow. The solid line shows the new process. First of all, a formal review of the logic of the input ensures that the formality of the thing is correct, in line with the characteristics that a transaction should possess. Then, abstract rule matching is performed. The abstract rules mainly include the following types:

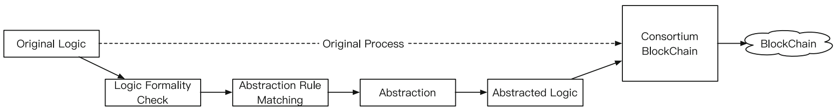


Fig. 6. The abstraction flow

- Only to record the last operation object of the thing;
- Only to record the first object of operation;
- Only to record odd-numbered (or even) atomic-level operation objects;
- Record only Create and Delete operations;
- Only to record the update operation of key data objects;
- Only the fan-in data objects and/or paths affected by the exposed interfaces of the overall system service interface API is recorded;
- Only the key data object changed in the service interface API are recorded, that is, changes are recorded by memorizing the state.

4.4 An Example Network

Figure 7 shows an example network for conference blockchain. The blockchain network consists of five nodes, each of which records the process from 'Call For Paper' ('CFP') to 'Online'. Once a record of a stage, for instance a presentation of a paper in 'Present' process, is bookmarked, such the record will be published to all of the nodes. Throughout the network, all the records will arrive *eventual consistency*. We applied the proposed ideas to the network, benchmarked the network, and found the performance satisfies the requirement of a conference blockchain, where the original required TPS is often smaller than 10.

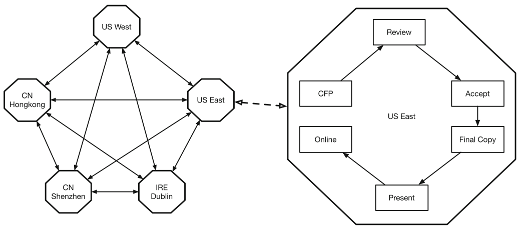


Fig. 7. An example network for conference blockchain

FaaS architecture reduces the overheads and allows developers to focus on the business logic. For example in the conference blockchain, by using FaaS, we are able to concentrate on the 6 business logical stages, namely 'CFP', 'review', 'accept', 'final copy', 'present' and 'online', without taking care of the underlying complex components. Normally in the conventional BaaS solutions, the actual bottleneck always hides in the network transmission, and we had to consider the network itself. With leveraging the FBaas, the performance could be naturally improved with the advanced in the underlying FBaaS dependent components.

5 Conclusion

This paper proposed the first blockchain service model FBaaS over the serverless architecture. We also proposed abstraction method for reducing the complexity of developing business logic over blockchain networks and improving its performance. Not only the consortium blockchain that the proposed FBaas could apply to, but also the partially the public blockchain could. The issue raised when applied to public blockchain might be from the storage oversize that might be further improved by abstraction techniques. As a result, future research directions include developing FBaaS for public blockchain and further improvement of abstraction techniques.

Acknowledgement. This work is partially supported by the projects No. 2016YFB1000803, No. 2017YFB0802703, No. 2017YFB1400604, No. 2012FU125Q09, No. 2015B010131008 and No. JSGG20160331101809920.

References

1. Amazon web services (2006). https://aws.amazon.com
2. CUDA: Compute Unified Device Architecture (2007). http://www.nvidia.cn/object/cuda-about-cn.html
3. Apache Kafka: a distributed streaming platform (2011). http://kafka.apache.org
4. IFTTT (2011). https://ifttt.com
5. Zapier (2011). https://zapier.com
6. The AWS Lambda (2014). https://aws.amazon.com/lambda/
7. BDOA: big data open architecture. Int. J. Big Data (IJBD), **2**, 24–48 (2015)
8. IBM Blockchain (2016). https://www.ibm.com/blockchain/
9. R3 Corda (2016). https://www.corda.net/zh-hant/
10. The serverless framework (2016). https://serverless.com
11. Microsoft Azure Blockchain Solutions (2017). https://azure.microsoft.com/en-us/solutions/blockchain/
12. OpenFaaS (2017). https://www.openfaas.com
13. Ethereum Blockchain as a Service on Azure (2018). https://azure.microsoft.com/en-us/blog/ethereum-blockchain-as-a-service-now-on-azure/
14. SCF: Services Conference Federation (2018). http://blockchain1000.org/2018/about.html
15. Chappell, D.: Enterprise Service Bus. O'Reilly Media Inc., Sebastopol (2004)
16. Fromm, K.: Why the future of software and apps is serverless (2012). http://readwrite.com/2012/10/15/why-the-future-of-software-and-apps-is-serverless
17. Zhang, L.J., Zhou, Q.: CCOA: cloud computing open architecture. In: IEEE International Conference on Web Services, ICWS 2009, pp. 607–616 (2009)

LedgerGuard: Improving Blockchain Ledger Dependability

Qi Zhang[1](✉), Petr Novotny[1], Salman Baset[1], Donna Dillenberger[1], Artem Barger[2], and Yacov Manevich[2]

[1] IBM Research, Thomas J. Watson, Yorktown Heights, USA
q.zhang@ibm.com
[2] IBM Research, Haifa, Israel

Abstract. The rise of crypto-currencies has spawned great interest in their underlying technology, namely, Blockchain. The central component in a Blockchain is a shared distributed ledger. A ledger comprises series of blocks, which in turns contains a series of transactions. An identical copy of the ledger is stored on all nodes in a blockchain network. Maintaining ledger integrity and security is one of the crucial design aspects of any blockchain platform. Thus, there are typically built-in validation mechanisms leveraging cryptography to ensure the validity of incoming blocks before committing them into the ledger. However, a blockchain node may run over an extended period of time, during which the blocks on the disk can may become corrupted due to software or hardware failures, or due to malicious activity. This paper proposes *LedgerGuard*, a tool to maintain ledger integrity by detecting corrupted blocks and recovering these blocks by synchronizing with rest of the network. The experimental implementation of *LedgerGuard* is based on Hyperledger Fabric, which is a popular open source permissioned blockchain platform.

Keywords: Blockchain · Ledger · Dependability · Fault tolerance Hyperledger Fabric

1 Introduction

A distributed ledger is the central component of any blockchain platform. Each peer in the Blockchain network maintains its own replica of the ledger. The ledger is an immutable append-only data structure, which contains a sequence of historical transactions grouped into blocks. The ledger is formed by chaining the blocks together with hash pointers (i.e., a subsequent block contains the hash of its previous block).

The integrity of the ledger is essential for correct functioning of the peer. With corrupted ledger, the peer is not able to generate valid transactions when the smart contract needs to retrieve historical transactions from the ledger. Moreover, when historical transactions recorded in the ledger are requested by external tools such as analytical or auditing applications, the peer first verifies the integrity and validity of the relevant blocks before extracting the transactions. Any corruption of blocks discovered by these operations leads to significant degradation of the peer functionality. Typically, the peer will cease to function till the correct ledger is available. Furthermore, the applications accessing the corrupted data may become significantly impaired as well.

The peer protects its ledger from introducing corrupted data. When a new block is received the peer validates the integrity of the block before appending the block to the ledger. However, the peer lacks the capability of detecting and recovering the corrupted blocks existing in the ledger during its runtime.

A corruption of the ledger may have one of several different causes. Various types of ledger corruptions have been observed on the public Blockchain platforms, such as Bitcoin [16] and Ethereum [10]. For instance, on Ethereum platform users reported corrupted data files due to false positives of antivirus software [5]. Ledger corruptions were also reported by Bitcoin users due to block checksum mismatch [6]. In private Blockchains such as Hyperledger Fabric [7] or R3 Corda [9], it is critical to maintain the nodes hosting peers highly secure. However, when a peer is hosted in a less secure environment, an external attacker or malicious user can hack into the peer node and modify the content of the ledger files. Moreover, since the ledger files are typically stored on a storage medium such as magnetic disks or SSDs, a hardware failure [13,15,17] may also cause a corruption of the ledger files.

In this paper, we introduce *LedgerGuard*, a mechanism that enables the peer to maintain the integrity of its ledger. *LedgerGuard* enforces the integrity of the ledger with the following two techniques. First, it validates the content of each block and the hash links between blocks. Second, if corrupted block is identified, *LedgerGuard* recovers the block and corrects the affected part of ledger without the need for rebuilding the whole ledger. *LedgerGuard* is designed in a highly configurable manner. It can be used as a tool (e.g., by an operator) to validate and correct online or offline ledger. It can also be used as a service of the peer node, to continuously monitor and correct the ledger and thus increase the resiliency and availability of the peer.

2 Background

In this section, we briefly describe the ledger design of Hyperledger Fabric. Although the ledger design varies among different Blockchain platforms, they follow the same principles. Due to the space limitation, we do not describe the details of Hyperledger Fabric design in the paper, but we recommend readers to refer to [11,19,20] for more information.

In Hyperledger Fabric, a transaction is submitted by the client and endorsed by multiple peers. If being successfully endorsed, the transaction with its endorsements will be further sent to the orderer, who collects transactions from multiple clients and organizes them into blocks. After that, the orderer delivers the block to the peers, who finally validates the block and commits it into ledger.

Figure 1 shows the design of the Blockchain ledger in Hyperledger Fabric, which consists of a chain of blocks that are connected by hash pointers. Normally,

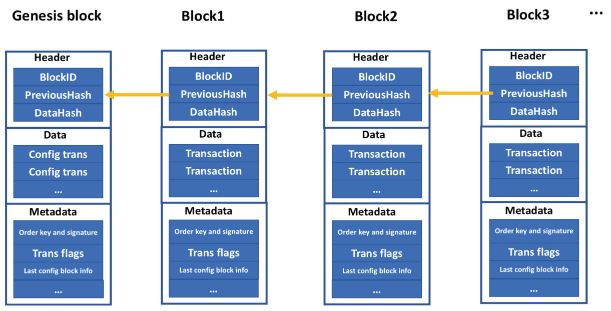


Fig. 1. Hyperledger Fabric Blockchain ledger

a block will never be changed after being committed into the ledger. The first block is called the genesis block, which contains configuration information of this Blockchain network. Blockchain configuration can be changed over time, for example, when a new peer joins or an existing peer leaves. The new configuration transactions will be recorded in the other blocks.

Each block has three sections: block header, block data, and block metadata. The block header section includes the sequence number of this block, the hash value of the previous block, and the hash value of the data section in the current block. The block data section contains a series of transactions with some additional information such as the read/write sets and the endorsers' signatures. For the medadata section, it incorporates the certificate, public key and the signature from the orderer. When creating the block, the orderer signs the block header and stores the signature into the metadata section. Depending on the architecture, a block can be signed by a single or multiple orderers. The metadata section also contains information such as the flags of the validity of each transaction in the block.

3 Ledger Correction

In this section, we describe *LedgerGuard*, which improves the Blockchain ledger dependability by providing a runtime self-correction mechanism for ledger.

Approach Overview. In order to minimize the negative impact brought by the corrupted ledger, we introduce a runtime self-correction mechanism, *LedgerGuard*, for the Blockchain ledger. *LedgerGuard* runs as a service on each peer, checks the integrity of the ledger on the peer, and recovers the corrupted block if there is any. We provide several options for the users to activate *LedgerGuard*. First, it can be setup as a periodically running process in the peer, which is initialized when the peer starts and runs after every period of time. Second, in order to not affect the peer performance, *LedgerGuard* can be triggered by a resource monitor in the peer when the hardware resource utilization, such as CPU, is under a pre-configured threshold. Third, *LedgerGuard* can be provided as a peer built-in tool and explicitly activated by the user when he or she wants to know the integrity of the ledger.

Ledger Corruption Detection. As shown in Fig. 1, the blocks in the ledger are connected by the hash pointers. *LedgerGuard* validates the ledger integrity from two aspects: (1) each single block in the ledger is not corrupted, and (2) the hash pointers between the blocks are valid. In Hyperledger Fabric, a block is created by the ordering service, which signs the block header and stores the signature in the block metadata. Therefore, *LedgerGuard* uses the certificate of the ordering service to validate the correctness of each block header. Since the block header contains the hash value of the block data section and the signature is collected from the block metadata section, a successfully verified signature indicates the block has not been tampered with. We assume the root Certificate Authority in the Hyperledger Fabric Blockchain platform is trusted, thus *LedgerGuard* can get a valid ordering service certificate to validate the blocks. To validate the correctness of the hash pointer, *LedgerGuard* calculates the hash value of the current block (e.g. Hash(block X)), and compares this hash value with the value of "PreviousHash" in the header of block X+1. The hash pointer is integrated if these two value matches. Otherwise, at least one of the two blocks are corrupted.

Corrupted Ledger Recovery. Once a corrupted block is detected by a specific peer (e.g. peer 1), *LedgerGuard* on this peer will send a request to the other peers (e.g. peer 2) in the same Blockchain network, and ask for the block with the same ID as the corrupted block. Since all the peers have the same copy of the ledger, after peer 1 obtains the block from peer 2, it use the approach described in the previous subsection to validate the correctness of this newly received block. If this block is invalid, peer 1 will keep asking the other peers for the same block. Otherwise, peer 1 uses this newly received block to recover the corrupted ledger. Sometimes multiple blocks need to be retrieved to fix the ledger even though only one block is corrupted. In Hyperledger Fabric, a ledger consists of one or multiple fix sized files, and each file contains a continuous series of blocks. A corrupted block can be either larger or smaller than the original block, thus simply replacing the corrupted block with a correct block still breaks the integrity of the ledger. As an example depicted in Fig. 2, block A in file 2 is detected as a corrupted

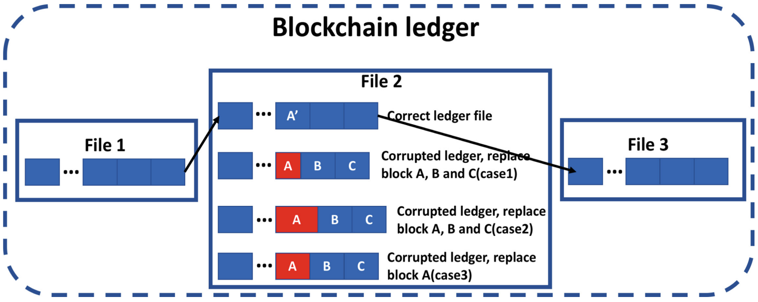

Blockchain ledger
File 1
File 2
File 3
A'
Correct ledger file
A
B
C
Corrupted ledger, replace block A, B and C(case1)
Corrupted ledger, replace block A, B and C(case2)
Corrupted ledger, replace block A(case3)

Fig. 2. Blockchain ledger stored in files

block. Block A' is a correct block retrieved from another peer. If the size of block A is not equal to that of block A', simply replacing block A with block A' will either overwrite part of block B or leave a gap between block A and block B. In order to solve this problem, *LedgerGuard* first checks whether the size of block A has changed. If it is, as shown in Fig. 2, the process will replace all the blocks in file 2 that are subsequent to block A (case1 and case2). Otherwise, only block A needs to be overwritten (case3).

Optimization. As an in progress work, we are exploring optimizations for *LedgerGuard*. For example, since hash value calculation and signature verification are CPU intensive, *LedgerGuard* can use file level verification to decrease its CPU resource consumption. Concretely, when *LedgerGuard* validates the ledger for the first time, it temporarily keeps the validated blocks in memory until all the blocks in a file have been validated. If all the blocks as well as the hash pointers are correct, *LedgerGuard* calculates the hash value of the whole file. The hash values of the files will be kept by system administrators in a separate secure storage. Therefore, when the same portion of the ledger needs to be checked for a second time, only the hashes of the files need to be calculated and compared. Since a file usually contains many blocks, this will largely reduce the amount of hash value calculations. The linkage between the two files can be validated by checking hash pointer between the last block of the previous file and the first block of the next file.

4 Evaluation

In this section, we evaluate the effectiveness of *LedgerGuard* on a 4-core VMWare virtual machines, with Intel(R) Xeon(R) CPU E5-2698 2.20 GHz with 4 GB of RAM. The ledgers used in this section is generated by a tool, which closely simulates the blocks generation on a real Hyperledger Fabric Blockchain network. The tool first loads peer and orderer private keys and certificates, then crafts transaction proposals and endorsements signed by the peer private key, and finally batches the resulting transactions into blocks signed with the orderer private key. A Hyperledger Fabric Blockchain network used in this section is setup with 4 peers, and each peer loads the generated ledger.

Figure 3 shows how much time *LedgerGuard* takes to validate all the blocks in the ledger. The ledger size of 1000 blocks, 2000 blocks, 5000 blocks, and 10000 blocks are used. For each ledger size, we vary the block size from 50 transactions per block to 150 transactions per block, and each transaction is 3 KB. We observe that first, with different ledger sizes but the same block size, the larger the ledger is, the longer it takes to finish validation. For example, with 50 transactions per block, it takes 69 s to finish the validation of the ledger with 10000 blocks, while it takes 69 s when there are 5000 blocks in the ledger. This is because *LedgerGuard* sequentially scans through each block in the ledger, the more blocks the ledger contains, the longer time *LedgerGuard* takes to finish validation. Second, with the same ledger size but different block sizes, the larger the block size is, the longer it takes for validation. Taking the ledger with 5000

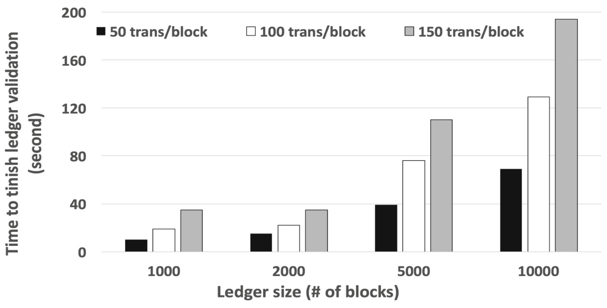


Fig. 3. *LedgerGuard* ledger validation time

blocks as an example, it takes 39 s to finish validation when each block contains 50 transactions, while that number increases to 110 s when each block includes 150 transactions. Our measurement shows that the block hashing time does not vary much when the block size increases from 50 transactions to 150 transactions, and also the order signature verification time is independent of the block size. Therefore, the difference in the ledger validation time is mostly because the larger the block size is, the more time is spent on I/O to read the blocks.

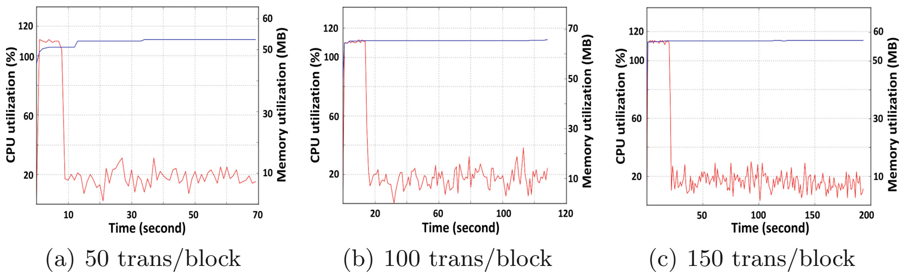


Fig. 4. *LedgerGuard* CPU and memory utilization (ledger size is 10000 blocks)

We also measure how much CPU and memory does *LedgerGaurd* consume during ledger validation, and the results are depicted in Fig. 4. It shows that, no matter for what block size, *LedgerGuard* uses about 60 MB memory, and the CPU utilization of *LedgerGuard* starts with around 110%, then drops to around 20% and stays stable. The reason for the initial CPU utilization spike is *LedgerGuard* needs to do some initialization work such as opening the ledger, reading configuration of the blockchain network, and initializing the MSP (Membership Service Provider) manager. After that, the *LedgerGuard* works as a single process to scan through the ledger and validate each block. Since calculating the block hash value and verifying the signature are both CPU intensive, the *LedgerGuard* occupies the whole CPU core, which leads to around 20% CPU utilization in a 4 core machine.

Moreover, we measure the speed of recovering the ledger. It shows that with the size of 150 transactions per block, a peer node can fetch the block from the other peer, validate and commit it in a speed of 8.5 blocks per second. As part of our on going research, we are working on creating a ledger with different distribution of corrupted blocks, and measure the effectiveness of *LedgerGuard* to recover the corrupted ledger.

5 Related Work

As blockchain technologies gain popularity, issues about the Blockchain platform reliability and security have been observed. Some Ethereum users reported that the Blockchain ledger on his or her machine has been corrupted due to a false positive of antivirus software [5]. This was confirmed by another user who has seen report saying that an antivirus software corrupted an Ethereum Blockchain by deleting some file from the ledger. The suggested solution was to delete the ledger data, and restart the client to re-download the whole ledger. Error of "block checksum mismatch" has also been observed by users of Bitcoin [16], Litecoin [3], and Dogecoin [2] when Btrfs [18] is used. The reason was due to single-bit errors when reading from disk, and the proposed solution was to change the filesystem to EXT4 [12] and re-downloading the whole ledger [6]. Moreover, since smart contracts are programs that could move large value assets on the Blockchain, they always become the victims of attackers who want to steal the assets. The DAO attack [4] showed that a program built on the Ethereum Blockchain platform was breached in a case that results in $50 million worth of Ether being stolen. Researchers and practitioners are making great efforts to improve the reliability and security of the Blockchain platform. Nicola [8] did a survey of the attacks on Ethereum smart contacts by exploiting a series of attacks and providing a taxonomy of programming pitfalls which can lead to such vulnerabilities. Zcash [14] was invented by creating transactions that reveal neither the payment's origin, destination, nor the amount. This approach prevents leakage of the users' spending habits by Blockchain mining [1].

6 Conclusions and Future Work

Blockchain ledger can be corrupted due to many reasons, and ensuring the integrity of the ledger is critical to the functionality and the performance of the Blockchain platform. In this paper, we propose *LedgerGuard* - a mechanism to keep track of ledger integrity by detecting corrupted blocks and recover the ledger by synchronizing it with rest of the network, implement a preliminary prototype, and evaluate its effectiveness and overhead. As the on-going and future work, we are extending and improving the *LedgerGuard* from multiple aspects. For example, we are exploring algorithms to enable the *LedgerGuard* to smartly select the other peers based on the network connection quality when it tries to fetch a block, which will further improve the performance of recovering the ledger. Furthermore, we are investigating more alternative approaches to detect corrupted blocks other than sequential scan.

References

1. Chainalysis. https://www.chainalysis.com/
2. Dogecoin. http://dogecoin.com/
3. Litecoin. https://litecoin.org/
4. Understanding the DAO attack. https://www.coindesk.com/understanding-dao-hack-journalists/
5. Antivirus corrupting ethereum block (2018). https://github.com/ethereum/mist/issues/581
6. Bitcoin Block Checksum Mismatch (2018). https://github.com/bitcoin/bitcoin/issues/6528
7. Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., Enyeart, D., Ferris, C., Laventman, G., Manevich, Y., et al.: Hyperledger fabric: A distributed operating system for permissioned blockchains. arXiv preprint arXiv:1801.10228 (2018)
8. Atzei, N., Bartoletti, M., Cimoli, T.: A survey of attacks on ethereum smart contracts (SoK). In: Maffei, M., Ryan, M. (eds.) POST 2017. LNCS, vol. 10204, pp. 164–186. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-662-54455-6_8
9. Brown, R.G.: Introducing r3 corda: A distributed ledger for financial services. R3, April 5 (2016)
10. Buterin, V., et al.: A next-generation smart contract and decentralized application platform. white paper (2014)
11. Cachin, C.: Architecture of the hyperledger blockchain fabric. In: Workshop on Distributed Cryptocurrencies and Consensus Ledgers (2016)
12. Cao, M., Bhattacharya, S., Ts'o, T.: Ext4: The next generation of Ext2/3 filesystem. In: LSF (2007)
13. Gray, J., Van Ingen, C.: Empirical measurements of disk failure rates and error rates. arXiv preprint cs/0701166 (2007)
14. Hopwood, D., Bowe, S., Hornby, T., Wilcox, N.: Zcash protocol specification. Technical report, 2016–1.10. Zerocoin Electric Coin Company (2016)
15. Meza, J., Wu, Q., Kumar, S., Mutlu, O.: A large-scale study of flash memory failures in the field. In: ACM SIGMETRICS Performance Evaluation Review, vol. 43, pp. 177–190. ACM (2015)
16. Nakamoto, S.: Bitcoin: A Peer-to-peer Electronic Cash System (2008)
17. Pinheiro, E., Weber, W.D., Barroso, L.A.: Failure trends in a large disk drive population. In: FAST, vol. 7, pp. 17–23 (2007)
18. Rodeh, O., Bacik, J., Mason, C.: Btrfs: the linux b-tree filesystem. ACM Trans. Storage (TOS) **9**(3), 9 (2013)
19. Sousa, J., Bessani, A., Vukolić, M.: A byzantine fault-tolerant ordering service for the hyperledger fabric blockchain platform. arXiv preprint arXiv:1709.06921 (2017)
20. Vukolić, M.: Rethinking permissioned blockchains. In: Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts, pp. 3–7. ACM (2017)

Blockchain-Based Research Data Sharing Framework for Incentivizing the Data Owners

Ajay Kumar Shrestha(✉) and Julita Vassileva

University of Saskatchewan, Saskatoon, SK S7N 5C9, Canada
ajay.shrestha@usask.ca, jiv@cs.usask.ca

Abstract. Data sharing practices are much needed to maximize knowledge gain by researchers. However, when and what data should be shared with whom, and how credit should be awarded to the data owner needs to be clearly addressed to create an individual incentive for data owners to share their data. A platform that allows owners to control and get rewards from sharing their data would be an important enabler of research data-sharing, since presently, such incentives for researchers to share their data are largely missing. Our approach delivers a usable blockchain based model for a collection of researchers' data, providing accountability of access, maintaining the complete and updated information, and a verifiable record of the provenance, including all accesses/sharing/usages of the data. Data owners will not only enjoy increased transparency and protection of data from falling into the wrong hands, but they will also be incentivized with digital tokens, acknowledgment, or both to share their data with the interested data seekers, thus becoming active participants that stand to benefit from the research data economy.

Keywords: Blockchain · Smart contract · Incentive · Data sharing
Privacy

1 Introduction

Over the last decade, there has been a huge technological innovation bringing many research consortiums to use data-driven approaches and to collaborate in making intelligent decisions to improve their scientific research activities. Data sharing practices are undoubtedly necessary to maximize the knowledge gain from the research effort. They can also reduce duplicative trials and accelerate the discovery and generation of new ideas for research. However, when and what data should be shared with whom and by what means, and how credit should be awarded to the dataset owner is still the matter of intense debate and research. This research spirit has been further renewed by the emergence of the privacy issues associated with the users' data collected by different parties whose primary motive is to have enhanced model while enabling the maximal research knowledge and scientific benefits. Data Analytics methods can improve significantly the quality of services, but they depend on collecting, sharing and mining research data. The user contributes much of the data voluntarily; others are obtained by the system from observation of user activities, or inferred through advanced analysis of volunteered or observed data [1].

S. Chen et al. (Eds.): ICBC 2018, LNCS 10974, pp. 259–266, 2018.
https://doi.org/10.1007/978-3-319-94478-4_19

In many important domains, for example, medicine and healthcare, both personalized patient care and medical research can benefit from sharing research data from clinical trials in order to maximize the knowledge gain from the research effort [2]. Not often, all possible purposes for use of those data are known in advance, and data owner's consent needs to be asked again, which can be obtrusive to a researcher/data owner who doesn't see what there is to gain. In addition, it becomes hard or even impossible for data owners to remember what consent they have given to which enterprise and to keep track of who accesses their data and for what purpose. A flexible mechanism for obtaining and renewing consent for research data usage and sharing is required that provides appropriate and meaningful incentives to capitalize from data sharing and ensures transparency for researchers to be aware of which of their dataset has been accessed, by whom, for what purpose and under what conditions.

It has been observed that the creativity and the advancement of the technologies have given birth to so many computational backbones to ensure privacy and data sharing across intelligent computing and security against hackers [3]. However, these services are often criticized for the centrality issues, as at most cases, they do not collect and share the diverse fragments of user data coming from the enormous autonomous and independent entities [4]. The trust resides within the centralized service providers for all the storage and management of data [5]. In the past few years, distributed ledgers and blockchain technology have evolved as a promising means to support immutable and trusted records in various use cases including healthcare, agricultural research works, tourism domains etc. In addition, many blockchain systems provide a technology called "smart contract" that allows for building automatic verification of the conditions for access or modification of each data entity. Smart contracts can be deployed to encode allowed purposes of research data use, allowed software applications who can access the data, time limitations, the price for access, etc.

This paper proposes a blockchain based research data sharing framework that incentivizes the dataset owners with digital tokens, proper acknowledgment or both, while giving access to the detailed information of all data to them in an immutable and incorruptible database. The rest of the paper is organized as follows. Section 2 describes the overview of blockchains and smart contract. A brief analysis of the existing architectures with their limitations is given in Sect. 3. After that, Sect. 4 presents the solution architecture and our implemented model for the decentralized data sharing in a research domain while ensuring users' privacy and user control over the data. Finally, Sect. 5 concludes with future directions.

2 Background

2.1 Blockchain

If there were any critical assets going on a supply chain, we could use distributed ledger so that we could see where those goods/assets are, what they are doing and we will also have the trust mechanism behind them so that it will be very difficult for the fraudulent people to inject false goods into the supply chain [5]. This breakthrough technology is also called blockchain. The idea was first stated in the original source code for the digital cash system, Bitcoin [6], but its effect is being observed to be far wider than just the virtual cryptocurrency. The blockchain provides a digital ledger of every record organized in ‘blocks’, which are linked together by cryptographic validation. Each block aggregates a timestamped batch of transactions to be placed in the chain. All those blocks refer to the signature of the previous block in the chain, and that chain can be traced back to the very first genesis block created in the chain.

The blockchain can be private, public or hybrid. MultiChain [7] is best suited for private blockchain which provides the privacy and control required in an easy to configure and deploy package. Unlike any other blockchains, MultiChain solves the problems of mining, privacy, and openness via integrated management of user permissions [7]. Once a blockchain is private, problems relating to scale are easily resolved, since the chain’s participants can control the maximum block size. In MultiChain, all privileges are granted and revoked using network transactions containing a special metadata. The miner of the first “Genesis” block automatically receives all privileges, including administrator rights to manage the privileges of other users. Future versions of MultiChain could also introduce “super administrators” who can assign and revoke privileges on their own.

Similarly, Ethereum is another open source platform to create decentralized applications (dapps) where users interact with the online services in a distributed peer to peer manner that takes place on censorship proof foundation. Developers can create interfaces and business logic with any of the known programming languages and tools. Ethereum has Ether (ETH) as its own virtual currency, which can be used to pay a transaction fee and to provide a primary liquidity layer for exchanging digital assets. There are “messages” in Ethereum being created by either an external entity or a contract unlike the Bitcoin transaction, which can only be created externally [8]. There is also an explicit option for Ethereum messages to contain data and the recipient of an Ethereum message, and if it is a contract account, it has the option to return a response as well. In addition, the Ethereum has “transaction” as the signed data package that stores a message to be sent from an externally owned account. The state in Ethereum is made of accounts each consisting of 20 bytes address and state transitions [8]. We have Ethereum blockchain as a semi-financial application such as on-blockchain escrow, which allows users to enter into contracts and manage them using their ETH to deal with non-monetary assets such as the research data.

2.2 Smart Contracts

The Smart contracts are instances of contracts deployed on some blockchains, for example Ethereum [8]. They consist of different functions that might be called from outside of a blockchain or by other smart contracts. Blockchain coupled with smart contract technology removes the reliance on a central node between the transaction parties. Since the smart contracts are broadcasted on the blockchain, all the connected parties across the entire cryptocurrency network will have a copy of them. A Smart contract, as an important piece of software, stores the rules that negotiate the terms of the contract, automatically verifies the contract and executes the agreed terms [8]. The smart contract can execute agreed stored process when triggered by an authorized or agreed event just like traditional systems. All contract transactions are stored in chronological order for future access along with the complete audit trail of events. If any party tries to change a contract or transaction on the blockchain, all other parties can detect and prevent it. If any party fails, the system continues to functions with no loss of data or integrity. Therefore, it creates a single large secure computer system logically, without the risks, costs and trust issues of a centralized model. The Ethereum Virtual Machine (EVM) code is used in the contracts that consists of bytes, each representing an operation. The code can be written in Solidity language and can access the value of sender and data of the incoming message, block header data, and return a byte array of data as an output.

3 Related Works

According to Bierer et al. [9], data sharing is the "use of research data by persons other than those who originally gathered the data, for no longer a hypothetical or occasional occurrence". Most of the research on data sharing is relevant to the design framework that focus on the optimization of those properties. However, the technical performance of a data sharing system alone does not guarantee the practicality of the systems. Decentralized approaches for data sharing has achieved the research trend in order to overcome the limitations brought by the centralized architecture, which has a predefined point of access that leads to the central point of failure.

A few prominent examples of data sharing systems include online P2P file-sharing networks and data management systems, collaborative repositories such as Wikidata [10] etc. Almost all of these systems implement different architectures and their evaluation is based on different non-functional requirements, such as efficiency, scalability, or reliability [11]. With regard to data sharing in the cloud, Liu et al. proposed an incentive mechanism into rational secret sharing schemes and a fair data access control scheme for cloud storage [12]. In the scheme, the decryption key reconstruction activity is to be formalized, and then its security, fairness, and correctness are defined. Afterward, the decryption key obfuscation is performed with a generation of a large number of fake keys over the shared data. When rational users exchange the shares, they adjust the action order through the agreed term.

Ozzie et al. [13] provided an architecture that facilitates user-controlled access to user profile information. A user is allowed to selectively mask (expose) portions of her profile to third parties. Advertisers and content providers can offer incentives or enticement in response to the acceptance of which a user exposes larger portions of their profile. Online social network, persona as in [14], can allow users to choose and define the rules who they want to share their personal information like photographs with and browse through highly sensitive data on web pages. It uses attribute-based encryption and public key cryptography to hide data and provide the flexibility needed. For the decryption and authentication by groups and users, it uses group-based access policies. Persona can perform just as well as the existing online social network with added privacy features. Similarly, Houdini framework enables the sharing of context-aware and privacy-conscious user data for global computing [15]. It comes with a method to collect data from various sources focusing on how and when to share them. It is built with an infrastructure to manage principles focusing on the preferences.

The main parts of the infrastructure are how well the underlying rules perform and providing the preferences by itself.

In the medical field, the research committee is increasingly recognizing the importance of sharing patients' level data from clinical trials. European Medicines Agency (EMA), a number of drug companies and one other trial funder have already implemented data sharing [2]. However, the issue with them is to address the appropriate and meaningful incentives to capitalize on the promise of data sharing, and of course, they rely on a centralized system for the data storage and management. Most importantly, with regard to sharing research data, well-developed incentive mechanisms in online communities positively motivate the users to willingly engage in knowledge sharing with others [16]. Next section will provide the methodologies and discussions on our model.

4 Solution Framework and Discussions

Figure 1 presents our general solution architecture that introduces a new way of incentivizing the users for sharing their research data. We have introduced blockchains to share the data among registered parties/enterprises in their private network by incorporating automatic contract so that access-control policies would be stored securely on the blockchain. A user can register into the system by providing her basic profile information, public wallet-address and activate the smart contract, which automates the functionality to support the user-controlled privacy in order: (a) to give the user full transparency over who accesses their data, when and for what purpose, (b) to allow the user to specify a range of purposes of data sharing, kinds of data that can be shared, and classes of applications/companies that can access the data through the smart contract, and (c) to provide an incentive to the user for sharing their research data (in terms of payment for the use of the data by applications, as specified by a smart contract). This user-incentive model with the blockchain is run by the public (Ethereum) blockchain network nodes.

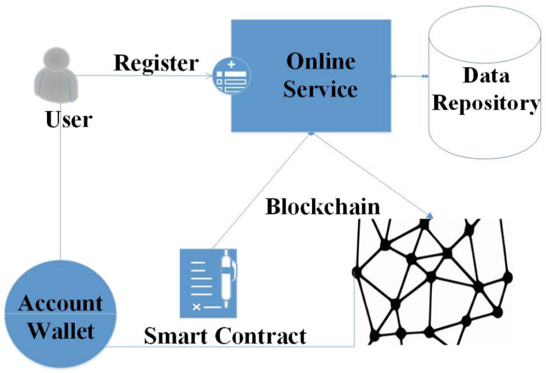


Fig. 1. General user-controlled privacy-preserving data sharing architecture.

Since the smart contract is stored on the public blockchain, the users should have their own digital token addresses safely stored in their personal wallet. Once the users' data are being used by any other participating parties, then the corresponding users will be incentivized with the digital tokens (ETH). And similarly, for sharing the data among enterprises, private (MultiChain) blockchains are installed on each participating registered node, which can publish the items (research dataset) into the stream to be shared among other nodes in the network.

With reference to this proposed general model, the actual implementation is portrayed with the solution framework in Fig. 2. One of the elements of data sharing would be to whom the data are available for sharing and by what means, and how can the researchers/data owners be incentivized either with digital tokens or with acknowledgment of their efforts in collecting the data. Our system clearly guides registered users about what the smart contracts do on their data. With the smart contract in the public Ethereum blockchain, researchers are able to retain the ownership of data with themselves and are incentivized as per the agreed term. Any academic or industrial unit as a data seeker with valid credentials and approval from a local institutional review board (IRB) is eligible to access the data. The local IRB must also be enlisted into the system by providing the certification that it is bound by regulations to look at scientific methods proposed by a node (data seeker) for accessing the research data. Through the smart contract, only the selected eligible nodes can access the items (dataset) by subscribing to the corresponding published streams. The data owner is incentivized as per the negotiation made on the options between the two parties. An acknowledgment can be given to the data owner during the publication of research article and/or a predefined incentive is offered in the form of the digital token by transferring ETH to the data owners' Ether addresses. An escrow service can be optionally added into the system so as to bind the users with legal obligations. The access-control policies are stored securely on the blockchain while retaining the same user-interface.

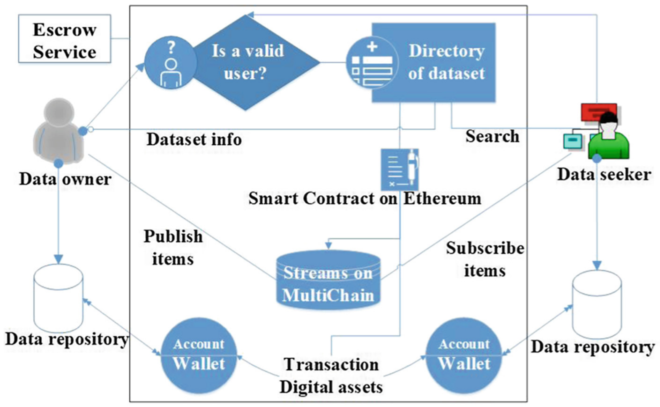


Fig. 2. User-controlled privacy-preserving research data sharing model.

The smart contract is deployed just once for each node on Ethereum blockchain which stores _billingAddress. The smart contract developed with Solidity contains the following functions:

```
contract ShareResearch is tested {
  function ShareResearch(address _billingAddress)
  function getStatus(uint externalIncentiveID) constant
returns (string)
  function getPrice(uint externalIncentiveID) constant
returns(uint)
  function startNewIncentive(uint externalIncentiveID,
uint price) onlyOwner
  function pay(uint externalIncentiveID) payable
  function finish(uint externalIncentiveID) onlyOwner }
```

In order to provide ETH to the data owner (say node1) for accessing the data, a participating eligible data seeker at some node (say node2) queries the system to use the specific filename. The public key cryptography is implemented to ensure the authenticity of the eligible users requesting the file. This results in the execution of the startNewIncentive function of smart contract with the incentiveID and total incentive to be paid to the data owner. The incentiveID is generated for the data owner during registration. Node2 invokes pay function of the smart contract with the incentiveID of Node1 and the ETH to be sent as an incentive to the data owner. The contract verifies the two parameters and then it receives the ETH and updates the status accordingly. It then calls the getStatus function to get the status and with the confirmation of ETH being provided by Node2, data is made available to the data seeker and finally calls the finish function to transfer ETH to the _billingAddress. The ETH is made available in the node1's account since the incentiveID is paired with the ETH address of the data owner. Thus, the data seeker is entitled to the data while incentivizing the corresponding data owner. There is also no scalability limit in terms of node count for MultiChain blockchain as demonstrated in [17], because each node does not need to connect to every other node to create a fully connected peer-to-peer network. However, for all the node catch-up time, new nodes joining the chain have to replay all transactions from the beginning, and so it can take them significant time before they are up-to-date. The exact amount of time will also depend on how many blocks and transactions are in the chain.

5 Conclusions

In summary, our paper presents a decentralized framework for incentivizing researchers for sharing their research data that provides a way to specify/control the parameters of sharing and providing full accountability of access to such data. The security, scalability, and privacy of those systems are gracefully realized by the implementation of the smart contract and blockchains, which can offer the secure distributed research data-sharing network. Our future works include improving the current model by studying users' attitudes to research data sharing with blockchain and the incentives they would find attractive for sharing their assets. We will also evaluate usability and usefulness of the approach, and the trust users can have in the system.

References

1. Poslad, S.: Ubiquitous Computing: Smart Devices, Environments and Interactions. Wiley Publishing, New York (2009). ISBN: 978-0-470-03560-3
2. Lo, B., DeMets, D.: Incentives for clinical trialists to share data. N. Engl. J. Med. **375**(12), 1112–1115 (2016). https://doi.org/10.1056/nejmp1608351
3. Shrestha, A.K.: Security of SIP-based infrastructure against malicious message attacks. In: Proceedings of the 8th International Conference on Software, Knowledge, Information Management and Applications. IEEE (2017). https://doi.org/10.1109/skima.2014.7083519
4. Dolog, P., Vassileva, J.: Decentralized, agent-based and social approaches to user modeling. In: Workshop DASUM-05, at the 9th International Conference on User Modeling (UM 2005), Edinburgh, Scotland (2005)
5. Shrestha, A.K., Vassileva, J.: Towards decentralized data storage in general cloud platform for meta-products. In: Proceedings of the International Conference on Big Data and Advanced Wireless Technologies (BDAW 2016). ACM, New York (2016)
6. original-bitcoin/main.h at master · trottier/original-bitcoin · GitHub. https://github.com/trottier/original-bitcoin/blob/master/src/main.h#L795-L803. Accessed 10 Feb 2017
7. Greenspan, G.: MultiChain Private Blockchain — White Paper, 1st edn. (2015)
8. Buterin, V.: A next-generation smart contract and decentralized application platform. In: GitHub White Paper (2010). https://github.com/ethereum/wiki/wiki/White-Paper. Accessed 19 Sep 2017
9. Bierer, B.E., Crosas, M., Pierce, H.H.: Data authorship as an incentive to data sharing. N Engl. J. Med. **376**(17), 1684–1687 (2017). Retrieved from http://www.nejm.org/doi/pdf/10.1056/NEJMsb1616595
10. Vrandec, D., Krotzsch, M.: Wikidata: a free collaborative knowledgebase. Commun. ACM **57**(10) (2014)
11. Davoust, A.: Decentralized Social Data Sharing. Thesis report. Ottawa-Carleton Institute for Electrical and Computer Engineering (OCIECE) (2015)
12. Liu, H., Li, X., Xu, M., Mo, R., Ma, J.: A fair data access control towards rational users in cloud storage. Inf. Sci. **418**, 258–271 (2017)
13. Ozzie, G.W.F., Horvitz, E.J., Goodman, J.T., Brill, E.D., Brunell, B.A., Dumais, S.T., Ozzie, R.E.: User-controlled profile sharing (2009)
14. Starin, D., Baden, R., Bender, A., Spring, N., Bhattacharjee, B.: Persona : an online social network with user-defined privacy categories and subject descriptors. In: Sigcomm 2009, pp. 135–146 (2009). https://doi.org/10.1145/1592568.1592585
15. Hull, R., Kumar, B., Lieuwen, D., Patel-Schneider, P.F., Sahuguet, A., Varadarajan, S., Vyas, A.: Enabling context-aware and privacy-conscious user data sharing. In: Proceedings on IEEE International Conference on Mobile Data Management, pp. 187–198. IEEE (2004). https://doi.org/10.1109/MDM.2004.1263065
16. Chen, C., Chang, S., Liu, C.: Understanding knowledge-sharing motivation, incentive mechanisms, and satisfaction in virtual communities. Soc. Behav. Pers. Int. J. **40**(4), 639–647 (2012)
17. Shrestha, A.K., Vassileva, J.: User-controlled privacy-preserving user profile data sharing based on blockchain. In: Proceedings of the Future Technologies Conference (FTC), Vancouver, Canada. IEEE (2017)

A Novel Blockchain as a Service Paradigm

Zhitao Wan[1], Minqiang Cai[2](✉), Jinqing Yang[1], and Xianghua Lin[1]

[1] Institute of Advanced Technology Research, Ge Lian Corporation, Hangzhou, China
zhitao.wan@pku.edu.cn

[2] Shanghai University of Finance and Economics, Shanghai, China
caiminqiang@126.com

Abstract. Blockchain brings a new vision for decentralized and trusted systems with more security, resiliency and efficiency. But the barrier of deployment is relative high and BaaS (Blockchain as a Service) emerged to meet the challenge. The current mainstream BaaS providers usually adopt APIs for user access or simply migrate blockchain application to Cloud. Obviously these BaaSs are provided in centralized way which erodes the trustless mechanism and incurs lock-in risk. This paper proposes a novel service paradigm to overcome the limitation of current BaaS. The paradigm adopts deployable components to reconstruct the open and decentralized blockchain service with corresponding long-life encapsulated blockchain service. The meta blockchain service generates dedicated easy to deploy encapsulated components to build up more parties BaaS network to guarantee decentralization, auditability and better efficiency. The encapsulated blockchain service components are the seeds of the whole blockchain system. A hyperledger based implementation demonstrates the paradigm can alleviate the erosion of characteristics of blockchain and lock-in risk caused by current BaaSs.

Keywords: Blockchain as a Service · Cloud computing · Service paradigm

1 Introduction

Blockchain is a type of distributed ledger in which value exchange transactions are sequentially grouped into blocks. Each block is chained to the previous block and immutably recorded across a peer-to-peer network, using cryptographic trust and assurance mechanisms. Blockchain offers a secure way to exchange any kind of digitalized assets and build up trusted partnerships. Blockchains have shaken up the financial industry and more agencies now believe the technology could rejuvenate the public sector. Proponents argue that its immutability will protect records from fraudsters, its transparency will keep employees accountable, and its ability to automatically process new entries can make agencies more efficient. The blockchain market is set to grow at a CAGR of 61.5% by 2021, with transparency and immutability as the driving factors behind the exponential growth of the blockchain market [1].

Nordrum reported in IEEE spectrum that Dubai wants one single software platform on which agencies will launch different blockchain projects, while Illinois designed a more experimental process that individual projects testing different types of blockchain

S. Chen et al. (Eds.): ICBC 2018, LNCS 10974, pp. 267–273, 2018.
https://doi.org/10.1007/978-3-319-94478-4_20

platforms and applications to find the best fit for their particular needs [2]. Design of blockchain system is a relative complex and error-prone process. The P2P interconnection, file storage, consensus mechanism and application should be elaborately developed and tested [3]. To address the challenge of complexity, Tech giants have jumped on the bandwagon and they are providing BaaSs through their built-in platforms and collaborations. BaaS is set to grow further and become the latest revolution worldwide that makes the mass adoption of blockchain technology happen.

This paper is an effort to improve the current emerging BaaSs. We propose a decentralized and trusted BaaS paradigm to keep the core characteristics of blockchain from the cloud migration incurred erosion. The rest of paper is organized as follows. Next section introduces blockchain and encapsulated BaaS and its limitation. Then we propose our novel encapsulated BaaS paradigm. A hyperledger based proof of concept is followed. In conclusion and future work, we discuss some possible future directions and conclude the paper.

2 Current Blockchain as a Service

Blockchain has four key characteristics: *Decentralization*. In centralized systems, the central trusted agency validates each transaction. The third party agency is no longer needed in blockchain. Consensus mechanisms in blockchain are used to maintain data consistency among distributed nodes in blockchain network. *Persistency*. Transactions can be validated and fraud transactions would not be admitted by honest nodes. It is almost impossible to modify a transaction once it has been included in the blockchain. *Anonymity*. Each user can join the blockchain with a meaningless address, which does not reveal the real identity of the user. *Auditability*. Blockchain stores all previous transactions and data. Once a transaction or data change submitted, the related transactions and data could be easily tracked [4]. The primary advantage of BaaS is lower entry barrier and easy to use, but we should check the compliance core blockchain characteristics.

Generally, the BaaS can be classified into two categories: cloudified native Blockchain as a Service (nBaaS) and cloudified encapsulated Blockchain as a Service (eBaaS). The nBaaS is to move blockchain nodes to cloud platform and make them cloud hosted. Compared with nBaaS the eBaaS also provides extra unified interfaces such as REST API. The typical nBaaSs include Microsoft Azure hosted R3's distributed ledger platform, Corda and R3Net, Ethereum [5], Amazon AWS hosted many kinds of blockchain applications [6]. The typical eBaaSs include IBM's Bluemix [7] based Hyperledger [8] with GUI to deploy blockchain instance with REST API interface (see Fig. 1). It depicts an example of a blockchain network that consists of four members with two or three peers for each. An ordering service that defines policies and network participants. Channel 1 is restricted to three members: Banks B, C, and D, channel 2 contains all four network members. A client in possession of a certificate is able to send calls to peers. It's conceivable that a client doesn't even know about the existence of the blockchain. The eBaaS is the focal point of this paper.

Tencent is more aggressive and its TrustSQL [9] is committed to providing enterprise-class blockchain infrastructure supporting secure, reliable and flexible blockchain cloud services (see Fig. 2). Besides its highlight features such as consensus

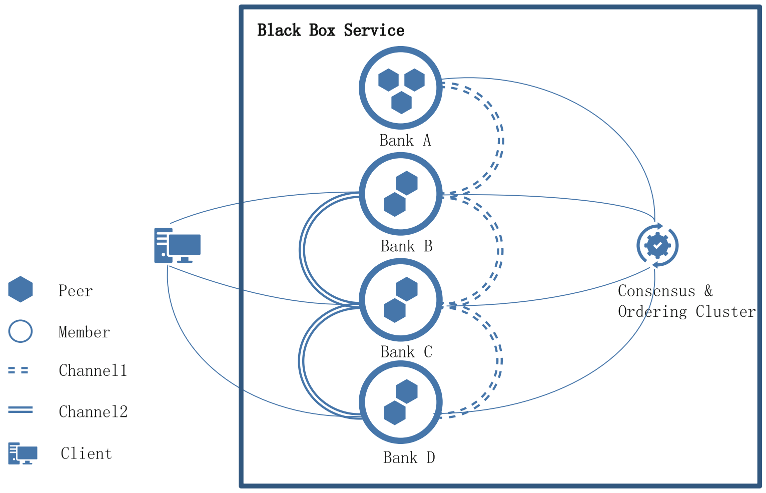


Fig. 1. An IBM BaaS bluemix blockchain network consisting of four members leveraging channels to isolate data.

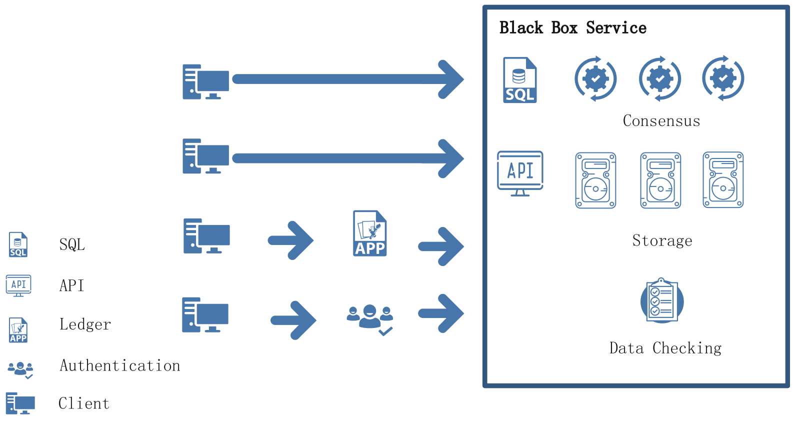


Fig. 2. Typical Tencent TrustSQL blockchain service and its application scenarios.

algorithm, transaction confirmation in seconds, low-cost fast access, security and efficient operation, TrustSQL provides SQL like interface for user access. TrustSQL Platform has a three layered system: core chain layer (Trust SQL), a product service layer (Trust Platform), and an application layer (Trust Application). The platform provides such as digital assets, shared books, proof certificates, stock swaps and proprietary transactions, as per the white paper. But, it only provides SQL-like insert and select operation. Its centralized way erodes decentralized trust machine, the core motivation of blockchain.

Lower layer networking, storage, computing resources are adopted by the higher layer cloud services. As we mentioned the nBaaS has little impact on a legacy blockchain system to be migrated to the cloud. But eBaaS is a black box that erodes the *Decentralization* and *Auditability* of blockchain.

We present a new BaaS paradigm to reconstruct *Decentralization* by providing deployable components, which intends to rebuild *Auditability* by a transparent eBaaS.

3 A Novel Paradigm

The cloudify BaaS should keep the benefit of blockchain and provide more flexibility, convenience, elastic and so on. As we discussed in previous section, the current eBaaS is more like a cluster of distributed nodes that provide a unify access point which breaks down the service side and client side. When a client accesses eBaaS by service interface such as REST API, it is identical to traditional service and the characteristics of blockchain are blur. To keep *Decentralization*, the key characteristic of blockchain, we propose a new paradigm of eBaaS. The new paradigm supports more flexible decentralized workload deployment. It is conceivable that a decentralized system has better *Auditability* as well.

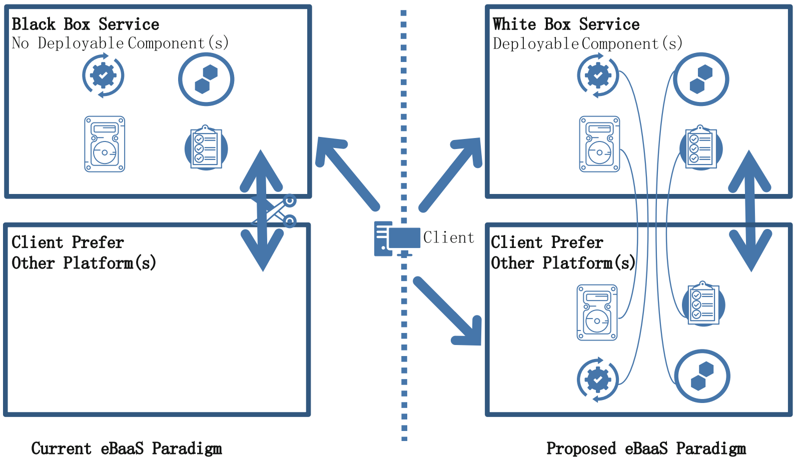


Fig. 3. A new service paradigm for encapsulated BaaS vs. current encapsulated BaaS paradigm.

There are many application scenarios and each instance of a scenario usually invokes a standalone BaaS. e.g., The Tencent blockchain platform has diversified implementations in the following scenarios: Proof of authentication, Smart contract, Shared economy, Digital assets, Shared ledger. IBM blockchain platform provides a way to develop, govern and operate a blockchain ecosystem. A BaaS invokes a few nodes to construct a blockchain network. Conventional service is defined as a black box. But blockchain is preferred to be white box which can be reviewed by public and easy to keep the *Decentralization* and *Auditability*. We solve the problem by adopting deployable component(s) (see Fig. 3), i.e., all components generated for an instance of blockchain are also released as part of the blockchain service. Client can deploy any component to any computing environment such as different cloud services and on-premises platforms.

4 Implementation and Evaluation

We implement the paradigm basing on the Hyperledger. The Hyperledger is a typical consortium blockchain. Docker is used as the container for nodes. Hyperledger Composer is a tool set for building blockchain business networks to create smart contracts and applications. It is built with JavaScript, leveraging modern tools including node.js and etc., Composer offers business centric abstractions as well as sample apps with easy to test devops processes and to create robust blockchain solutions that drives alignment across business requirements with technical development.

4.1 Hyperledger Fabric and Composer

Hyperledger Fabric [10] is a blockchain framework implementation as a foundation for developing applications or solutions with a modular architecture. Hyperledger Fabric allows components to be plug-and-play including consensus and membership services. Hyperledger Fabric leverages container technology to host smart contracts called chaincode. The chaincode is used to construct the application logic of the system. To prepare a Hyperledger Fabric environment, cURL tool, docker and docker compose, go programming language, Node.js runtime and npm and Python are prerequisites. Hyperledger Composer [11] includes the following main components:

Business Network Archive. Capturing the core data in a business network, including the business model, transaction logic and access controls, the Business Network Archive packages these elements up and deploys them to a runtime.

Composer Playground. This Web based tool allows developers to learn Hyperledger Composer, model out their business network, test that network and deploy that network to a live instance of a blockchain network. Composer playground offers a repository of sample business networks that can provide a base for building user own business network.

REST API support and integration capabilities. A Loopback connector for business networks has been developed that exposes a running network as a REST API which can easily be consumed by client applications and integrate non-blockchain applications. It provides that user interface for client to invoke blockchain requests and encapsulates the blockchain for easy to use.

4.2 Hyperledger and Composer Based Implementation and Evaluation

We use shell script to generate corresponding dockers and expose them as part of service using HTTP service for third party deployment (see Fig. 4). The main steps are following:

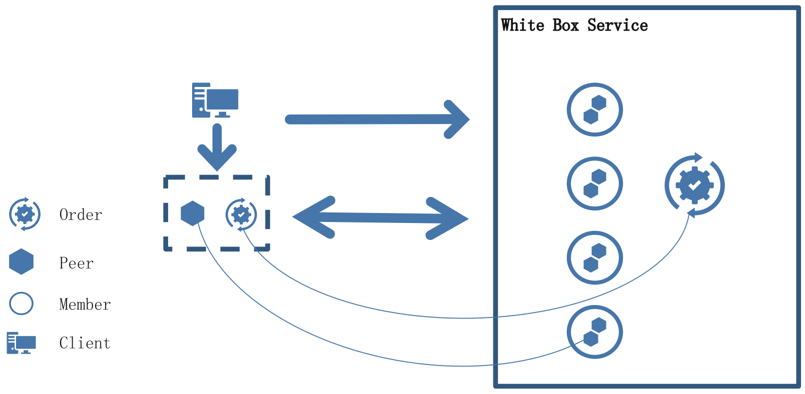


Fig. 4. A new BaaS paradigm instance consisting of four members, one order and client side deployable peer and order.

Define core components. This step includes the definition of assets, participants, transaction logic, and access controls for the business network, which can then be shared across multiple organizations and deployed on different platforms.

Generate JavaScript and REST APIs. This step is basing on the business network definition that can be used to interact with applications, integrate legacy systems, create skeleton applications and run analytics on the blockchain network.

Develop and test Composer playground, and then deploy the business network to a blockchain instance of Hyperledger Fabric.

The advantages of hyperledger and composer base implementation include:

Fast creation of blockchain applications. It eliminates the massive effort required to build blockchain applications from scratch. Edit the shell scripts and configuration file and compile the source code to get it done.

Flexibility. Hyperledger composer is higher-level abstraction. It is easy to connect to existing applications.

Deployable. All the components to deploy are in docker container. They can be easily downloaded and installed.

5 Conclusion and Future Work

The advantages of blockchain based solutions are more widely accepted. The BaaS is used to accelerate the deployment of blockchain. Compare with conventional blockchain it provides a universal access and easy to apply approach. However, current BaaSs erode the core characteristics of decentralization and auditability. The consequence is that the confidence of trust decreased to a lower level similar to legacy databases or services. We proposed a new paradigm to address the challenge. The proposed paradigm introduces deployable components as part of blockchain service. The deployable components can be easily deployed in cloud computing or on-premises environment. The hyperledger fabric and composer based implementation shows the paradigm is applicable for current main-stream blockchain system. The proof of concept system demonstrates an acceptable complexity of deployment. Currently, the deployment requires docker and need some manual installation and configuration. We will try to make it easier to deploy in more different environments in future with automatic installation and configuration. And, how to migrate more blockchains to cloud as BaaSs and find out more paradigms are interesting topics.

References

1. Gartner Homepage. http://www.gartner.com. Accessed 31 Mar 2018
2. Nordrum, A.: Govern by blockchain Dubai wants one platform to rule them all, while Illinois will try anything. IEEE Spectr. **54**, 54–55 (2017)
3. Mehar, M., et al.: Understanding a Revolutionary and Flawed Grand Experiment in Blockchain: The DAO Attack. Social Science Electronic Publishing (2017)
4. Zheng, Z., Xie, S., Dai, H., Chen, X., Wang, H.: An overview of blockchain technology: architecture, consensus, and future trends. In: Proceedings - 2017 IEEE 6th International Congress on Big Data, BigData Congress 2017, pp. 557–564 (2017)
5. Blockchain. https://azure.microsoft.com/en-us/solutions/blockchain/. Accessed 31 Mar 2018
6. AWS Blockchain Partners. https://aws.amazon.com/cn/partners/blockchain/. Accessed 31 Mar 2018
7. IBM Blockchain Platform. https://console.bluemix.net/catalog/services/blockchain/. Accessed 31 Mar 2018
8. Home-Hyperledger. https://www.hyperledger.org/. Accessed 31 Mar 2018
9. TrustSQL. https://trustsql.qq.com/. Accessed 31 Mar 2018
10. Hyperledger Fabric Homepage. https://hyperledger-fabric.readthedocs.io/en/release-1.1/. Accessed 31 Mar 2018
11. Hyperledger Composer Homepage. https://www.hyperledger.org/projects/composer. Accessed 31 Mar 2018

Short Paper Track: Application Researches

A Business-Oriented Schema for Blockchain Network Operation

Sheng He[1,2,3](✉), Chunxiao Xing[1], and Liang-Jie Zhang[2,3]

[1] Research Institute of Web Information, Tsinghua University, Beijing, China
heshengpku@gmail.com

[2] Kingdee Research of Kingdee International Software Incorporation, Shenzhen, China

[3] National Engineering Research Center for Supporting Software of Enterprise Internet Services, Shenzhen, China

Abstract. The earliest proposed blockchain is a completely open operating mode, that is the public blockchain, where all operating nodes can freely join or exit the blockchain network without any restriction. Because of the inefficiency and the private care of the public blockchain, the later proposed consortium blockchain or private blockchain restricts the behavior of joining of the operating nodes according to an advance agreement. The consensus mechanism, the most important feature of the blockchain, is however closely related to the chosen operating mode. Different from the endogenous incentives of the public blockchain, the operating nodes in consortium blockchain are usually based on the extrinsic values from commercial needs, which actually have weakened the incentives of blockchain system. This paper is trying to design a business-oriented schema for blockchain network operation, where the consortium-like nodes can make up a blockchain network but offer a public-like blockchain network services with a uniform standard. The fundamental blockchain network can set up enough incentive to drive the operating nodes focusing on how to improve their operational and service capabilities. Therefore, the business-oriented schema will enable the final business service providers (application developers) and the business service consumers (application users) to use the fundamental blockchain network services easily and conveniently as the today's Internet service.

Keywords: Blockchain · Blockchain applications
Business-oriented schema · Internet operation model

1 Backgrounds and Introductions

The original blockchain proposed in 2008, known as Bitcoin [1], is a completely open operating mode, i.e. the public blockchain model. In the public blockchain model, anyone from all over the world can join or exit the network as a node of the blockchain, freely and easily without any restricts. The participants' identity can be directly transformed between the developer, the provider, and the consumer of the blockchain services without any technical constraints. Therefore, a very important innovation of blockchain is that the blockchain technology can inimitably create a peer-to-peer (P2P) trust platform in such a completely trustless environment without a common centralized and trusted third-party [2]. In consequence, the public blockchain technology has contributed the most effort to the consensus mechanism to achieve the consistency of numerous nodes [3]. For an example, Proof of Work (PoW) powered blockchains currently account for more than 90% of the total market capitalization of existing digital cryptocurrencies. The PoW consensus mechanism requires every node must complete a lot of but might not be a commercial sense of computations in order to achieve the entitlement to record the ledger of the blockchain, e.g. a coin or token reward [4]. With this endogenous incentive mechanism, the blockchain operators (nodes) or so called the blockchain network service provider can be encouraged to stay in the blockchain network to promote the current blockchain network running more robustly and steadily.

While a completely open and decentralized operating mode can theoretically obtain one of the highest levels of trust, it also raises the performance issues of blockchain network, such as the current highest frequency of transactions is under 6 TPS of Bitcoin and about 15 TPS of Ethereum [5] which is much lower than the current centralized payment system. The complete decentralization also brings with the privacy and access control issues as well as particularly the supervision issues which must be faced by social and commercial applications [6]. Now, the way under discussion to solve these problems in the public blockchain, such as trading partitions and data sharing, will inevitably reduce the degree of decentralization.

The consortium blockchain [7] is a later proposed operating mode, which is trying to touch a compromise between trust level and performance requirements. In consortium blockchain model, each node must go through certain permission before joining the blockchain network, so the identities are basically not directly transformable between the blockchain service provider and the blockchain service consumer. The consortium blockchain can, therefore, using a more consistent Byzantine fault-tolerance (BFT) algorithm to solve the Byzantine fault tolerance [8,9] of the distributed ledgers. This will also bring an additional benefit that no bifurcation existed anymore. Nodes of consortium blockchain generally do not have a mechanism to achieve rewards from the endogenous blockchain system but are usually driven by the self's business interests of operators. The biggest problems that the consortium blockchain will face to put the project for actual working, is that most of the existing business scenarios are difficult to find multiple reliable nodes that can reach the consortium blockchain requirement. For the sake of participants' own interests, enterprises or other organizations prefer to control the blockchain network and finally convert the real blockchain network operating model to a private blockchain, which extremely weakens the most important characteristic of the blockchain – trust.

After introducing the smart contract [5], the blockchain network service will have a significant shift for the blockchain network consumer identity which is divided from a single-identity of blockchain network user to blockchain application developers (blockchain application service providers) and blockchain application users (blockchain application service consumer). The blockchain network services will play as an independent operator role in the service architecture of the blockchain application services. In general, the service providers of blockchain networks are different from the service providers of blockchain applications, and most importantly, their goals or incentives would not be the same. However, due to the uniqueness of the blockchain network, the inconsistencies of incentives may hinder the development of the blockchain network itself. For example, the service providers of blockchain networks in the public blockchain, known as the miners, may simply pursue greater token incentives rather than satisfying the needs of blockchain application providers and users. In the consortium blockchain network, the nodes will only support the applications in line with their own commercial interests, rather than to support other external users' applications. The different commercial interests of enterprises will disperse the users' data to different blockchain networks and impede the connection of data.

This paper is an attempt to design a business-oriented schema for blockchain network operation, where the consortium-like nodes can make up a blockchain network but offer a public-like blockchain network services with a uniform standard. The fundamental blockchain network can set up enough incentive to drive the operating nodes focusing on how to improve their operational and service capabilities. Therefore, the business-oriented schema will enable the final business service provider (application developer) and the business service consumer (application user) to use the fundamental blockchain network services easily and conveniently as the today's Internet service. In Sect. 2 we discuss the Internet operating mode and in Sect. 3 we introduce the idea of the Internet operating mode to blockchain network operating and discuss what is the business-oriented schema of the blockchain network service with some independent operators. In Sect. 4, we point out some important features of the business-oriented schema for blockchain network operation.

2 The Schema of Internet Network Operation

The current operation schema of Internet service is the network operator mode. The basic services of the Internet are provided by so-called network operators (most are companies), such as AT&T, China mobile and so on. The hardware and technology upgrade of the Internet is usually provided by network equipment providers, such as Huawei and Cisco. The network equipment providers would upgrade the Internet hardware capabilities under certain technical standards, such as upgrading from coaxial cable to optical fiber and mobile communication from 4G to 5G. However, on the service side of Internet applications, the developers of commercial applications only need to publish their own services via the Internet, and users can then access these services easily through the Internet. For applications, the Internet itself is just a medium, and different network operators or different hardware devices will not have a fundamental impact on the form of application services (Fig. 1).

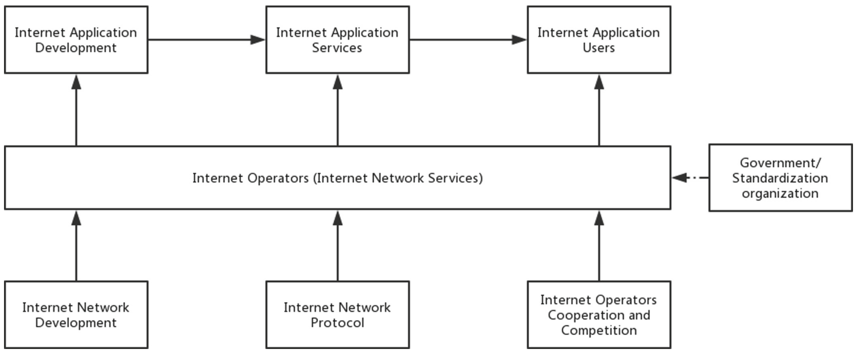

Internet Application Development
Internet Application Services
Internet Application Users
Internet Operators (Internet Network Services)
Government/ Standardization organization
Internet Network Development
Internet Network Protocol
Internet Operators Cooperation and Competition

Fig. 1. The operation model of Internet network

Internet network service providers, i.e. network operators, charge the connected Internet service users (including business application developers and users) according to certain standard rate, which would enhance their ability to keep up the Internet network service. Therefore, as the underlying technical support of the Internet, network equipment vendors and other roles are basically isolated from the service users of the Internet. The advantage of such a layering is that the iteration of Internet technology will not fundamentally affect the development, maintenance and the use of the Internet applications. At the same time, Internet operators must be competitive to improving the service capabilities of the Internet, while application developers can focus on their own business and explore application scenarios and improve the user experience.

Under the existing Internet operation model, different operators can use various technologies to offer gradational and multidimensional levels of Internet access services, such as the Internet access speed. Therefore, the connectivity between different Internet operators often requires two operators or a certain organization to achieve the global Internet connectivity in accordance with some agreements. Internet users that are switching services between different operators may have to pay for cross-operators or even service disruption. That is, where the Internet is open, but often because of the natural monopoly of network operators, developers and users are not completely free to choose their operators.

3 Introducing the Idea to Blockchain Network Operating

The nodes of public blockchain such as Bitcoin are completely open, so both individuals and companies can run blockchain networks equally as long as they can provide the corresponding server supports according to the blockchain agreement. However, as the proof of work (POW) to be more and more difficult, it is almost impossible for an individual's ordinary server to effectively participate in the mining competition, substantially out of the Bitcoin network operation. Thus, a large number of so-called miners whether individuals or companies have an increasingly convergence to form the “mine pools” to participate in the mining competition. For example, the top four mining pools of Bitcoin have now controlled more than 50% calculation power of the entire network. The situation of Ethereum, the second largest crypto-currency, is also similar to that. It can be seen that the public blockchain built up by the proof of work shows that with the increase of the calculation power of the whole network and the difficulty of mining, the aggregation of computing power is unavoidable from the game point of view, thus seriously questioning the blockchain. The most important one is based on the distribution, which has brought great hidden dangers to the unreliable modification and stability of the blockchain network. Using other consensus mechanisms such as POS, DPOS, etc., it is also unavoidable or even active to reduce the number of nodes actually participating in the consensus and writing ledgers, in order to reduce the huge loss of POW and expect to improve the operating performance of blockchain networks. In general, although the public blockchain upholds the spirit of openness and sharing, now and in the future, the number of nodes that can actually participate in the consensus is very limited or even not open to the public.

The consortium blockchain operators are permission as established. All operators must obey the consortium’s rules to run the blockchain network, so node to join and exit is limited by the consortium rules. However, in the real business scenarios, forming a stable consortium is difficult, especially the leader of the consortium is often applied the main stakeholders have absolute power consortium blockchain and makes consortium blockchain substantially degenerate to a private blockchain. This will lose the most important characteristics of building a blockchain – “trust”. Now in the crypto-currency market, a large number of independent operating blockchain network claim that they are consortium blockchain network or even public blockchain network, but in essence, they are just company‘s “private” blockchain just like the above situations.

This is why the Internet network operation model is introduced in the blockchain. Under the schema of blockchain network operation model, the owner of the node which is the network operator should be independent business organization or company. It is like today’s Internet operators, so they don’t need to profit from application developers and users according to the detail business scenarios, but from providing universal blockchain network services. Sequentially different operators will form a new blockchain network relationship with both cooperation and competition, where the operators can also give more credibility and more dynamic blockchain services. When more operators dominated in the consortium blockchain network, the corresponding blockchain network has more competitive between the various blockchain networks. Under the consortium rules, a few nodes will not be able to control the final blockchain network (Fig. 2).

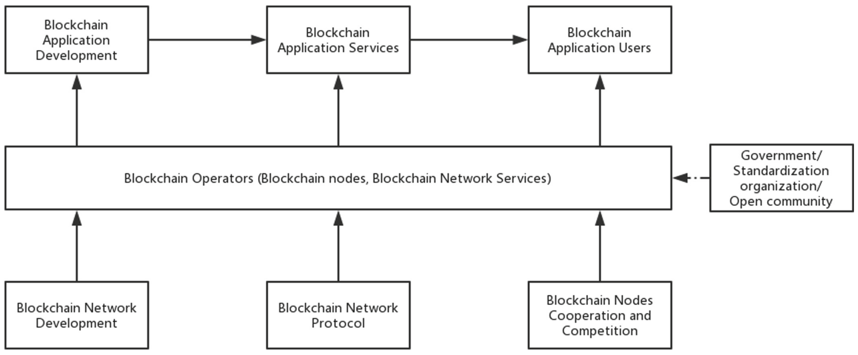

Blockchain Application Development
Blockchain Application Services
Blockchain Application Users
Blockchain Operators (Blockchain nodes, Blockchain Network Services)
Government/ Standardization organization/ Open community
Blockchain Network Development
Blockchain Network Protocol
Blockchain Nodes Cooperation and Competition

Fig. 2. The business-oriented schema for blockchain network operation

4 Important Features of the Business-Oriented Schema for Blockchain Network Operation

When establishing a new business-oriented schema for blockchain network operation, some features are very important and should be paid more attention to.

Separation of Network and Application. For Bitcoin, the application is not separated from the Bitcoin network. The Bitcoin network is just the Bitcoin application. After the introduction of smart contract concept, the basic network of the blockchain can be run independently, while the blockchain application can run in the form of a smart contract or a combination of several smart contracts on the blockchain network. Although the smart contract has already been implemented in the blockchain, the maintenance of network operation and application is still not enough separated. For example, ETC is used to stimulate the nodes in the Ethereum network. However ETC is also spent for the calculation and storage cost of the applications on the Ethereum network. The coupling of network and application results in that the network operators may not benefit the application from their own interests and harm the network overall interests. Therefore, in the schema of blockchain network operation model, the blockchain network should be independent of the application development and services. In particular, application development should only follow a unified protocol and some common underlying interfaces, which can make it easily to be migrated between different blockchain networks.

Separation of Network Development and Operation. Under the schema of blockchain network operation model, the network development should be separated from the network operations. Technology developers will focus on the development and planning of block chain technology. However, the network operators have the complete right to decide on which new blockchain technology to adopt as long as it follows the blockchain protocol, standard and consensus mechanism of the consortium blockchain. The technical differences between nodes bring some differentiation of services into blockchain network users, such as in guarantee under the premise of using the same blockchain network, some nodes can provide users with faster query service, while some nodes can provide users with more storage space outside the blockchain and so on. While running on the same blockchain network, different nodes can still have different target users, so they can offer personalized development tools and testing environments. This is a useful feature for implementing a general-purpose, efficient blockchain network.

Incentives for Different Roles at Multi-levels. For a blockchain system, the incentive is the most important means of governance. The public blockchain often has only a single incentive mode, namely the incentive of the coin or token to the blockchain node. After introducing the schema of blockchain network operation model, the incentive for different roles can be layered. For the blockchain network technique developers, they must provide better and more powerful blockchain network technology so as to be adopted by the blockchain network operators. This is one of the most important incentives for technique developers, unlikely from today's blockchain where the technology developers are often also the blockchain operators. Overlapping of the two roles usually tends to be bad for the entire network ecosystem and harms the progress and development. For the blockchain operators, the most important incentive is to provide stable and continuous blockchain services, thus the numerous blockchain operators must maintain a dynamic and effective blockchain service with a cooperation and competition relation. Although for a blockchain network, the network underlying protocol and the data are the same, but the operators can still provide differentiated technology and service ability. At the same time, a blockchain node can offer multiple blockchain network connectivity; so implementing different blockchain networks are also favorable for the formation of blockchain global ecological system. For blockchain application developers, just like the Internet application developers do not need too much concerned with the actual operators, they can focus to provide more valuable, professional application on the blockchain services. On the basis of the characteristics of blockchain, the application server is entirely operated by blockchain network i.e. operator is responsible for the continuing and tamper-resistant blockchain service after the application released. The interests of blockchain application users can be protected effectively based on the smart contracts which are written in the clear and open source programming.

After the establishment of a complete data system [10], the blockchain can be used as a very suitable big data platform. The data value of the blockchain system can be continuously mined by using a Big Data Open Architecture [12] and other big data techniques [11,13].

5 Summary and Conclusion

In this paper, we propose the schema of blockchain network operation model based on the existing Internet operation mode. The consortium-like nodes can also make up a blockchain network but offer a public-like blockchain network services with a uniform standard. The fundamental blockchain network can set up enough incentive to drive the operating nodes focusing on how to improve their operational and service capabilities. It also drives the blockchain network technique developers focusing on blockchain technique, while the blockchain application developers focusing on valuable and professional applications.

Acknowledgement. The author would like to gratefully acknowledge Dr. Huan Chen and Dr. Cheng Li for the useful discussions and helps. This work is partially supported by the technical projects No. 2017YFB0802703, and No. JSGG20160331101809920. This work is also supported by NSFC (91646202) and the National Hig-tech R&D Program of China (SS2015AA020102).

References

1. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system (2008). https://bitcoin.org/bitcoin.pdf
2. Beck, R., et al.: Blockchain-the gateway to trust-free cryptographic transactions. In: ECIS (2016)
3. Eyal, I., et al.: Bitcoin-NG: a scalable blockchain protocol. In: NSDI (2016)
4. Gervais, A., et al.: On the security and performance of proof of work blockchains. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. ACM (2016)
5. Buterin, V.: A next-generation smart contract and decentralized application platform. White paper (2014). https://github.com/ethereum/wiki/wiki/White-Paper
6. Zyskind, G., Nathan, O.: Decentralizing privacy: using blockchain to protect personal data. In: Security and Privacy Workshops (SPW). IEEE (2015)
7. Zheng, Z., et al.: An overview of blockchain technology: architecture, consensus, and future trends. In: 2017 IEEE International Congress on Big Data (2017)
8. Castro, M., Liskov, B.: Byzantine fault tolerance. U.S. Patent No. 6,671,821, 30 December 2003
9. Castro, M., Liskov, B.: Practical Byzantine fault tolerance. In: OSDI, vol. 99 (1999)
10. Zheng, Z., et al.: Blockchain challenges and opportunities: a survey. Work Paper 2016 (2016)
11. Zhang, L.-J., Zeng, J.: 5C, a new model of defining big data. Int. J. Big Data (IJBD) **2**(4), 10–23 (2015). https://doi.org/10.29268/stbd.2015.2.4.2
12. Zhang, L.-J., Chen, H.: BDOA big data open architecture. Int. J. Big Data (IJBD) **2**(4), 24–48 (2015). https://doi.org/10.29268/stbd.2015.2.4.3
13. Zhang, L.-J.: Data value chain and service ecosystem - a way to achieve service computing supporting 'Internet+'. Int. J. Big Data (IJBD) **2**(4), 49–56 (2015). https://doi.org/10.29268/stbd.2015.2.4.4

Your Device and Your Power, My Bitcoin

Song Li[✉] and Scott Wu

NewSky Security Solution Inc., 4018 148th Avenue NE,
Redmond, WA 98052, USA
LS@newskysecurity.com

Abstract. The dramatic increase of cryptocurrency price has reshaped the cyber security landscape. Many IoT malwares are created, turning many kinds of infected IoT devices into crypto currency mining machines. Crypto currency miner machines, dedicated IoT devices that are built for the sole purpose to mine cryptocurrencies, are also becoming hacker's new target. This paper describes how attackers are exploiting both the horizontal and vertical of aspects of IoT devices to mine cryptocurrency and proposed a method to evaluate the cost of such exploits.

Keywords: Cryptocurrency · Malware · Mining · Power consumption

1 Introduction

The rise of cryptocurrency opens a new way for cyber attackers to profit. Attackers first built ransomware, a type of malware that encrypts victim's data. To decrypt their data, victim needs to pay cryptocurrency to obtain the private key. In the past year attackers built new malware that directly converts infected devices into cryptocurrency mining machines. Recently attackers are turning toward a new type of IoT devices, cryptocurrency miner machines, computers that are built for the sole purpose of mining cryptocurrencies.

Here at NewSky Security, we analyzed both cryptocurrency mining malware, and the firmware of popular miner machines, to find out how a device can be infected and mining for attackers. The goal is to find out ways to protect devices from being controlled by attackers pursuing cryptocurrencies. We also proposed a way to evaluate the power lose caused by mining malwares, by using miner machine as a baseline.

The rest of this paper is organized as follows: using Bitcoin as example, the second chapter gives a brief introduction to cryptocurrency, highlighting features in its algorithm that makes any IoT device a mining machine. The third chapter explains how mining malware exploits the horizontal IoT world, using as many machines as possible. The fourth chapter describes how a miner machine works, and vulnerabilities a miner machine exposes to attackers. In the fifth chapter we propose a quantitative way to evaluate the power loss mining malware causes, using miner machine as a baseline. The sixth chapter concludes this paper.

2 Cryptocurrency and the Competition of Computing Power

Since the first appearance in 2009, first as a whitepaper with author's alias Satoshi Nakamoto [1], Bitcoin defines a new way to compute ownership in the digital world. Machines work together to reach a consensus, ownership, and more importantly, transitions of ownership, is recorded on all participating machines, every machine has a copy of the ownership transition, hence the name "Distributed Ledger".

The ledger is designed to use the hash of the latest transaction and the previous records, making it easy to verify and extremely hard to forge. Each transaction is called a Block, pointing to previous transaction, forming a chain of transaction history, hence the name "Block Chain". All participating machines agree to follow the longest chain, to prevent discrepancies among machines, such as double-spending.

Cryptocurrencies rely on participating machines, or nodes, to keep track of all ownership transactions. To encourage nodes to participate, different cryptocurrency system provides different incentives. Bitcoin, for example, will reward one of the nodes with the ownership of a Bitcoin. To win this ownership, nodes need to compute a problem, and the node that came up with a result that is closest to the goal will win the ownership. This problem, naturally, is calculating hash. The massive calculation of hash makes it harder to tamper a chain of blocks.

By contributing the computing power to calculate hashes, nodes show their proof of work, or PoW. In a PoW system, the more hash one node can generate in a given time, the higher the chance this node can find the best answer and be awarded with the new ownership, in other words, nodes with more computing power have better chance to be awarded. Other cryptocurrency systems use Proof of Stake PoS to decide which node is awarded with the new ownership, where nodes with more ownership have higher chance to be awarded with the new ownership.

Since Bitcoin is the first and most popular cryptocurrency system that is based on PoW, mining for Bitcoin, and other PoW cryptocurrencies, is a war of competing for computing powers. Mining machines evolves from regular computers, to servers in the cloud, to computers with GPUs, and eventually to special purpose machines equipped with ASIC chips whose sole purpose is to compute hash.

When people realize they are essentially competing for the percentage of computing power in the mining world, they unite and work together, and split the profit. This effort once reached a point it threated the foundation of Bitcoin. Bitcoin is built on the assumption that nodes are decentralized, and no more than half of the nodes will collaborate. If more than half of the nodes collaborate, they can work together and alter the ledger. The worst case, they can claim all ownership. It was believed back in 2014 that the collective computing power of mining facilities once went beyond 50% of the entire computing power of Bitcoin community [2].

3 The Rise of Mining Malware

Note that the basic operation of cryptocurrency is mining, or computing hashes, computing a single hash does not require much computing power. The PoW arrangement does not enforce data dependency among calculations, making it suitable for a highly parallel computing scheme – any computing device that can communicate its result to the network can participate. This observation made mining feasible horizontally, across all devices, from super computers to IoT devices operating on battery power. If a device is less protected, it becomes an easy target of malware. If there are many of the same kind of vulnerable devices, it is possible to infect them and create a virtual mining farm. The same observation also made mining more efficiently vertically – building a machine that is composed of thousands of single-purpose computing device and dedicated for mining. We will cover this in the next chapter.

A typical mining malware's behavior is like a botnet – it first tries to gain access to an IoT device, either through trying weak common passwords, or through CVE or 0 day vulnerability. Once it can create a process running on the target device, it converts the device into a mining machine, receiving instructions from Command-and-Control server and calculate hash values using the device's CPU.

In February 2018, a Czech Republic cybersecurity company reported 15,000 connected devices could be hacked to mine cryptocurrency worth of $1,000 in just 4 days [3].

By running on other people's device, mining malware calculate and generate computing power that help attackers profit on cryptocurrencies.

On March 26th 2018 Brian Krebs' website shows CoinHive site has malicious code to turn visitor's device into mining machines [4].

Fortinet reported new mining malwares are using Javascript to convert a browser visiting a website into a mining machine [5].

4 The Mining Economy: Miner Machines, Mining Facilities and Pools

In previous chapter, we described the basic calculation in cryptocurrency, is calculating hashes, and the calculation can be highly parallel, making it feasible for many computing units, though weak by itself, to create enormous computing power when combined. We described how attackers use mining malware to collect computing power from large number of devices and generate cryptocurrency. In this chapter, we look at how those simple computing units, when packed into a single machine, can also generate intense computing power, and naturally, make themselves targets of attackers.

4.1 Miner Machines

Miner machines used to be a computer with cryptocurrency mining software installed, talking to the rest of the nodes in a cryptocurrency system. As owners of the machines compete for computing power, they start to use workstations, servers and instances in clouds. Aside from CPUs, people soon find out special processing chips, such as GPUs and FPGAs can handle large number of parallel hashing tasks and start running miner machines with GPUs and FPGAs. Eventually people found Application Specific ICs, or ASIC, offer the best Performance/Price ratio.

This is not the first time we see ASICs, specialized processors and CPUs compete for the same killer application – when MPEG4 was introduced, ASIC was the first to offer acceptable performance for MPEG decoding, thanks to Moore's law, special purpose processors such as DSPs caught up, and eventually general-purpose CPUs offer the same performance while handing other tasks such as sending emails at the same time. This time for cryptocurrencies, the fast-growing price of coins, and other factors such as the simplicity of calculation, and energy consumptions, made ASIC the best choice over GPU, FPGA and CPU.

When this paper is being written, BitMain, one of the major ASIC mining chip and miner machine providers, claims their AntMiner^{TM} S9 can run 14 Terra Hashes per second. In this paper we will use BitMain's S9 as a baseline for performance and power consumption. We made this choice because BitMain's firmware is publicly available for download, and we bought one AntMiner machine for research purpose.

Unlike individuals who mine with several miners, mining companies combine miner machines into containers, this helps reduce cost of shipping and management.

4.2 Mining Facilities

Like any high-performance computing facility, the main cost of a miner machine comes from the energy it consumes. With today's rate at which Bitcoins are mined, it is believed that when the Bitcoin price is below $8,000 mining companies will not be able to make profit, and most of the cost is energy cost. Like data centers, mining companies constantly seek for low cost electricity, and such locations with low daily temperature is preferred, as it saves cooling cost and reduces the chance a miner machine is overheat (Fig. 1).

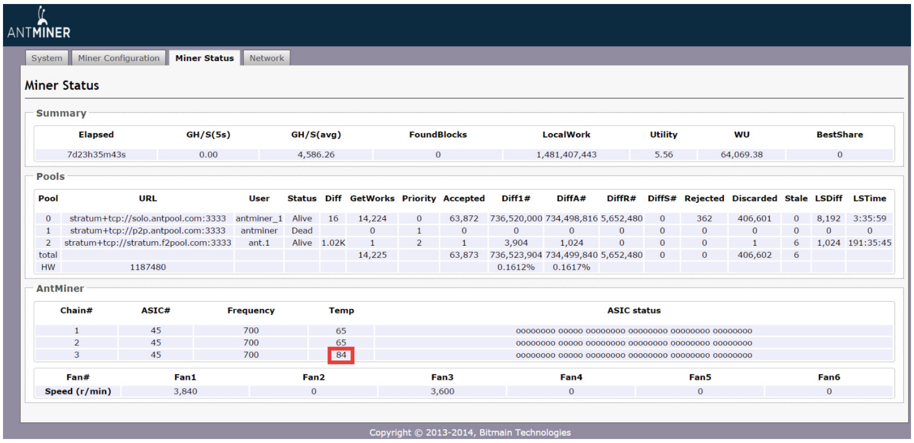
Fig. 1. A mining facility. Image credit: https://www.politico.com

4.3 Pools

When miner machines are installed at the mining facilities, they need to be powered on and connect to the internet, typically they are connected to one or more websites that are called Mining Pools or just Pools. A pool is a website that collects the computing power of miner machines and powers the cryptocurrency computation. A pool will issue tasks to miner machines, the job is then hashed on the miner machine, and results are sent back to the pool for evaluation. If a miner's result is the best answer among all computed hashes in the entire Bitcoin system, a bitcoin is awarded to the pool. When a coin is issued to the pool, it is typically split among all members of the pool, according to their share of computing power contributed. The abundance of computing power, along with coins that reside in the pool's database, made them natural targets for attackers.

4.4 Attacking the Miner Machines and Pools

When a miner machine is powered up, it needs to be configured to communicate with the pool. The owner of the miner enters his account information to the miner, when a coin is awarded, the owner's share will be deposited into her account (Fig. 2).



Miner Status

Summary

Elapsed	GH/S(5s)	GH/S(avg)	FoundBlocks	LocalWork	Utility	WU	BestShare
7d23h35m43s	0.00	4,586.26	0	1,481,407,443	5.56	64,069.38	0

Pools

Pool	URL	User	Status	Diff	GetWorks	Priority	Accepted	Diff1#	DiffA#	DiffR#	DiffS#	Rejected	Discarded	Stale	LSDiff	LSTime
0	stratum+tcp://solo.antpool.com:3333	antminer_1	Alive	16	14,224	0	63,872	736,520,000	734,498,816	5,652,480	0	362	406,601	0	8,192	3:35:59
1	stratum+tcp://p2p.antpool.com:3333	antminer	Dead		0	1	0	0	0	0	0	0	0	0	0	0
2	stratum+tcp://stratum.f2pool.com:3333	ant.1	Alive	1.02K	1	2	1	3,904	1,024	0	0	0	1	6	1,024	191:35:45
total					14,225		63,873	736,523,904	734,499,840	5,652,480	0	0	406,602	6		
HW	1187480							0.1612%	0.1617%							

AntMiner

Chain#	ASIC#	Frequency	Temp	ASIC status
1	45	700	65	oooooooo ooooo oooooooo oooooooo oooooooo oooooooo
2	45	700	65	oooooooo ooooo oooooooo oooooooo oooooooo oooooooo
3	45	700	$\boxed{84}$	oooooooo ooooo oooooooo oooooooo oooooooo oooooooo

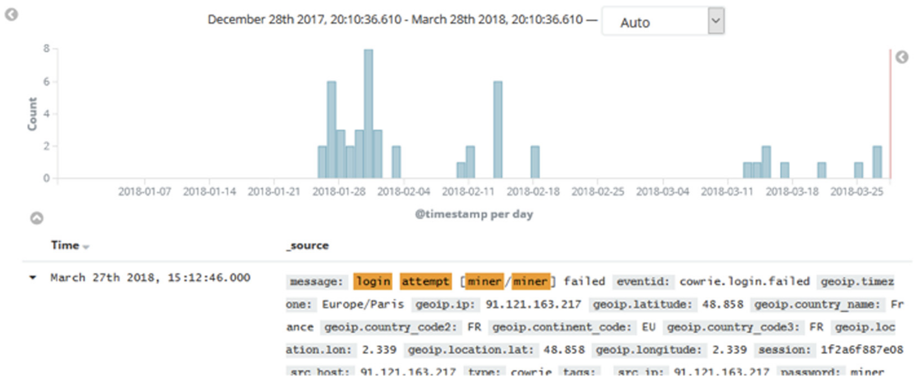
Fan#	Fan1	Fan2	Fan3	Fan4	Fan5	Fan6
Speed (r/min)	3,840	0	3,600	0	0	0

Copyright © 2013-2014, Bitmain Technologies

Fig. 2. Miner's configuration page. Image credit: http://support.bitmain.com

Naturally, the pool site, where every owner's account information is stored, is one of the most attractive targets in the cryptocurrency cybercrime economy. Once the pool is compromised, miner owner's information, along with their accounts, will be at danger.

Besides attacking pool site servers, if a miner machine is directly connected to the internet, its configuration web interface can be attacked, and the attacker can use his account to replace the owner's account. Another attack approach, is to build an attacker's pool, and reconfigure the compromised miner machine to work for the attacker directly, powered by the owner's electricity supply. Hacking miner machines is now automated, attackers are using automated tools to scan the internet, looking for miner machines that are exposed and trying different ways to gain control of the miners. Figure 3 shows the spike of Mining OS hack attempts captured by NewSky honeypot.



```
Time              _source
March 27th 2018, 15:12:46.000   message: login attempt [miner/miner] failed eventid: cowrie.login.failed geoip.timez
                                one: Europe/Paris geoip.ip: 91.121.163.217 geoip.latitude: 48.858 geoip.country_name: Fr
                                ance geoip.country_code2: FR geoip.continent_code: EU geoip.country_code3: FR geoip.loc
                                ation.lon: 2.339 geoip.location.lat: 48.858 geoip.longitude: 2.339 session: 1f2a6f887e08
                                src_host: 91.121.163.217 type: cowrie tags: src_ip: 91.121.163.217 password: miner
```

Fig. 3. NewSky honeypot showing spike of Miner hacking attempts. Image credit: https://newskysecurity.com

5 Evaluating the Loss Caused by Mining Malware Botnet

Prior to mining malware, the loss of a cybercrime incidence is often evaluated against the damage of the software, hardware or business value. For example, in a DDOS attack, the loss is estimated on the downtime of the site affected, multiplied by the site's average revenue per second. None of the existing methods can be directly applied to estimate the loss caused by mining malware. Mining malware profit by draining the extra processor and network bandwidth from the infected device. To some extent, an infected device runs without interruption benefits both the owner and the attacker.

We propose a new mechanism to evaluate the loss caused by mining malware. Instead of estimating the data value or business value, we estimate the power consumption of a mining malware. As we pointed out in previous chapters, the major cost factor of a mining facility is its power bills. While mining malware try to stay under the radar and only consumes excessive processor and network bandwidth, the extra power consumption it introduces adds burden to the device owner or service provider.

To evaluate the power consumption, we argue that each mining machine, either a dedicated miner, or an infected device, is essentially a connected machine that is capable of manipulating bits. Since the mining operation is essentially calculating hashes, we can model the simplest form of a mining machine, is a connected computing device that takes in a vector of bits, after some time and consuming some energy, returns another vector of bits. We call this simplest form of mining machine a Communication Fabric, where Communitation is a combination of Communication and Computation, describing both the connecting and computing aspects of an IoT device.

The efficiency of a Communitation Fabric is evaluated as Bits/(Second * Joule), where Bits is the number of bits produced, at the cost of Second and Joule.

We argue that each mining machine, either specially built, or infected and recruited by mining malware, is essentially a collection of Communitation Fabric. A miner machine is a collection of well-designed, uniformly high efficiency fabrics, which is a vertical of mining IoT devices, while a botnet of mining devices is a collection of different kinds of Communitation Fabrics, spreading horizontally across the IoT world. Driven by the energy bill, the miner machine's Communitation Fabric is the most energy efficient, and costs less since mining facilities are often using the cheapest energy that can be found. Given two set of hashes, each has the same number of hashes, Sh and Sv, where Sh is calculated using horizontal fabrics such as a mining botnet, and Sv is mined using dedicated miner machines, the energy cost of Sv will not exceed Sh.

Using the latest BitMain AntMiner^{TM} S9 as example, according to S9 manual [6], one S9 miner's peak performance is 14Th/S and consuming power at 1323 W. Assuming this S9 is hashing for Bitcoin network, which hands out hash work at 640 bits. For each second a S9 handles 14T * 640 bits input for hashes, consuming 1323 Joules. Its efficiency is 6.77TBits/(Second * Joule). For a mining malware botnet to generate the same rate of hashing, it needs to consume at least 1323 W of power. This is the baseline of power consumption of a mining botnet. Assuming the Bitcoin rewarding system is fair, meaning each hash value calculated has the same chance to be the best result, a mining botnet will have the same rate of accumulating Bitcoin as a miner machine with equal computing power. Bitcoin wallets are tracked online, and its Bitcoin accumulation rate is publicly available. By monitoring the wallet of mining malware's author(s), it is possible to evaluate how many Bitcoins the malware botnet has calculated and estimate how much power the botnet has consumed by comparing the gain rate of the botnet to a S9 owner.

6 Conclusion

In this paper we described the ever-changing landscape of cryptocurrency malware, and how attackers are using all kinds of IoT devices to harvest cryptocurrency, as well as directly attacking mining facilities and steal cryptocurrency. We described the significant difference of cryptocurrency attackers from traditional cyber attackers, as cryptocurrency attackers need the infected system to run consistently, so as to drain more CPU and network bandwidth for mining. At the end of the paper we proposed a new way to evaluate the damage caused by cryptocurrency mining botnet.

References

1. https://bitcoin.org/bitcoin.pdf
2. https://www.extremetech.com/extreme/184427-one-bitcoin-group-now-controls-51-of-total-mining-power-threatening-entire-currencys-safety
3. https://www.cnbc.com/2018/03/01/thousands-of-iot-devices-can-be-hacked-to-mine-cryptocurrency-avast.html

4. https://krebsonsecurity.com/2018/03/who-and-what-is-coinhive/
5. https://www.fortinet.com/blog/threat-research/the-growing-trend-of-coin-miner-javascript-infection.html
6. https://file.bitmain.com/shop-bitmain/download/AntMiner%20S9%20Installation%20Guide.pdf

Blockchain in Global Trade

Jack Duan[1] and Milan Patel[2(✉)]

[1] Gliding Eagle Inc., San Ramon, CA 94583, USA
jduan@gliding-eagle.com
[2] IBM, Durham, NC 27603, USA
mspatel@us.ibm.com

Abstract. This paper summarizes the actual adaptation of the Blockchain (BC) technology in the global trade space. Each product with a unique identification is tracked from the source manufacturer to the end user across countries. Data is collected from manufacturers, logistics providers and end users. A third-party company serves as the BC network operator so that each transaction is recorded in a private BC network with periodical hash saved onto a public BC network to ensure data immutability. This paper further explores the use of BC to manage the owner and users of data sources. There are two business applications for this system: (1) track and deliver using direct-to-consumer model for over 100 premium California wineries to over 20 countries in the world; (2) track and deliver using direct-to-hospital model for specialty pharmaceuticals (cancer drugs, etc.) from the US to designated hospitals in China.

Keywords: Blockchain · Global trade · Supply chain · Self-sovereign identity Decentralized identity

1 Global Trade

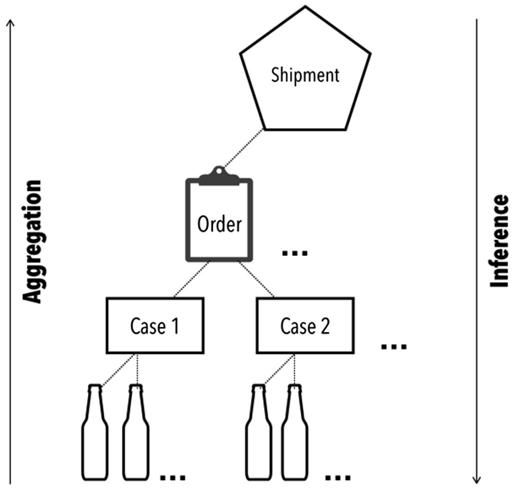
1.1 Pre-identification and Tracking

In the origination country, each product item once manufactured is labeled with a unique identification number. Traditionally each product is identified with an industry classified UPC code, but this new level of identification makes each product uniquely marked. For example, if there are 12 bottles of the same vintage wine, traditionally they will share the same UPC code, but with per-item tracking, there are 12 unique codes, with each assigned to every bottle. These identification codes will be saved in the BC network.

1.2 Aggregation and Inference

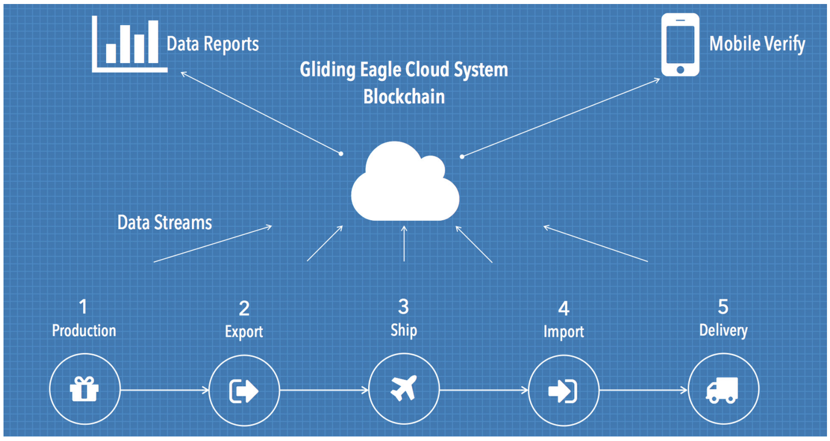
Using the same example above for 12 bottles of wines. They will be aggregated into a case, which also has a unique identification code. This case will be linked to an order. And an order (or orders) will be further aggregated at the shipment level. Once a shipment is being transported from one party (e.g. manufacturer) to another party (e.g. shipper), an actual transaction activity occurs. If the links are preserved, we can infer that all connected orders, cases and bottles will have the same transaction activity recorded.

S. Chen et al. (Eds.): ICBC 2018, LNCS 10974, pp. 293–296, 2018.
https://doi.org/10.1007/978-3-319-94478-4_23



1.3 Data Collection from Multiple Parties in Global Supply Chain

Initially the BC network is a private one; with all participants defer peer manageability to a mutually trusted third-party (i.e. Gliding Eagle) as the network operator. This is due to the initial high cost for each party (manufacturer, shipper and end user) to host their own peers. With increased data volume in the future, many parties have the option to fully host and manage their own peers to replicate data. Each party uploads data to the private BC network hosted in the global cloud infrastructure using well-defined API calls. These transactional data can be recorded at any aggregation/inference levels, i.e., bottle, case, order, or shipment. The entire data set consists aggregation links, data source and owner identity, and transactional activity data over time.



1.4 Data Reporting

Data once recorded in BC network in ledger format and in traditional SQL structured format, two applications are used to manage and report data to relevant parties. For manufacturers and other participants (logistics providers, exporter and importers, retailers, etc.) a desktop and mobile-based reporting application can display data based on proper access authorizations. The network operator manages all data access permissions based on participants' consensus. For the end user, a subset of accessible data will be displayed on a mobile application. For example, when the end consumer uses a mobile phone to scan the unique label on the wine bottle, the product information along with most essential logistical steps in global trade supply chain is displayed pertinent to that particular bottle.

1.5 Identity Management

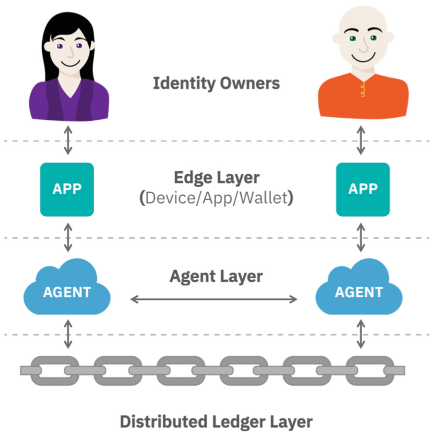
Today, each participant within a BC network requires trust with other participants to do business with confidence. That trust today is based on knowing the participants, either through previous interactions or through a scaled down "web of trust".

As BC networks look to scale to open the aperture of participation in business networks beyond participants who are already "known", deficiencies are created in having to perform "Know your customer/participant" to onboard other potential participants onto the network that are not previously known to existing participants. Certain attestations will be required from trusted institutions, such as governments, banks, and more to prove participants are who they say they are.

In global trade, establishing trust among participating parties is paramount to facilitate high efficiency in both product logistics flow and money transfers. These participants such as manufacturers, exporters, transporters, retailers, government agencies and end users will benefit from the higher efficiency in trade, which means lower transactional cost economically.

As the spectrum is broadened, in a world with many BC networks that are built on use case specificity, each network will be considered a "mini city" with specific network defined governance and policies. The question then becomes, not only, how do participants of BC networks identity themselves within a network but also how do participants identity other participants across different BC networks. For example, one manufacturer can belong to a number of different BC networks as different trade networks where different lines of business are conducted. Identity could be shared among different BC networks, predicated on standards and interoperability rooted with trust and non-correlation.

BC is also accelerating the evolution of identity. Identity is entrenched in every interaction. Enablers such as BC have now made decentralized, self-sovereign identity a reality. It can also accelerate in rebooting the web of trust at scale and trust is not compromised. As every identity owner creates their own identity and permissioning of keys, blockchain provides an immutable, distributed ledger allowing the identity of people, organizations, and things to be resolved, ensuring trust in direct, point to point relationships.



Identification is global trade can leverage self-sovereign, decentralized identity to ensure more trust and control over how information is exchanged in a more secured manner. For example, manufacturers can trust shippers to decrease fraud and time in the shipment of wine. Retailers and buyers can transact with more transparently knowing who they are doing business with, in point to point relationships. The identities can be shared between different BC networks, along with their established trust as an asset to decrease friction in conducting business in global trade.

2 Conclusion

Blockchain as a distributed peer based network with immutable ledger based data storage is useful for transactional data for global trade. This has been shown for premium American wines and specialty pharmaceuticals (both are highly regulated products) use from the US to other countries. The unique direct-to-user business model simplifies the traditional export and import business models. With all data saved in BC network it benefits all participates with much trusted channel accountability, product authenticity, and logistical efficiency. With further BC based identity management features becoming mature, exchange of identity in global transactions will be more secured and under the control of identity owners.

Author Index