2017 IEEE 41st Annual Computer Software and Applications Conference

Blockchain as an Audit-able Communication Channel

Shigeya Suzuki
Graduate School of Media and Governance,
Keio University
and
Blockchain Laboratory, Keio Research Institute at SFC
Fujisawa, Kanagawa, Japan 252-8520
Email: shigeya@wide.ad.jp

Jun Murai
Faculty of Environment and Information Studies,
Keio University
and
Blockchain Laboratory, Keio Research Institute at SFC
Fujisawa, Kanagawa, Japan 252-8520

***Abstract*— Applications requiring strict access control, such as medical record query, often require auditing of the query. The current typical design relies on server side logging. However, logging on server-side do not provide strict means of auditing, since the server can be tampered with attackers, and also anybody who has permission to write can modify the log. We propose a scheme using blockchain technology, as a request-response channel for a client-server system, to record both client request and server reply in an audi-table manner. We have implemented a proof-of-concept system on top of a publicly available blockchain testbed. By using a blockchain as a client-server request-response channel, the request-response sequence can be verified by anybody who has access to the blockchain, providing a way to implement audit log for strictly controlled resources.**

I. Introduction

Applications requiring strict access control demand strict auditing – such as logging of access to a server – in a sustainable, tamper-proof manner. Many of the systems nowadays are implemented as a client-server model. Systems that handle personal records, especially sensitive information such as medical records are one example of such a requirement [1]. They require evidence of a request and also the result of the request – either refused or granted. If the request is granted, the record of provision needs to be recorded too.

Implementing strict auditing is challenging on any distributed system. Servers can be tampered by itself. If logs are not secured by some means, they can be modified. Deploying a highly available log server, then concentrating the logs on the server might be a solution, but since the server is running with software, it is not tamper-proof.

Blockchain is a way to implement a verifiable, difficult to tamper ledger over distributed servers [2], [3]. Due to the blockchain technology based system's nature, once the transaction is confirmed, the transaction is very hard to modify in a verifiable way. Blockchain is highly fault-tolerant because it is implemented as a peer-to-peer system.

We propose to use blockchain as an audit-able communication channel. A sender who wants to send data to a receiver puts a message marked with the receiver's identity onto a blockchain based communication channel. The receiver is waiting for incoming messages which includes marking of the receiver's identity. Once the receiver finds a message, the receiver can process it. For a client-server model based communication, the server processes the message, then creates a message and sends the message back to the client, via the blockchain based communication channel. Since the messages are placed on the blockchain, the message, either in clear text or encrypted form, can be retrieved and verified as needed.

The remainder of the paper is organized as follows. In Section II, we will discuss current techniques of auditing and its issues, then provides basic background on blockchain technology in terms of our proposal. In Section III, we will describe our proposal. In Section IV, we will describe our proof-of-concept implementation. In Section V, we will evaluate our scheme. In Section VI, we will overview related works. Finally, we conclude the paper in Section VII.

II. Background

In this section, we discuss auditing of the client-server model, provide an overview of blockchain and its properties.

A. Client-Server model and Auditing

The client-server model provides a way to request a function on a server from a client, possibly on different networked nodes. A typical client-server system works as follows (see Fig. 1).

1) The client sends a request message to the server. The message may contain some data related to the function that the client want to execute, and also the client's identity and/or a credential for access control purposes. The message may contain a function selector, if the server provides multiple services on the same interface to receive the request.
2) The server receives the message, and checks whether the client has rights to use the server with the client's identity or the credential. Then it executes a function by using the data included in the message.
3) The server sends a message as a reply to the client, with some data generated by the executed function. The

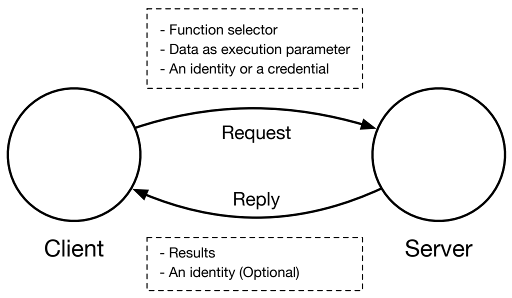

Fig. 1. Typical client-server based request-response model

message optionally includes the server's identity, for the client's verification.

4) The client receives the reply message. It optionally checks the server's identity in the message. Then, it uses the data in the message if needed, accordingly.

The messages exchanged between the client and the server of interest for auditing, but it is hard to create such an audit without the help of either the client or the server. Firstly, if it is possible to monitor and record the message exchange between the client and the server, we can create an audit for it. However, this is difficult, since we cannot monitor such a channel easily. Secondly, even if we can monitor the messages, it is usually encrypted nowadays. This makes third party auditing impossible.

Even if the client or the server helps to create an audit, the authenticity of the audit is doubtful. The server can be tampered with attackers. The software on the server can be tampered by attackers too. The data on the server can be altered not only by attackers, but also by malicious maintainers. Also, software bugs or hardware malfunction of the server can corrupt audits.

For these reasons, creating reliable audit on either on the client or the server is difficult or even impossible. This also means that creating a reliable audit on a distributed system is difficult or even impossible. It might be possible to create an audit with the help of hardware, like using a write-once device, hardware assisted scheme we will overview in Section VI, or hard copies, but such a scheme has limitations.

B. Blockchain

Blockchain is a peer-to-peer (P2P) distributed ledger which is inherently resistant to modification of data. It is resistant to modification of data, because 1) The ledger is replicated among participating nodes – makes a modification to the replica on the nodes to be meaningless, and 2) Nodes are working together to achieve agreement on the up-to-date snapshot of the ledger by a consensus algorithm. Bitcoin is using revolutionary consensus algorithm called "Proof of Work." By combining these two, the nodes can create a distributed, modification resistant ledger.

The name "blockchain" comes from the design[1], in which nodes are continuously creating chained blocks using a cryptographic algorithm. The idea of a cryptographically secure chain of hash has existed in the past [5], [6]. While the proposed schemes provide better protection over simple logging scheme, it does not solve the issues caused by an insecure server. The first design concept of blockchain appeared as the original Bitcoin paper by Satoshi Nakamoto [2], and later its implementation appeared in 2009 [7].

[1]Originally, name "block chain" was used in the Bitcoin implementation [4]

In a blockchain, each of the chained blocks contains multiple transactions. A transaction is created by a client and send to one of the blockchain network nodes. The client is not a necessary part of the blockchain network node. When the blockchain P2P network node receives a transaction from a client, it broadcasts the received transaction to the P2P network it is joining. Part of the blockchain P2P network node is running some consensus algorithm. The nodes store the received transactions into a pool. Each node selects a set of transactions to be in a new block, then, it tries to agree on the next block to be part of the blockchain with other nodes by using a consensus algorithm. Once nodes have reached consensus, each node makes the agreed block – with agreed contents – to be part of the blockchain.

To chain the blocks, a consensus algorithm is used to agree on which block to connect next. Bitcoin, which is the first implementation of blockchain, is using an algorithm called "Proof of Work" (hereafter called "PoW"). Each PoW participating node tries to solve a cryptographic puzzle. Since the puzzle is computationally heavy, the probability of solving the puzzle faster depends on the amount of computational power in use to solve the puzzle. In other words, how much the node "worked." PoW nodes are called "Miner" due to the fact the first solver of the puzzle get two kind of coins as rewards: transaction fees and newly generated coins given on each solution of the puzzle. Since PoW is inefficient in various regards, consensus algorithm like Proof of Stake [8], [9] is proposed. For another blockchain implementation, Hyperledger Fabric [10], Practical Byzantine Fault Tolerance [11] can be used for deployment in a limited context.

Blockchain technology has several useful characteristics, but the two most important properties are, first, that it works without any authority, and second, it is resistant to modification. Since blockchain shares the characteristics of a P2P network system, the system is inherently resilient to fault, and also scalable. By keeping copies of the whole blockchain, modification to any participating nodes does not affect the blockchain's trustworthiness.

Blockchain was born as an underlying technology for cryptocurrencies, but due to these useful characteristics, applications using this technology are increasingly spreading. MedRec [12] is one example of such an application.

III. Proposed Scheme

In this section, we introduce our scheme. We describe the requirements of the scheme, describe the messaging of the scheme, then describe the characteristics of the scheme.

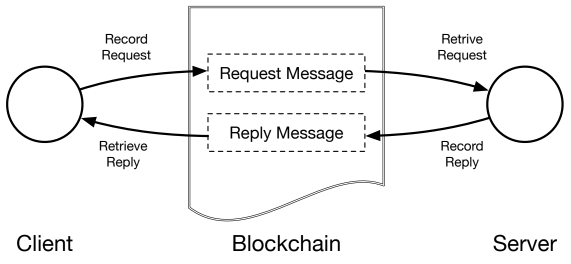

Fig. 2. Proposed Scheme

A. *Blockchain Properties Required by the Scheme*

The proposed scheme has the following requirements. Any blockchain platform can be used as a basis for the proposed scheme, as long as these following conditions are met:

- A blockchain records transactions in chained blocks.
- Each transaction may contain extra data area freely used to store the request-reply data.
- Participating nodes can retrieve the identity of the sender and the receiver of the transaction, either by the signature used in the transaction or by an authentication credential stored in the extra data area mentioned above.
- The chain can be accessed by auditors in charge, not necessary, but possibly in public.

Many blockchain implementations including, but not limited to Bitcoin [2], Ethereum [3] and Hyperledger Fabric [10] meet the above conditions. With any of the above three blockchain implementation as a basis, we can implement as follows:

- Use a transaction as a unit of a message.
- Use the data area for the payload of the message, and if needed, for credentials.
- Use the identity of the payer and the payee in above cryptocurrency implementations to identify the sender and the receiver.
- The party who has access to the ledger may audit the communication.

B. *Messaging of the Proposed Scheme*

The proposed scheme work as the following steps (refer to Fig. 2):

1) The client prepares a request message with the selector for the function and the parameters for the function. Parameters can be encrypted optionally, as long as the identity of the destination can be visible in the message or the transaction.
2) The client sends the message, by recording the request message as a transaction, destined for the server. The client and the server's identities are provided as part of the transaction.
3) The server keeps watching for any transactions destined to it, by watching for the server's identity, to appear in the last chained – confirmed – blocks.
4) When the server finds a transaction with its identity as the destination, it retrieves and decode the data, then executes the function selected by request with the given parameters.
5) After finishing executing the function, the server prepares a reply message, with the execution result. Results can be encrypted optionally, as long as the identity of the destination can be visible in the message or the transaction.
6) The server sends the message back in a transaction destined to the client.
7) The client keeps watching for any transactions destined to it, by watching for the client's identity, to appear in the last chained blocks.
8) When the client finds a transaction with its identity as the destination, it retrieves and decodes the data, and uses the returned result.

C. *Characteristics of the Proposed Scheme*

There are two interesting characteristics of the proposed scheme: 1) Highly available, modification resistant and verifiable audit and 2) Lower visibility of the node location on the network. We describe them in this order.

1) Highly Available, Modification Resistant and Verifiable Audit: The proposed scheme provides a highly available, modification resistant and verifiable audit. Since it is implemented on top of a blockchain, it is assumed to be P2P based, thus it's highly available. Anybody who has access to the system can verify the communication. Anybody who has access to the system can verify the existence of communication between the client and the server. Anybody who has access to the system can verify the actual request made by the client, and the actual reply from the server, if it is not encrypted, or, if the verifier has the ability to decode the message. All of these characteristics can be the basis of secure auditing system.

2) Lower Visibility of the Node Location on the Network: All of the participating nodes can be located anywhere; It can be invisible from any third parties or even from the request-response peer. Since the system is implemented as a P2P system, the request from the client and the reply from the server can be sent from any node which has access to the system. Neither the clients nor the servers must provide its network address, so it can safely hide its network location. The only information that must be visible is participant nodes' identities.

IV. Proof of the Concept Implementation

We have implemented a Proof of Concept client-server system on top of the Bitcoin network and experimented with it. In the following subsections, we describe the proof of concept system, the experiment setup, and the experiment results. Note: to be terse in the description, we skip detailed description on Bitcoin implementation.

A. *Proof of Concept System*

The proof of concept system is designed based on the following decisions. Refer to Fig. 3 for what is in the request message and the reply message.

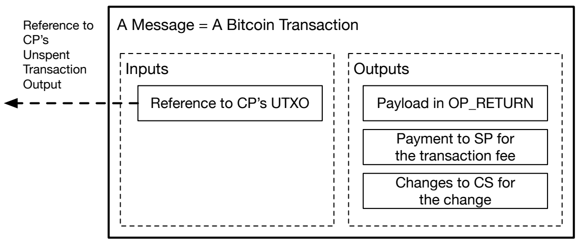
Reference to CP's Unspent Transaction Output
A Message = A Bitcoin Transaction
Inputs
Reference to CP's UTXO
Outputs
Payload in OP_RETURN
Payment to SP for the transaction fee
Changes to CS for the change

Fig. 3. A proposed scheme client to server request message stored in a Bitcoin transaction

- Since we use transaction in the Bitcoin blockchain, we need Bitcoin currency (BTC) for both transaction fee and actual transfer between the sender and the receiver. For a Bitcoin transaction, multiple inputs and multiple outputs can be specified.
- The client and the server each create and keep two sets of public key cryptography key pairs for Bitcoin addresses. The Bitcoin address is where the BTCs are bound to. The Bitcoin address can be created from the public key. We call each key or each Bitcoin address with the following names: client primary (CP), client secondary (CS), server primary (SP) and server secondary (SS). CP and SP are used for actual transactions. CS and SS are used to receive changes since it is not good practice to return changes to the payer's Bitcoin address directly.
- To store the message, we use the `OP_RETURN` script opcode [13] to store extra data in the transactions. By using the opcode, up to 40 bytes of data can be embedded in any transactions. The sender and the receiver of the messages are identified by the payer and the payee of the transaction, respectively.
- When the client sends a message to the server, the client creates a transaction with input(s) from one or more of CP's unspent transaction (called UTXO), three outputs for `OP_RETURN` script, SP for the actual transfer of BTC between participating nodes, and CS for the change. Also, transaction fees are implicitly included as the difference between the sum of the outputs and the sum of the inputs. The reason we need to return changes to CS is because it is necessary to use the whole UTXO as an input for any transactions.
- During the communication, CP and SP have enough balance to fulfill transaction fee. Assuming CP and SP communicate as client-response, CP and SP do not consume BTCs since CP sends to SP, then SP pays back to CP. However, the transaction fees are for the miner. We will review transaction fees in Sec. V.
- Both nodes monitor the blockchain and wait for any transaction for the receiving node to be included as a confirmed block. Bitcoin network adjusts the puzzle difficulty parameter so that every puzzle solution happens in just about 10 minutes. Thus, the expected delay for the confirmation block is about 10 minutes.

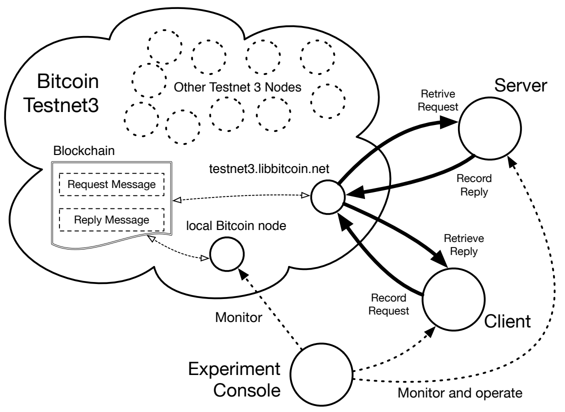

Fig. 4. A setup used for the experiment

TABLE I
EXPERIMENT CONFIGURATION

Item		Configuration
Operating System		Ubuntu 16.04 LTS
Blockchain		Bitcoin v0.14.0
Console, Client and Server	Hypervisor Hardware	iMac (Late 2015, Core i7 6700K 4Ghz), 24GB Memory, SSD
	Hypervisor	VMware Fusion (version 8.5.6)
	VM Config	2GB memory, 2 vCPU

We have implemented a client-side program and a server side program both in Ruby scripting language, with the help of a command line tool "Bitcoin Explorer" [14]. These programs send a dummy data in both requests and responses.

B. Experiment Setup

Fig. 4 shows the experimental setup, and Table I shows the software version and the configuration parameters used in the experiment.

We use "Testnet3" [15] for the experiment. Testnet3 is a Bitcoin network operating for testing purposes. Even for the testing purpose, we need active transactions other than ours to have chained confirmed blocks created by miners. Due to this reason, setting up a Bitcoin network by ourselves is not suitable.

We ran a Bitcoin node on a VM, which connect to the Testnet3 to monitor the activity during the experiment. We ran two VM nodes each for the client and the server. The client and the server nodes send and receive transactions via Testnet3's open accessible node [15] since we expect the node is close to the mining nodes. Both of our locally run Bitcoin node and Testnet3's open accessible node keep watching Testnet3's Bitcoin blockchain.

In addition to the above experiment setup, we used various transaction visualization websites which support Testnet3, such as Blockr [16] for transaction analysis purposes.

C. Experiment Results

While monitoring the transaction on our local Bitcoin node, we started the server, then ran the client. The client sends and waits for the transaction destined to the client to appear as the reply.

We summarized the timeline of the events in Table II. We experimented on April 22nd, 2017. Timestamps are in GMT. The "Block Time" column shows the time in the confirmed block. The "Monitor Time" column is the time recorded in the local Bitcoin monitor node's log. The numbers in parenthesis in the Block Time column shows the difference in seconds since the previous event, and the number in the parenthesis in Monitor Time column is the difference from the Block Time.

As the table shows, the request-response sequence is correctly ran. The time of the requested message confirmed as a block for the request was 720 seconds, and the time of the reply message confirmed as a block was 2265 seconds. Since the confirmed block number of the request and the reply transactions are 1119495 and 1119497, there is one confirmed block in between the confirmed blocks for the request and the reply. We evaluate the results in Sec. V.

V. Evaluation

In this section, we first evaluate the results of experiments using the proof of concept system, then later evaluate the scheme itself.

On the experiment on the proof of code, we observed that the transaction over a blockchain is possible, but the duration between the message sending and the message reception is long for typical client-server requests. Also, we have observed that the transaction had been modified due to a Transaction Malleability Attack [17], [18].

Firstly, the proposed scheme works as expected. We could encode our message as part of the transaction. The transaction was distributed over the P2P network and finally confirmed as a block in the blockchain. Since the blockchain of the Testnet3 is accessible by anybody, anybody can verify the existence of the transaction.

Secondly, and unfortunately, the Testnet3 was under Transaction Malleability attack at the time of the experiment, which caused our experiment to be difficult, unexpected, and unreliable. We need more experimental work in the Testnet3, real experiments with the Bitcoin production network, or experiment with other blockchain technology as a basis. On the reply transaction, while our purpose was achieved – we could identify our transaction by using the source and the destination addresses, and we could retrieve the data in `OP_RETURN` opcode without problem – the tweaked version of the transaction was included in the confirmed block. During the testing period, from time to time, the Testnet3 nodes would stop accepting transactions, or the open Testnet3 node would refuse to watch for any transaction destined for our destinations. Thus, while the request message was processed in reasonable time, the reply message looks unreliable. Unfortunately, up to the submission of this paper, we could not submit a new request to Testnet3. One other observation we want to add is, while 37 minutes passed since confirmed block that included the request appeared in the blockchain, only one block was confirmed before the block included the reply. This means there are not enough transactions flowing among the mining nodes to generate a confirmed block in the targeted 10-minute interval, causing longer communication interval for our proposed scheme. We will study the results more closely, and possibly run a test on the real Bitcoin production network.

The characteristics of the proposed scheme depend on the characteristics of the underlying blockchain technology. We separately discuss in Bitcoin-specific, and non-specific way.

With the Bitcoin, due to its design, the confirmed block will not appear quickly. For faster transactions, we need to use other blockchain technology or apply some technology to make the transaction to be part of the blockchain quickly. Bitcoin's PoW puzzle difficulty is tuned towards current computation power. The difficulty is tuned to solve the puzzle in just about 10 minutes. Also, the original Bitcoin paper estimated that the user should wait for confirmation of the five following blocks after the confirmed block includes the transaction, to achieve less than 0.1% probability of attack success [2]. In other words, we may need to wait up to 60 minutes to be sure for the transaction. If we need to apply the proposed scheme with quicker response time, we need to use different blockchain technology, or apply some acceleration technology to Bitcoin, such as Lightning Network [19]. Of course, there is an application which does not require fast response time, such as DRM key deliveries. For such an application, use of Bitcoin as a basis is still possible.

For the blockchain technology based cryptocurrencies, we need a fee to record a message, which can be expensive. With Bitcoin, any party that submits a transaction can decide the amount of fee paid to the miners. The miners will pick which transaction to be included in the next block mining. Since the miner who solves the puzzle for a block earn fees in all transactions in the mining block, there is a motivation to select transactions that gives higher fees. This affects how long it will take a transaction to appear in a confirmed block. In other words, transactions with higher fees will be processed more quickly. According to statistics on Predicting Bitcoin Fees for Transactions [20], as of April 24th 00:00 GMT, the fastest and cheapest transaction fee is 160 satoshis/byte. The typical transaction requires 226 bytes; thus 36,160 satoshis are the fee for a typical transaction (Note that we have used 50,000 satoshis for our transaction). 36,160 satoshis are about $.45USD at April 24th 00:00's BTC exchange rate, that is expensive for a request-response auditing. To solve this issue, we need to use non-cryptocurrency based blockchains or use a blockchain technology which do not require such amount of fees.

The security, reliability, or scalability of this scheme solely depends on the underlying blockchain technology used, require further study. On the safety of the blockchain, researchers are still working on evaluating how blockchain technology is secure or not in general, or how specific blockchain implementation is secure.

TABLE II
RESULT SUMMARY

Block Time	Monitor Time	Event
15:59:19 (0)	-	Request Message sent to the network (Transaction ID: 77d15a0b869402e6eacc68ef4df0464bf6f3fd0bc59e5b7ba097bb0b24ce2faa)
16:11:19 (720)	16:11:19 (0)	Request Message confirmation on the block #1119495
16:11:20 (1)	-	Reply Message sent to the network (Transaction ID: bee3f93c0f73eb8ade0c5b4c310d1ba3f06a0ff64357396462794d6d4fc23a8d)
16:49:10 (2265)	16:49:48 (38)	Reply Message confirmation on the block #1119497

VI. RELATED WORKS

In this section, we review related works. We first present a logging service model about the placement of the log. Then, we will present related work categorized into hardware assisted scheme and software only scheme.

When the information to be logged is generated on a server, the log will be either stored locally (local logging), or sent to a networked logging service. A networked logging service may consist of a single server, or a set of servers. The hardware assisted scheme for a local logging may apply to the servers which form networked services.

On hardware assisted scheme, Houlihan proposed an auditing system using Trusted Platform Module (TPM) for a cloud-based system [21]. Accorsi proposed BBox [22] using TPM, and System Management Mode (SMM) available for x86 architecture CPUs [23]. Alam proposed SBBox which implement a black-box-like hardware connected via PCIe bus [24]. Although this hardware assisted scheme is effective, the hardware based implementation poses limitations such as lower resilience to hardware trouble, possible single point of failure. The proposed scheme using blockchain does not rely on any hardware. Since the whole blockchain is replicated on a multiple number of nodes, and if the blockchain consists of a large number of nodes, the blockchain is resistant to hardware failure. The proposed scheme is also resistant to hardware failure if the underlying blockchain is resistant to hardware failure. For Bitcoin, there are nearly 7,000 nodes are up and running as of 15 May 2017 [25].

On software only scheme, Ma proposed FssAgg [26], a scheme using public key cryptography to sign a log entry. The key pair is generated during the initialization phase. The public key will be registered to a Certificate Authority for later verification. Yavuz proposed BAF [27], also using public key cryptography to sign a log entry. In this scheme, an offline trusted third party (TTP) generates a key pair to be used. Yavuz also proposed LogFAS [28]. This scheme also depends on an offline TTP to generate keys, but this scheme generates and uses a system-wide key pair and key pairs for each signer. For the computational cost difference between this software only schemes, refer to [28]. All of the above software only scheme require a TTP for key generation or a Certificate Authority to issue certificates. On the contrary, the proposed scheme in this paper does not need to rely on any TTP to verify the log. The amount of computation power required to prepare and send a request, are similar to FssAgg's update since creating of a transaction require several hashing operations and a signing operation.

VII. CONCLUSION

In this paper, we proposed using blockchain transactions as an audi-table communication channel. We have discussed the scheme in detail, then shown that the proposed scheme works on a proof of concept system which runs on top of the Bitcoin test network.

The authors believe that the applicability of the proposed scheme is good. The scheme can be applied to, but is not limited to, shared key delivery mechanism with auditing, and content encryption key delivery on DRM System or IP Multicast members. For a shared key delivery example, the authors plan to apply this for a personal health record storage system, based on an information management scheme we developed for RFID system [29], [30]. The scheme requires a shared key server which provides a shared key for a dataset. The key server returns the shared key for a dataset according to the requester's identity and query specification. The proposed scheme is applicable, and suitable to the key server protocol used for the system.

On the scheme itself, we can extend our scheme to multi-party conditional scheme, or transfer of the access rights, as a form of smart contract. For example, the service provider can enforce some condition to provide a service. The conditions can be implemented as a smart contact. For the personal healthcare system mentioned above, we want to extend this scheme to use access delegation, e.g., the patient and the physician jointly delegate access to the personal health records to another party, say, other physicians.

We also think that developing a blockchain system designed specifically for the proposed scheme's purpose is useful. For example, we can jointly operate an audit-able channel as a community effort. Large organizations are also able to operate their audit-able channel. To deploy such a system since the number of the nodes are limited, typical PoW based blockchain is not suitable because a small number of participating node weaken modification resistance and system resilience of the blockchain. Hyperledger Fabric [10] is designed modularity in mind, and a user can select consensus scheme such as Practical Byzantine Fault Tolerance [11]. The use of PBFT for consensus algorithm cause limitation, but using it can be suitable in such a context. By introducing such a limitation, we may develop a new blockchain system suitable for an audit-able channel.

The authors believe that based on the proposed scheme – or our finding on the use of blockchain as communication channel – can initiate blockchain technology's applicability beyond cryptocurrencies.

References

[1] J. King and L. Williams, "Secure Logging and Auditing in Electronic Health Records Systems: What Can We Learn from the Payment Card Industry Position Paper," in *Usenix HealthSec'12*, 2012.

[2] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[3] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, 2014.

[4] L. Trottier. original-bitcoin. Self-published code collection, "This is a historical repository of Satoshi Nakamoto's original bit coin source-code". This URL collected from Wikipedia. [Online]. Available: https://github.com/trottier/original-bitcoin/blob/master/src/main.h#L795-L803

[5] B. Schneier and J. Kelsey, "Cryptographic Support for Secure Logs on Untrusted Machines." *USENIX Security*, 1998.

[6] ——, "Secure audit logs to support computer forensics," *ACM Transactions on Information and System Security (TISSEC)*, vol. 2, no. 2, pp. 159–176, 1999.

[7] Bitcoin source at GitHub. (Verified on 2017/4/23). [Online]. Available: http://github.com/bitcoin/bitcoin

[8] Ethereum White Paper. (Verified on 2017/4/23). [Online]. Available: https://github.com/ethereum/wiki/wiki/White-Paper

[9] Proof of Stake FAQ – Ethereum Wiki. (Verified on 2017/4/23). [Online]. Available: https://github.com/ethereum/wiki/wiki/Proof-of-Stake-FAQ

[10] Hyperledger Fabric – Hyperledger Project Wiki. (Verified on 2017/4/23). [Online]. Available: https://wiki.hyperledger.org/projects/fabric.md

[11] M. Castro, B. Liskov *et al.*, "Practical byzantine fault tolerance," in *OSDI*, vol. 99, 1999, pp. 173–186.

[12] A. Ekblaw, A. Azaria, J. D. Halamka, and A. Lippman, "A Case Study for Blockchain in Healthcare: "MedRec" prototype for electronic health records and medical research data," 2016, (Verified on 2017/4/23). [Online]. Available: http://dci.mit.edu/assets/papers/eckblaw.pdf

[13] OP_RETURN – Bitcoin Wiki. (Verified on 2017/4/23). [Online]. Available: https://en.bitcoin.it/wiki/OP_RETURN

[14] Bitcoin Explorer. (Verified on 2017/4/23). [Online]. Available: https://github.com/libbitcoin/libbitcoin-explorer

[15] Testnet – Bitcoin Wiki. (Verified on 2017/4/23). [Online]. Available: https://en.bitcoin.it/wiki/Testnet

[16] BlockrIO on Bitcoin Testnet. (Verified on 2017/4/23). [Online]. Available: https://tbtc.blockr.io/

[17] Transaction Malleability – Bitcoin Wiki. (Verified on 2017/4/23). [Online]. Available: https://en.bitcoin.it/wiki/Transaction_Malleability

[18] C. Decker and R. Wattenhofer, "Bitcoin Transaction Malleability and MtGox," in *Computational Science and Its Applications – ICCSA 2012*, 2014.

[19] J. Poon and T. Dryja, "The bitcoin lightning network: Scalable off-chain instant payments," 2015, (Verified on 2017/4/23). [Online]. Available: https://lightning.network/lightning-network-paper.pdf

[20] Predicting Bitcoin Fees for Transaction. (Verified on 2017/4/23). [Online]. Available: https://bitcoinfees.21.co/

[21] Trusted Computing Group. TPM Library Specification. (Verified on 2017/5/14). [Online]. Available: https://trustedcomputinggroup.org/tpm-library-specification/

[22] R. Accorsi, "A secure log architecture to support remote auditing," *Mathematical and Computer Modelling*, vol. 57, no. 7-8, pp. 1578–1591, Apr. 2013.

[23] Intel Corporation. EFI System Management Mode Core Interface Spec(SMM CIS) v0.91. (Verified on 2017/5/14). [Online]. Available: http://www.intel.co.jp/content/www/jp/ja/architecture-and-technology/unified-extensible-firmware-interface/efi-smm-cis-v091.html

[24] M. Alam, Z. C. Lee, C. Nicopoulos, K. H. Lee, J. Kim, and J. Lee, "SBBox: A Tamper-Resistant Digital Archiving System," *International Journal of Cyber-Security and Digital Forensics (IJCSDF)*, vol. 5, no. 3, pp. 122–131, 2016.

[25] Bitnodes – Global Bitcoin Nodes Distribution. (Verified on 2017/5/15). [Online]. Available: https://bitnodes.21.co/

[26] D. Ma and G. Tsudik, "A new approach to secure logging," *ACM Transactions on Storage (TOS)*, vol. 5, no. 1, pp. 2–21, Mar. 2009.

[27] A. A. Yavuz and P. Ning, "BAF: An Efficient Publicly Verifiable Secure Audit Logging Scheme for Distributed Systems," in *2009 Annual Computer Security Applications Conference (ACSAC)*. IEEE, 2009, pp. 219–228.

[28] A. A. Yavuz, P. Ning, and M. K. Reiter, "Efficient, Compromise Resilient and Append-Only Cryptographic Schemes for Secure Audit Logging." *Financial Cryptography*, vol. 7397, no. Chapter 12, pp. 148–163, 2012.

[29] S. Suzuki, R. Van Meter III, O. Nakamura, and J. Murai, "Otedama: A Relocatable RFID Information Repository Architecture," *IEICE Transactions on Information and Systems*, vol. Vol.E93-D, no. 11, pp. 2922–2931, Nov. 2010.

[30] S. Suzuki, "A Relocatable RFID Information System," Ph.D. dissertation, Keio University, Graduate School of Media and Governance, Fujisawa, Kanagawa, Japan, Feb. 2012.