Towards Blockchain-based Auditing of Data Exchanges*

Xiaohu Zhou, Antonio Nehme, Vitor Jesus, Yonghao Wang, Mark Josephs, Khaled Mahbub

Birmingham City University,
School of Computing and Digital Technology, Birmingham B4 7XG, UK
Xiaohu.zhou@mail.bcu.ac.uk
{antonio.nehme, vitor.jesus, yonghao.wang, mark.josephs,
Khaled.Mahbub}@bcu.ac.uk

Abstract. Auditing operations in multi-party data exchange, and over an arbitrary topology, is a common requirement yet still an open problem especially in the case where no trust on any participating party can be presumed. The challenges range from storage of the audit trail to tampering and collusion of participating entities. In this paper, we propose a blockchain-based auditing scheme. It is designed based on public key infrastructure and Shamir secret sharing scheme.

Keywords: Blockchain, Data Share, Auditable, Smart-contracts.

1 Introduction

Controlling how sensitive data is shared is an open problem with no complete solution in sight. Beyond the impact of the loss of data itself, it also brings a sharp negative impact on the public's trust and discourage them to engage with electronic systems or share their data [1]. Auditing of workflow is thus a key element when handling data flows. Considering a simple supply chain scenario in Fig. 1, which involves a customer (C), a sales company (S), a manufacturer (M), and a logistics organization (L). When C places an order with S, S receives the order and then sends the product requirements to M. M produces the goods after the received requirements and asks L to deliver the product to C within the agreed upon timeframe. Then C receives the goods from L. Let's assume C is not satisfied with the product due to a defect and needs to return it. The key issue is which entity is responsible for this error. L may be responsible for the fault because of a failure in handling the package or M may have given a defective product to the delivery company. If no companies admit the error, and all parties produce their own internal records showing no fault, one can only assume some of them intentionally modified the existing records in their system to prevent truthful auditing. Having a robust audit system with immutable audit trail is vital to assure non-repudiation and assign accountability for malpractice [2].

* Vitor Jesus is the corresponding author.

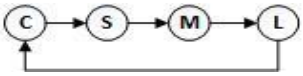

Fig. 1. An example scenario

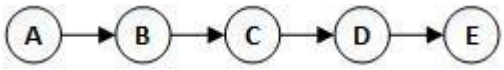

Fig. 2. A representation of a data exchange workflow

Blockchain is a decentralized distributed ledger that contains an ordered list of records in a chain [3]. It is a promising innovation technique given its intrinsic distribution and immutability properties having found application in both financial and non-financial areas [4] [5], such as government public management [5], healthcare industry [6] [7], and privacy preserving in data sharing networks [8] [9]. Blockchain also enables a peer-to-peer transactions without intermediaries or trust relationship agreement.

In this paper, we propose a novel scheme to construct tamper-resisted audit trails by leveraging the blockchain technology. We also provide a theoretical support of data exchange in the confidentiality and accountability. In the reminder of this paper, Section II reviews related work and section III formulates our problem. Section IV proposes our approach and implementation is discussed in section V. Section VI concludes our paper.

2 Related Work

Research in blockchain is covered in diverse domains most notably aspects of traceability and immutability, such as auditing workflow in government processes [10], enterprise business [11], and healthcare data exchange [12][13]. For government applications, permissionless blockchain is not considered to be suitable for government audit systems due to the difficulty of verifying user identities and enforcing strict data governance [10].

Some prior work give a literature review of blockchain technology in auditing environment [14][10], which provided a theoretical support without empirical practice. Blockchain technology provides a solution to automate mechanism for trust without intermediary [10], such as any central authorities. It can also be used to minimize fraud, optimize the existing procedures, and reduce workloads of auditors [14]. However, those papers have not mentioned more details on how to integrate the blockchain technology with the existing auditing processes.

In the prototype design, many previous researches have involved in the proof of concepts development with blockchain. To audit transactions in the data exchange workflow, Ahmad et al. propose a system that records distributed and immutable logs in the Hyperledger blockchain against the external and internal attacks [11]. The transparent logs are stored in the public blockchain without access restrictions. Therefore, this system is not suitable for credential authorities or institutions that require secrecy. Pourmajidi et al. [15] propose an approach based on the super-blockchain and circled blockchain to record and receive logs. Individuals can access logs through some APIs to the immutable hierarchical ledger. The key issue is that this scheme may increase the time to retrieve logs because of the multiple-hierarchical structure of blocks storage. An evaluation is required to verify the impact their proposal on per-

formance. Suzuki et al. design a prototype system based upon the test environment of Bitcoin [16], which is to use blockchain to construct audit logs for strictly access controlled in client-server communication channel. It cannot solve the high-energy consumption as well as the latency in system implementation caused by the mining process, although there is compensated through coin returns.

3 Problem Statement

This section formulates the problems that we tackle, presents a threat model, and lists the designed goals.

3.1 Problem Definition

To illustrate, we use a linear topology - see Fig. 2. Nodes (A, B - E) represent the involved organisations or individuals that they are objects to transmit data. The arrow represents the direction of data flow. The processes of data flow and the related entities are pre-established, which means the interaction between workflow participants are pre-defined. When A is the information sender, who wants to send information to B. A knows the receiver is B and B knows the sender is A. If an outside attacker plants a forged data instead of the payload that B sent to C, we need to ensure that the honest node C can detect this action. If B colludes with D that they tamper with the existing audit information and repudiate performed actions to avoid incrimination during inspection, there should be enough evidence to make other honest nodes spot the incorrect data. If a confidential data is exfiltrated, it is necessary to ensure that the data is encrypted and exposed minimal information.This paper focuses on the level of security improvement in aspects of the accountability of data exchange and transactions reflecting performed actions of the involved participants. We propose a blockchain-based smart auditable check scheme to solve problems that mentioned above.

3.2 Threats Model and Assumptions

In this section, we present out threat model and security assumptions. The audit server includes codes of a smart-contract run on the blockchain that is trusted to perform the protocol, which stores audit records and conducts the verification triggered by the workflow participants. The workflow participants are trusted but some of them may collude with others to intentionally deny their mischievous actions or modify the existing information in the storage after the fact. The outside attackers can eavesdrop on message from the transmission channel and plant forged message instead of the true one in the workflow. Any of participants in the workflow can collude with others to repudiate the performed actions. Therefore, we propose a scheme that is based upon assumptions as below:

Assumption 1: the blockchain is deemed as trusted to immutable store data.

Assumption 2: the workflow participants do not intentionally expose their private keys.

Assumption 3: there is at least one honest participant in every workflow.

3.3 Designed Goals

We design our proposed scheme to satisfy the following goals:

- Confidentiality and integrity. All workflow participants cannot forge or tamper the existing information after-the-fact. Only the data owner can generate correct encrypted audit logs. The nodes of blockchain and workflow cannot forge or tamper the audit logs even if they are dishonest individuals or collude with others. Besides, the audit logs are only stored and verified in cipher form. They cannot be exposed intentionally in a plaintext form. In other words, they cannot be viewed or modified in an undetected or unauthorized way. What's more, the audit server is only store the related encrypted audit logs and keys.
- Availability. Participants cannot escape the audit processes when they require a service. All encrypted audit logs are tamper-resistant and stored in the blockchain. The honest node can access the audit trail to verify the received data.

The above security aspects help to achieve accountability assurance that enabled by having reliable evidence. Our security model renders our approach suitable for applications in which the confidentiality of digital evidence is a requirement. We also aim to assure the availability and integrity of audit trails.

4 Proposed Approach

Our proposed scheme relies on public key cryptography (PKI), a group of signatures, records verification, and Shamir secret sharing scheme. PKI is used to encrypt exchanged messages which improves the confidentiality of workflow. Shamir secret scheme has a positive impact on the protection of encrypted data (in our case is the audit trail). It is theoretically not feasible to decrypt the audit records with one split of the key [17]. The usage of a group of signatures is to mark each action that ensures the data integrity. Audit records verification is an important component, which enables participants to check the correctness of audit records equivalent to a transaction that was received. In this section, we introduce the description of notation, system architecture, the related protocol, and key management.

4.1 Notation

For easier of description and reference, symbols used in the proposed scheme are summarized as below. The keyGen is an abbreviation of key generation.

Table 1. A table of notation description

Symbols	Descriptions
N	$N \in n$, $n = \{1 \dots m\}$. n is a set of nodes that includes many (m) nodes in a workflow, N presents one node
WK, WSK	A public and private keys of each workflow
K_n	A split Shamir key of WSK for a single node
PK_N	A public key of a single node
SK_N	A secrete private key of a single node
M_{NM}	A plaintext message sent from node N to M
E_N, E_{WK}	A message encrypted with PK_N of N, or WK of workflow
S_N	A message signed by node N
$Payload_{NM}$	A processed message sent from node N to M
$Notice_{NM}$	A feedback of payload received from node N to M
$SysLog_{NM}$	A system log records operation of node N for M
$AudLog_{NM}$	An audit log contains hash value of the related system log

4.2 System Architecture

We show a view of the system architecture of our proposed scheme (see Fig. 3), which includes three main components: nodes, audit server, and certificate authority.

- Nodes. They are participants involved in the workflow, such as authorities, stakeholders, and so on. In this paper, each node represents one of entities that collaborates and exchanges information in a workflow.
- Audit server. We run the audit server in the Ethereum blockchain. All audit trails are encrypted and then stored in the blockchain that can be accessed by nodes.
- Certificate authority (CA). It is a trusted authority to generate keys for diverse workflows. This can be a professional authority that depends on the workflow.

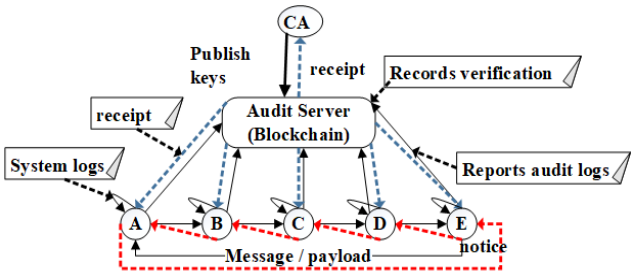

Fig. 3. The system architecture of the proposed scheme

4.3 Key Management

We assume that all entities have a unique identification and it can be used in the different workflows. Each workflow has a specified single pair of keys that can be only used in this workflow.

- Identity key management. To identify all relevant participants, all of them have their unique pair of keys when they register in the blockchain. Every public key is

stored in the blockchain. Participants save their private keys as identities and use them to approve transaction in the workflow.

- Workflow key management. A CA provides a unique single pair of keys for every workflow. The private key of the workflow is divided into pieces of partial keys based upon the cryptographic algorithm of Shamir's secret sharing [17]. The amount of Shamir threshold keys depends on numbers of participants in the workflow. Every participant has its own part of *WSK* for each workflow. At the same time, the CA stores all workflow public keys to the blockchain.

4.4 Protocol

We show a protocol to implement our auditable check scheme in this section. It is composed of three phases, which are system initialization, data exchange, and records verification. Fig. 4 shows a part of sequence diagram of the proposed protocol.

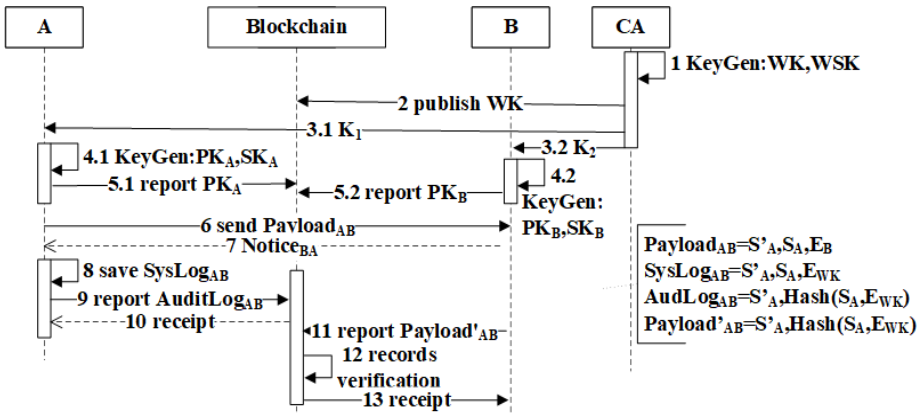

Fig. 4. A sequence diagram of our scheme. The initial phase (step 1-5) is key generation and distribution. Phase 2 (step 6-10) is data processes between participant and blockchain. Phase 3 (step 11-13) is records verification.

Phase 1: System Initialization. It aims to initialize keys of the participants and workflow. All participants have a cryptographic key pair (PK_N and SK_N) as their identities. A CA provides a single pair of keys for each workflow (*WK* and *WSK*). Each node has a public key (*WK*) and a split of the private key (K_n) of each workflow. In the Fig. 4, A has PK_A, SK_A, *WK*, and K_1. The blockchain stores keys of PK_A and *WK*.

Phase 2: Data Exchange. In this phase, the message sender signs and encrypts the predefined message to ensure the security of transmission. When A wants to send message to B, the initial payload is M_{AB}. First, A needs to sign M_{AB} and then encrypt it with key PK_B of B. The payload is represented by

$$Payload_{AB} = Encrypt_B\,[Sign_A\,(M_{AB}) + M_{AB}] = (S_A, E_B) \quad (1)$$

Then, A signs the encrypted payload again to mark the previous performed action before the payload transmission. The payload is expressed by

$$Payload_{AB} = Sign'_{A}\ (Payload_{AB}) + Payload_{AB} = (S'_{A}, S_{A}, E_{B}) \quad (2)$$

During the date exchange, a system log is generated to record the exchanged data. Each node stores their system logs in the local storage. The hash values of these system logs (namely, *audit log*) are published timely to the audit server as the immutable blocks. A system log includes an encrypted message with a key *WK* and a group of signatures. The second signature is to verify that encrypted logs have not been tampered with without having to decrypt the logs. When an audit log is saved in the blockchain, the message sender receives a receipt from the scheme. In the Fig. 4, $SysLog_{AB}$ is the system log that records the data exchange between A and B. $AudLog_{AB}$ is the published audit log to the blockchain for the audit trail. They are represented respectively by

$$SysLog_{AB} = Sign'_{A}\ [Encrypt_{WK}\ (Sign_{A}, M_{AB})] + Encrypt_{WK}\ (Sign_{A}, M_{AB}) \quad (3)$$

$$AudLog_{AB} = [S'_{A}, Hash\ (S_{A}, E_{WK})] \quad (4)$$

Phase 3: Records Verification. This phase is to verify all performed actions of data flow from workflow participants. The participant always checks whether the hash value of encrypted payload ($Payload'_{NM}$) is matched with audit log ($AudLog_{NM}$). When the recipient received the payload ($Payload_{NM}$) from the sender, the cipher message (M_{NM}) is decrypted with a private key (SK_{N}) of the recipient. Before the match, M_{NM} is encrypted again with a workflow public key (*WK*) by the recipient and conducted as a new payload ($Payload'_{NM}$). Then, it is the comparison of the hash value of $Payload'_{NM}$ and $AudLog_{NM}$ in a smart-contract. If the result of match is false, the workflow is stopped. Considering the integrity of data in the flow, the recipient needs to give a feedback ($Notice_{NM}$) to the sender when the payload is transferred. For example, when B receives payload from A successfully, a notice is sent to A.

Then, B gets the M_{AB} from the payload through the decryption of the $Payload_{AB}$ with key S_{KB}. In the match, B encrypts M_{AB} with key *WK* and calculates a hash value of it. The new payload is represented by

$$Payload'_{AB} = [S'_{A}, S_{A}, Hash\ (E_{WK})] \quad (5)$$

5 Performance Evaluation

We implement our scheme in the Ethereum blockchain, with the blockchain as the audit server that is conducted in a smart-contract for the data verification and audit log storage. We design a simple user interface as the interaction client for the workflow participant, which is to report and download audit log, and trigger with the smart-contract. The following context is also to analyze how the scheme achieves the security requirements.

5.1 Implementation

The implementation of our scheme is mainly to build codes of smart-contracts. Fig.5 shows a representative smart-contract code. There are two smart-contracts to enable the records verification and audit logs reporting. First contract 'AuditLog' constructs a function 'generateLog' to save audit trail into the blockchain as the immutable storage. Second contract 'Verification' is an inheritance contract of the first one, it is developed to access audit trail from the blockchain and verify the records. The function 'getLog' is to get audit trail by the specified address, notably, the account address of audit log reporter. The function 'compareLogs' is to compare hash values of audit trail and payload. This function is required to only operate by the current account of participant. 'ownerOf' function is a modifier to implement the operation control for function 'compareLogs'. When a node performs the data transmission in the workflow, a new contract will be created to save audit log into the blockchain. Once a node receives a payload from the previous node, the node can verify the payload through the smart-contract.

```
Contract AuditLog {
   string hashOfMessage;
   string signature;
   event NewLog(uint logId, string signature, string hash);
   struct Log { string signature; string hash; }
   Log[] public logs;
   Mapping (uint => address) public logOwner;
   function generateLog(string _signature, string_hash)
      internal returns (bool) {
      _hash = hashOfMessage;
      _signature = signature;
      uint id = logs.push(Log(_signature, _hash)) – 1;
      logOwner[id] = msg.sender;
      NewLog(id, _signature, _hash);
      return true;
   } }
Contract Verification is AuditLog {
   mapping (address => string) previousLog;
   modifier ownerOf(uint _logId) {
      require(msg.sender == logOwner[_logId];
      _;
   }
   function getLog (address _myAddress) public view returns(string) {
      return previousLog[_myAddress];
   }
   function compareLogs(uint _presentId, uint _targetId)
      public view ownerOf(_presentId) returns (bool) {
      Log storage presentLog = logs[_presentId];
      Log storage targetLog = logs[_targetId];
      if (keccak256(abi.encodePacked(presentLog.hash))
         == keccak256(abi.encodePacked(targetLog.hash)) )
      { return true; }  else { return false; }
   } }
```

Fig. 5. The central smart-contract. One contract is to generate a new block to save the audit log with function 'generateLog'. The second one is an inheritance of the first one that verifies the data from audit log and payload, which consists of function 'getLog' and 'compareLogs'.

5.2 Security Analysis

We discuss the security requirements for the proposed scheme in malicious operations as below. It includes malicious participant and collusion attacks.

Malicious Participant. In a workflow, a dishonest entity can eavesdrop data from the transmission channel, disrupt the data flow, or plant a forged message into the flow. However, the honest entity can detect these attacks with the audit record verification mechanism in the proposed scheme. We discuss internal and external aspects of malicious attacks. For the internal attacks, if an internal node tries to withhold a payload to interrupt the data exchange, the next node cannot receive the related payload. Therefore, this malicious attempt is detected on the fly. If the internal node uses the fraudulent data instead of the original payload, it can be detected in the records verification. There is an error when the hash value of the fake payload matches with the original one in the audit log. If the node tampers with or removes a local audit record, records on the audit server will reveal the malicious activities because of the immutability of blockchain. As for the external attacks, based on the assumption 2 and without the knowledge of participants' private keys, the external node cannot plant a forged message to pass the verification. The honest node can detect the attempt. In addition, the message are exchanged in encrypted form, it makes eavesdropping on the data flow useless to external attacks.

Collusion Attacks. When two or more than two nodes collude with each other in the data flow, their fraudulent actions can be exposed by the honest node (assumption 3). For instance, we assume that B colludes with C in Fig. 3. When D receives the forged payload ($Payload_{CD}$) from C, the hash value comparison between $Payload_{CD}$ with $AudLog_{CD}$ is triggered by D. If there is not match in the comparison, C is suspected of that malicious behavior. What's more, even if C repudiates it and ask B to frame A, we can verify $AudLog_{AB}$ and $Payload_{AB}$ to against it. If B colludes with D, they plant forged payloads ($Payload_{BC}$, $Payload_{DA}$) and deny their performed actions. For this case, the honest node can also detect it. C and A can verify payloads separately when they receive payload. As seen, our proposed protocol mitigates the impact of collusion attacks as possible.

6 Conclusion

In this paper, we discussed the usage of the Ethereum blockchain to enable auditing of workflow transactions. We provided a blockchain-based smart auditable check scheme that constructs a complete immutable audit trail for every action of participants in data transmission. Our audit scheme satisfies our aim to enable confidentiality, integrity, and accountability for a generic topology of data flow. As for future work, we will test the scheme in the real Ethereum network. The latency of new block generated is a consideration that affects the data flow efficiency. Besides, the generation of key pair for each workflow is also concern due to human factors from the certification authority. Furthermore, the security and stability of smart-contract need to be analyzed.

References

1. C. Reddick and L. Anthopoulos, “Interactions with e-government, new digital media and tradtional channel choices: citizen-initiated factors,” *Transforming Government: People, Process and Policy*, vol 8, no.3, pp. 398-419, 2014.
2. A. Nehme, V. Jesus, K. Mahbub, and A. Abdallah, “Decentralised and Collaborative Auditing of Workflows,” *16th International Conference on Trust, Privacy and Security in Digital Business*, in press.
3. C. Esposito, A. De Santis, G. Tortora, H. Chang, and K. K. R. Choo, “Blockchain: A Panacea for Healthcare Cloud-Based Data Security and Privacy?,” *IEEE Cloud Comput.*, 2018.
4. M. Crosby, Nachiappan, P. Pattanayak, S. Verman, and V. Kalyanaraman, “Blockchain technology: Beyond bitcoin,” *Appl. Innovvation Rev.*, no. 2, pp. 6–19, 2016.
5. M. Nofer, P. Gomber, O. Hinz, and D. Schiereck, “Blockchain,” *Springer*, vol. 59, no. 3, pp. 183–187, 2017.
6. R. Guo, H. Shi, Q. Zhao, and D. Zheng, “Secure Attribute-Based Signature Scheme with Multiple Authorities for Blockchain in Electronic Health Records Systems,” *IEEE Access*, 2018.
7. H. Li, K. Fan, Y. Yang, Y. Ren, and S. Wang, “MedBlock: Efficient and Secure Medical Data Sharing Via Blockchain,” *J. Med. Syst.*, 2018.
8. K. Gai, Y. Wu, L. Zhu, M. Qiu and M. Shen, "Privacy-Preserving Energy Trading Using Consortium Blockchain in Smart Grid," in *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3548-3558, June 2019.
9. K. Gai, Y. Wu, L. Zhu, L. Xu and Y. Zhang, "Permissioned Blockchain and Edge Computing Empowered Privacy-preserving Smart Grid Networks," in *IEEE Internet of Things Journal*.
10. T. Antipova, “Using blockchain technology for government auditing,” in *Iberian Conference on Information Systems and Technologies, CISTI*, 2018, vol. 2018-June, pp. 1–6.
11. A. Ahmad, M. Saad, M. Bassiouni, and A. Mohaisen, “Towards Blockchain-Driven, Secure and Transparent Audit Logs,” in *15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, 2018, pp. 443–448.
12. L. Castaldo and V. Cinque, “Blockchain-based Logging for the Cross-Border Exchange of eHealth data in Europe,” in *First International ISCIS Security Workshop*, 2018, vol. 821, pp. 46–56.
13. J. Anderson, “Record Audit Logs Through Permissioned Blockchain Technology.” 2018.
14. P. W. Abreu, M. Aparicio, and C. J. Costa, “Blockchain technology in the auditing environment,” in *Iberian Conference on Information Systems and Technologies, CISTI*, 2018, vol. 2018-June, pp. 1–6.
15. W. Pourmajidi and A. Miranskyy, “Logchain: Blockchain-Assisted Log Storage,” in *IEEE International Conference on Cloud Computing, CLOUD*, 2018, vol. 2018-July, pp. 978–982.
16. S. Suzuki and J. Murai, “Blockchain as an Audit-Able Communication Channel,” in *IEEE 41st Annual International Computer Software and Applications Conference*, 2017, vol. 2, pp. 516–522.
17. A. Shamir, “How to share a secret,” in *Comm. ACM*, 1979, vol. 22, pp. 612–613.