Distributed Authorized Medical Data Management

Chunmiao Li[1], Yang Cao[2], Zhenjiang Hu[3], Masatoshi Yoshikawa[4]

[1, 3] National Institute of Informatics, Japan [2, 4] Kyoto University, Japan

[1, 3] SOKENDAI (The Graduate University for Advanced Studies), Japan [3] University of Tokyo, Japan

Email: [1] chunmiaoli1993@nii.ac.jp, [2] yang@i.kyoto-u.ac.jp, [3] hu@nii.ac.jp, [4] yoshikawa@i.kyoto-u.ac.jp

***Abstract*—Electronic Medical data sharing between stakeholders, such as patients, doctors and researchers can promote more effective medical treatment collaboratively. These sensitive and private data could only be accessed by authorized users. Given a total medical data, users may care parts of them and other unrelated information might interfere the user-interested data search and increase the risk of exposure. Besides accessing these data, users may want to update them and propagate to other sharing peers so that all peers keep identical data after each update. To satisfy these requirements, in this paper we propose a medical data sharing architecture that addresses the permission control using smart contracts on blockchain and splits data into pieces shared with different peers then synchronize complete data and these pieces after updates with bidirectional transformations. Medical data reside on each user's local database and permission-related data are stored on smart contracts. Only all peers have gained the newest shared data after updates can they start to do next operations on it, which are enforced by smart contracts. Blockchain-based immutable shared ledge enable users to trace data updates history. This paper can provide a new perspective to view complete medical data as different slices to be shared with various peers but consistency after updates between them are still promised, which can protect privacy and improve data search efficiency.**

***Index Terms*—medical data, authorization, update, distributed**

I. Introduction

Now a lot of medical data are digitalized so as to be stored and accessed conveniently. A medical record is produced after a patient go to see doctor and often resides on the hospital's database. Medical records contain highly sensitive information about patient privacy. HIPPA Privacy Rule [1] in the U.S. regulates the use and disclosure of personally identifiable health information to protect patients' privacy. However, it hard to make sure that all medical institutes would follow these rules and they may expose patient privacy deliberately or for profit. Moreover, a patient might visit many hospitals and leave his records scattered [2] in different places, which make it is hard for he to manage his records efficiently. So patient should be provided a platform to manage and review his historical medical data in case of exposure or being tampered. Better communication between patients and doctors can contribute to patient adherence [3] and improved health [4]. In addition to provide data to doctor, patients tend to share their medical data with health experts to help understand some statistics. Many researches has been done to study how medical specialists with different expertise collaborate with each other [5]. Researchers can identify public hearth risks and then develop a better treatments by analyzing existing medical data [6]. As presented in [7], patients and experts can exchange information to develop better plans to satisfy individual routines. Sharing medical data under some constraints could benefit all relating stakeholders such as patients, researchers and doctors.

To be shared by multiple parties, shared medical data could reside in encrypted format on the trusted cloud. Only authorized stakeholders can access the shared data. Centralized access control might lead to single point of failure and become the bottleneck of sharing system. Some medical data sharing systems [8]–[10] are proposed to manage authentication based on blockchain [11] technology. Being a immutable shared ledger, blockchain can achieve consensus among distributed nodes via proof of work. Encoding the access control logic of medical data into smart contracts [8] or Chaincode [12] can prevent unauthorized party to access medical data, which will guarantee data security. Anyone who want to access medical data should be verified permission from blockchain side and their access process will be recorded on blockchain.

We realized that there are still another requirements on data sharing and state them as follow.

Firstly, generally different stakeholders may have different focus on the same medical records. For example, doctors might be more concerned with the clinical data and researchers are interested in mechanism of action, whereas patients care more about the medicine dosage standard. To reduce the size of shared data to improve access efficiency and avoid additional data interference, the complete medical data (data source) might be split into lots of smaller data pieces (data views) which are shared by different parties. Each view can be regenerated from the source. Source and view should satisfy predefined consistency relationship. Different views from one source might have overlapped data. Moreover, each party will have a local complete data and different data pieces to share with others. In this way, shared data are distributed among sharing parties.

Secondly, not only limited to access data, patients or doctors may want to update existing shared data. Although some works [8] claimed to allow updates on shared medical data, they did not propose any concrete scheme to implement updates and show how to synchronize all sharing parties to keep them still have same shared data after updates.

In this paper, we aim to solve authorized updates issues on distributed shared medical data as stated above. Each

participant may have multiple shared data [1] with different peers and keep this shared ones consistent with local complete database. Bidirectional transformations [13] (BXs hereafter) can help synchronize them after updates on either one side. For example, we can use *put* function in BX to reflect modifications on shared data to complete data and *get* to reproduce shared data from complete data. Moreover, any operations on the shared data should be conducted after the peer has been authorized. Any modifications on the existing shared data should be updated to all sharing peers immediately. We encodes the sharing peers not pointers to the raw data into smart contracts in case of private data leakage. Each sharing peer will receive the notification from smart contracts after an update on shared data are verified. Then each peer will request the newest data from updater and then use it to update his local complete data. Smart contracts will refuse any further operations on shared data before all sharing peers have pulled the newest data to their local database.

Our contribution are as follow.

1) We surveyed existing blockchain-based solutions and clarify their disadvantages, which are presented in Section V;
2) We designed a decentralized medical data sharing architecture where data reside on owner's local database and metadata are stored on smart contract of blockchain.
3) We proposed that the consistency relationship between local complete data and shared data and apply BX to keep consistency after updates;
4) We discussed the essential components needed for medical data sharing and proposed our solution to build a holistic solution utilizing blockchain to control updates permission.

The remainder are organized like this. Section II gives some preliminaries about blockchain and bidirectional transformations. Section III sketches our system architecture and provides a case analysis. Section IV discusses more details about our contributions and propose countermeasures against identified threats to our system. Section V compared our work with existing ones to clarify our improvement over them. Section VI concludes and directs our future work.

II. Preliminary

A. Blockchain

Proposed with Bitcoin [11] in 2008, blockchain technology has been widely used in many fields. Blockchain provides a solution for data storage, data transfer and consensus protocol in a distributed and decentralized environment. Generally speaking, blockchain is a shared ledger and replicated by all nodes on a distributed network, which records the historical valid transactions in a chronologically chained blocks. The nodes who generate new blocks via a proof-of-work are called miners.

Not only can support the platform of cryptocurrency, blockchain can also be applied to other scenes. Ethereum [14] extend blockchain with additions such as a built-in Turing-complete programming language so that one can use this scripts (i.e., Ethereum Virtual Machine (EVM) byte codes) to write programs (i.e., smart contracts [2]) on blockchain. We can just write Solidity [3] programs and then it can be compiled to EVM code. Besides the user accounts controlled by private keys like in Bitcoin, the accounts for smart contracts are allowed in Ethereum. Anyone can build decentralized applications which consists of a collection of smart contracts. Once a transaction involving smart contract creation gets confirmed, an address is generated for the contract and later anyone can send transactions to this address for executing the logic on it. A smart contract transaction is enforced when a miner includes it in a new produced block. Other nodes will validate it and re-run contracts if it is valid.

B. Bidirectional transformation

Maintaining consistency between different data representations (i.e., source and view) with "the same" underlying meaning [4] is important [15]. For example, view update problem in database [16] studies how to translate the updates on a view table to the updates on the relating base (source) table. To achieve this, one may think to provide two separate programs to represent two directions to propagate the updates to one side to another. But it is hard to prove source and view can still keep consistency after updates. Bidirectional transformations (BXs hereafter) was proposed [17] to solve this.

A BX program can be invoked from forward and backward transformations. A forward transformation (denoted as *get*) extracts elements from the source to build an abstract view, and the backward transformation (denoted as *put*[5]) embeds information of the view back into the source and produces an updated source. This pair of transformations should satisfy the *round-tripping* laws (i.e., *well-behaveness*) called *PutGet* and *GetPut* properties.

$$get(put(\textbf{source}, \textbf{view})) = \textbf{view} \qquad (\textit{PutGet})$$

$$put(\textbf{source}, get(\textbf{source})) = \textbf{source} \qquad (\textit{GetPut})$$

Specifically, *GetPut* refers that no extra update should be performed back on the source when there is no change on the view, while *PutGet* hints that *put* should apply all changes on the view to the source so that the view can be regenerated from the updated source by *get*. One BX program can represent two directions and we can achieve the transformation from one side to another by invoking *get* or *put* direction of it. The most distinguished point of BX is that view can contain only a few part elements of source. By designate some consistency between source and view, BX programs can synchronize

[1]Our work assume that the initialization of shared data has been finished. We only consider management on existing shared data.

[2]Hyperledger and others still provide platforms to write smart contracts.

[3]https://solidity.readthedocs.io/en/v0.5.2/

[4]Here we consider view is a part of source.

[5]*put* is not a simple inverse of *get*. Instead, it accepts the view and the original source as input and produces an updated source as output.

the source and view and promise that they still satisfy the consistency after updates on either one side.

One recent big progress towards practical use of BX is the observation that the essence of bidirectional programming is nothing but writing a well-behaved backward transformation (*put*) [13], [18], [19], because the corresponding forward transformation can be derived uniquely and freely. A core language, BiGUL [20], has been developed for putback-based bidirectional programming.

III. System architecture

We are ready to illustrate our system architecture in this section. Part A describes the data [6] distribution between sharing peers. Part B presents our system design and later explain it using an update scenario.

A. Data distribution

For later description, we regulate some nomenclature which are shown in TABLE I.

TABLE I
ADOPTED NOMENCLATURE

Nomenclature	Meaning
Nodes	A device connected to blockchain network
Users	stakeholders in medical scenarios
Sharing peers	Some users who share the same data

Shared data can exist in different peers, as shown in Fig. 1. Suppose there are three entities, doctor, researcher and patient. Doctor and Researcher share some data which are stored in D12 on Doctor side and D21 on researcher side respectively. D12 and D21 should contain the same data, which means if doctor or researcher update this shared data on their own side, then this modification should be propagated to the shared part on the other side. Accordingly, D13 and D31 have the same content and so do in D23 and D32. Notably, the format and contents are determined by sharing peers.

Each node will store its all data on a bigger database, such as D1, D2 and D3. Each shared data can be seen as a view which is produced by this bigger database named as source. For example, D12 can be made from D1 by using *get*. If D12 is modified, then the D1 need to be reproduced from original D1 and D12 by using *put*. We just write BX in researcher and patient side to express the pair of *get* and *put* functions.

Note in our data distribution, shared data between any two parties will not be exposed to the third party, which can keep privacy between two parties in some degree. For example, any update on D12 and D21 can only be known by patient and researcher. However, after the change on D21 are reflected to D2, since D2 has been modified, D23 might need to be updated to a new version by using *put* on D2.

Fig. 2 presents an example to show data distribution between patient Alice, doctor Bob and researcher Charlie. Each party have their own local base table, named as D1, D2, D3 respectively. For each table, there are some attributes such as medication name and address on D1. Table D13 (also D31) are shared by Alice and Bob and can be produced from either D1 or D3 by *get*. Table D23 (also D32) are shared by Charlie and Bob and can be generated from D2 or D3 by *get*. Shared tables all stay in sharing peer's local databases. Note here the metadata collection table resides in the smart contract on blockchain. Each entry of metadata collection table corresponds to a shared table. For example, the entry for D13 or D31 declare that it is shared by Alice and Bob and Bob can update all attributes value but Alice can only change the clinical data. The "Latest Update Time" shows when the metadata was changed most recently. Moreover, "Authority to Change Permission allow Bob to change other peers (here just Alice)'s authority so that Alice can update dosage by change the value for "Dosage" attribute to "Bob, Alice".

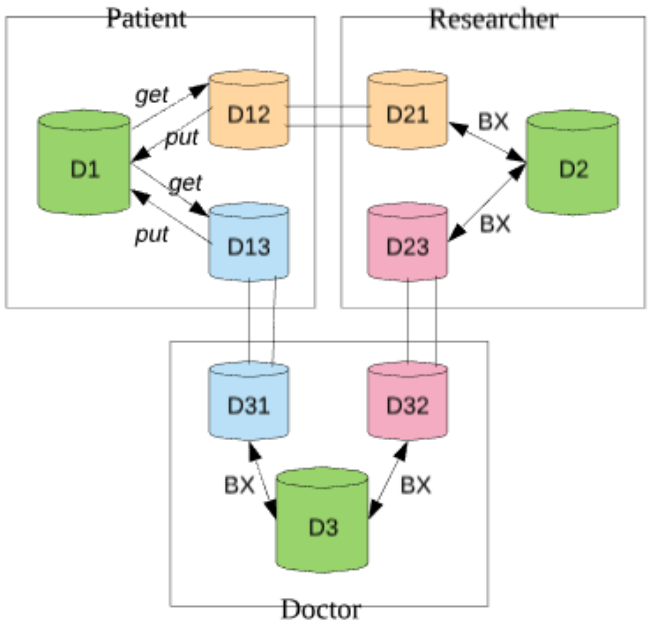

Fig. 1. DB architecture

B. Permission on update data

The metadata are created immediately once some peers wants to share data with each other. Suppose doctor Bob initiates the data sharing with Alice. He will deploy a smart contract on blockchain which stipulates the metadata about the shared data, such as sharing peers (i.e., Alice and Bob) and so on. Table Metadata Collection on Fig. 2 are built by Bob.

Since smart contract can not be altered after it was deployed on blockchain. There are two ways to update the permission for update to shared data:

1) deploy a new contract and notify all nodes on blockchain that this new one should be used to verify authority later;
2) update the state of variables in contract.

We choose the latter way. As shown in Fig. 2, the authority to modify the update permission are encoded in the field named

[6]In our prototype, medical data are entered directly by nodes such as doctor. Later we may consider to use the data from wearable devices.

Patient ID	Medication Name	Clinical Data	Address	Dosage
188	Ibuprofen	CliD1	Sapporo	one tablet every 4h

D1 (Patient **Alice**)

Patient ID	Medication Name	Clinical Data	Mechanism of Action	Dosage
188	Ibuprofen	CliD1	MeA1	one tablet every 4h
189	Wellbutrin	CliD2	MeA2	100 mg twice daily

D3 (Doctor **Bob**)

Patient ID	Medication Name	Clinical Data	Dosage
188	Ibuprofen	CliD1	one tablet every 4h

D13 (also D31)

Medication Name	Mechanism of Action
Ibuprofen	MeA1
Wellbutrin	MeA2

D23 (also D32)

Medication Name	Mechanism of Action	Mode of Action
Ibuprofen	MeA1	MoA1
Wellbutrin	MeA2	MoA2

D2 (Researcher **Charlie**)

Metadata ID	Sharing peers	Authorized Update Peer			Last Update Time	Authority to change permission
D13 & D31	Alice, Bob	Medication Name	Dosage	Clinical Data	2018.12.22	Bob
		Bob	Bob	Alice, Bob		
D23 & D32	Bob, Charlie	Medication Name		Mechanism of Action	2018.12.23	Bob
		Bob, Charlie		Charlie		

Metadata Collection (stored in smart contract)

Fig. 2. Data distribution

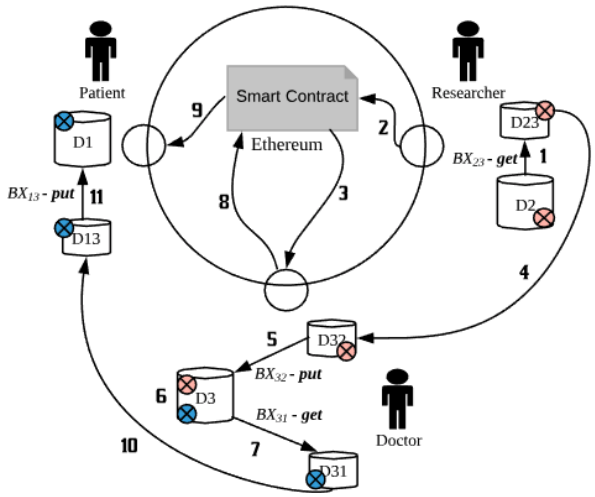

Fig. 3. An work flow for updating shared data

as ”authority to change permission”. For example, for D13 and D31, Bob can change the permission to update ”Dosage” to ”Bob, Alice” so that Patient Alice can also update the ”Dosage”.

C. System design

Our system consists of following components, which can be seen from Fig. 3.

- Front-end user interface: control the interaction between users and other components.
- Database: each user has an overall database and many shared databases with other users. The latter can always be reproduced from the former.
- Blockchain: keep a record of access and update to the shared data. Also, fonrt-end user interface communicate with blockchain network via a blockchain node. The smart contract on blockchain contains the metadata of shared medical data and maintain an update log to store historical modification for each metadata.
- Database manager (server) dispose the synchronization between shared data and local data in terms of consistency logic relations.

D. Case analysis

Fig. 3 depicts an scenario where researcher initiates the update the shared data. The numbers indicates the corresponding operations sequence. D1, D2, D3 are user's local database. D23 are identical with D32 and shared by the doctor and the researcher. D31 and D13 are same and shared by the doctor and the patient. Every shared data correspond to a BX program. For example, $BX_{23} - get$ represents the invoking on *get* direction on the BX program which is used to synchronize D2 and D23. The red and blue circles with wrong sign indicate the updated places.

Let's try to describe this work flow for updating shared data. After the update on D2, the researcher wants to propagate the update to the shared data D23 so that he uses the BX_{23}-*get* to regenerate D23 (step 1). Then he will call a smart contract via a node connected to Ethereum by sending the request for updates to the D23 (step 2). Note that the smart contract records all permission info about that D23. Only when the researcher are authorized his update are seen valid and can be propagated to D32. The smart contract will be executed until all nodes form consensus on this update request, which means researcher is permitted to update D23. Each node will conduct the smart contract locally. The entry relating D23 & D32 of the contract are modified and doctor will receive notification that D32 need to modified (step 3). Then he will request data from the researcher by sending a message based on some encrypted communication protocol and get the newest shared data to refresh D32 (step 4). After that, doctor will use BX_{32}-*put* to reflect the change on D32 to D3 (step 5). Since doctor shared some data (D31) with patient, he need to check whether D31 need to be reproduced (step 6). (If there is no need to reproduce, step 6 - 11 will not happen.) If yes, he will use BX_{31}-*get* to regenerate D31 (step 7) and request smart contract for permission to update D13 (step 8). Once allowed, patient will receive the notification about the change on shared data (D31) (step 9) and ask doctor to send the updated D31. After patient get the modified D31 (step 10), he will use this to update D1 via *put* (step 11).

Step 1 - 5 achieve the propagation of updates from one peer to another. Especially, step 6 check whether the images of D32 and D32 on D3 have some dependences. Step 7 - 11 have the same operational logic.

IV. DISCUSSION

In this section, we add more details about our contributions. In addition, we identify some threats to our system and propose relating countermeasures.

A. Detailed contributions

- Nodes can split total data into multiple pieces which can be shared with different nodes, which make sharing exists among only a group of nodes. So that it can avoid the interference or attack from other nodes. Meanwhile, data provider can choose what they want to share with others without exposing sensitive or private information.
- Any update on shared data can be reflected to local total database by using *put*. Consistency between shared data and local total data are firmly promised by BX.
- Permission to updates to medical data are stored in smart contracts so that blockchain can prevent operations from malicious nodes.
- Raw medical data always stay in each peer's local database and data transfer only exist between sharing peers, which avoid data being leaked to the third party so as to keep shared data security .
- Blockchain's consensus protocol scheme keep the shared data between sharing peers are same after updates since each peer will receive the notification from contracts and pull new shared data from other sharing peers.
- Any modification on shared data can be recorded on blockchain. Blockchain properties such as immutability, auditablility and transparency enable nodes to check and review update history on shared data.
- Simultaneously updates to the same shared data by multiple peers are forbidden. Smart contracts dispose the updates according to received requests in chronological order. If a transaction for updates on shared data has been included in a block, then other requests on this shared data will not be accepted, i.e., one block can contain one transaction at most on some shared data at one time. This can promise that only when all sharing peers have had newest shared data can they execute further operations.
- Each node can be a shared data provider. As referred in [7], many clinics encourage patients to collect data by themselves that are supposed to be gathered by doctors and expect to increase clinic efficiency and promote patient awareness.
- Our system can also be applied to other scenarios besides medical data sharing.

B. Threats and countermeasures

1) Throughput: We employ smart contracts to control access to shared data. As we all known that the block creation time is approximately 12 seconds on Ethereum. We argue that this time interval is acceptable since nodes may choose to collect a lot of updates then send requests to contracts. Usually it is not so urgent for a patient or doctor get the immediate updated shared data.

2) Correctness of smart contracts : Smart contracts might be inconsistent with specifications. We may apply some theorem prover such as Coq [21] to prove or verify the correctness of smart contracts to prevent these attacks.

3) Public blockchain: Once deployed to the public Ethereum blockchain, transactions relating to our systems might not be chosen into a block by miners. So a private blockchain might be a better choice for our system.

4) Incentive: Like in [22], we don't include any incentive for mining beyond the use of our system. We presume that all nodes on the blockchain already have incentives to keep medical data from being tampered and illegal access or updates.

V. Related work

In this section we review existing blockchain-based research on medical data sharing field and state advantages of our system compared with them.

Zyskind et al. suggested using blockchain for access control in [23] where encrypted data reside on the third party storage. But data might be exposed by this "trusted" third party so that data privacy are violated.

The idea of introducing Blockchain technology to healthcare was presented firstly in [24] where they use blockchain for data storage to guarantee medical data can not be modified by anyone. Also, they designed a Healthcare Data Gateway (HDG) to control access of the shared data. However, medical data size can become huge so that become a burden for blockchain nodes' storage since each node have the same copy of blockchain. Usually, the size of metadata is smaller than data. (It also depends on the the structure of metadata and data.) We store metadata on smart contracts so as to reduce the storage pressure for each blockchain node.

MedRec [8] choose to store raw medical data on providers' database and patients can download the data from it after authorized by smart contract on blockchain. They aimed to enable patient to engage in their healthcare. Whereas in our system, all parties, such as doctors, patients, and researchers can benefit from sharing data with others. MedRec recognized that not all provider data such as physician intellectual property can be exposed to patients [25], [26] so that they don't claim to manage contents automatically from physician's output. In our work, instead we allow each node share a piece of medical data not total but still keep consistency between them after the updates to the shared ones. Additionally, any modifications on data shared by two nodes will not be disclosed to the third party which keep the consistency only exists in sharing peers. Moreover, since all shared data with others can be a part of each nodes' local total databases, we can decide whether one shared data have some influence on the other shared pieces and then propagate this change to the third party.

Dubovitskaya at al. gave an architecture to mange and share medical data for cancer patient care [12]. They stored encrypted categorized shared data on cloud and relating metadata in blockchain and implemented the prototype on Hyperledger [27]. The access control policy are defined in the chaincode Logic by patients. Whereas we think that each data provider not just patients can use smart contracts to encode the control policy when they deploy them to Ethereum.

Notably, previous three works and others [9], [10], [22], [28], [29] mostly targeted to the access problem on shared data but did not pay much attention to updates on the them. Additionally, they presumed that different parties can share

the same data. Unlike them, we aims to solve the updates issues on the shared data and allow one party can split total data into multiple pieces (i.e, views) which are shared with different parties but still keep consistency between source and views.

VI. Conclusion

Medical data sharing are necessary and important, which allow peers (i.e., patients, doctors, researchers and others) to contribute their knowledge to better the medical treatment. Users may have different interest on the same complete medical record. Some peers might update some values of fields in the existing data. This updates need to be propagated to sharing peers. Our architecture divides a record into pieces shared with different users separately, which can protect data privacy by limiting essential data between two peers and reduce the unrelated data interference. Any updates on data pieces can be synchronized to complete records by bidirectional transformations. Moreover, based on smart contracts on blockchain, we can promise that only authorized users can update the existing shared data and only when all peers have updated to the newest data contents they can continue the operations on shared data.

We are still developing the prototype to implement our idea. In the future, we will use the real patient data to do experiment but use some de-identification technology to protect patient data from being exposed.

Acknowledgment

The authors would like to thank all PRL lab members for their helpful advice in the paper draft. Thanks Hsiang-Shang Ko for revising the BX parts and providing valuable suggestions to our work. Thanks Van-Dang Tran's help on building the initial data distribution idea sketch. This work is partially supported by the Japan Society for the Promotion of Science (JSPS) Grant-in-Aid for Scientific Research (S) No. 17H06099.

References

[1] H. Centers for Medicare & Medicaid Services *et al.*, "Hipaa administrative simplification: standard unique health identifier for health care providers. final rule." *Federal register*, vol. 69, no. 15, p. 3433, 2004.

[2] J. Zhang, N. Xue, and X. Huang, "A secure system for pervasive social network-based healthcare," *IEEE Access*, vol. 4, pp. 9239–9250, 2016.

[3] K. B. H. Zolnierek and M. R. DiMatteo, "Physician communication and patient adherence to treatment: a meta-analysis," *Medical care*, vol. 47, no. 8, p. 826, 2009.

[4] R. L. Street Jr, G. Makoul, N. K. Arora, and R. M. Epstein, "How does communication heal? pathways linking clinician–patient communication to health outcomes," *Patient education and counseling*, vol. 74, no. 3, pp. 295–301, 2009.

[5] G. Fitzpatrick and G. Ellingsen, "A review of 25 years of cscw research in healthcare: contributions, challenges and future agendas," *Computer Supported Cooperative Work (CSCW)*, vol. 22, no. 4-6, pp. 609–665, 2013.

[6] O. of the National Coordinator for Health Information Technology, "Report to congress: Report on health information blocking," 2015.

[7] C.-F. Chung, "Using personal informatics data in collaboration among people with different expertise," Ph.D. dissertation, 2018.

[8] A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman, "Medrec: Using blockchain for medical data access and permission management," in *Open and Big Data (OBD), International Conference on*. IEEE, 2016, pp. 25–30.

[9] K. Fan, S. Wang, Y. Ren, H. Li, and Y. Yang, "Medblock: Efficient and secure medical data sharing via blockchain," *Journal of medical systems*, vol. 42, no. 8, p. 136, 2018.

[10] Q. Xia, E. B. Sifah, A. Smahi, S. Amofa, and X. Zhang, "Bbds: Blockchain-based data sharing for electronic medical records in cloud environments," *Information*, vol. 8, no. 2, p. 44, 2017.

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

[12] A. Dubovitskaya, Z. Xu, S. Ryu, M. Schumacher, and F. Wang, "Secure and trustable electronic medical records sharing using blockchain," in *AMIA Annual Symposium Proceedings*, vol. 2017. American Medical Informatics Association, 2017, p. 650.

[13] Z. Hu, H. Pacheco, and S. Fischer, "Validity checking of putback transformations in bidirectional programming." in *FM*, 2014, pp. 1–15.

[14] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger."

[15] F. Abou-Saleh, J. Cheney, J. Gibbons, J. McKinna, and P. Stevens, "Introduction to bidirectional transformations," in *Bidirectional Transformations*. Springer, 2018, pp. 1–28.

[16] F. Bancilhon and N. Spyratos, "Update semantics of relational views," *ACM Transactions on Database Systems (TODS)*, vol. 6, no. 4, pp. 557–575, 1981.

[17] K. Czarnecki, J. N. Foster, Z. Hu, R. Lämmel, A. Schürr, and J. F. Terwilliger, "Bidirectional transformations: A cross-discipline perspective," in *International Conference on Theory and Practice of Model Transformations*. Springer, 2009, pp. 260–283.

[18] H. Pacheco, Z. Hu, and S. Fischer, "Monadic combinators for putback style bidirectional programming," in *Proceedings of the acm sigplan 2014 workshop on partial evaluation and program manipulation*. ACM, 2014, pp. 39–50.

[19] H. Pacheco, T. Zan, and Z. Hu, "Biflux: A bidirectional functional update language for xml," in *Proceedings of the 16th International Symposium on Principles and Practice of Declarative Programming*. ACM, 2014, pp. 147–158.

[20] H.-S. Ko, T. Zan, and Z. Hu, "BiGUL: A formally verified core language for putback-based bidirectional programming," in *Proceedings of the 2016 ACM SIGPLAN Workshop on Partial Evaluation and Program Manipulation*, ser. PEPM '16. New York, NY, USA: ACM, 2016, pp. 61–72. [Online]. Available: http://doi.acm.org/10.1145/2847538.2847544

[21] G. Huet, G. Kahn, and C. Paulin-Mohring, "The coq proof assistant a tutorial," *Rapport Technique*, vol. 178, 2004.

[22] G. G. Dagher, J. Mohler, M. Milojkovic, and P. B. Marella, "Ancile: Privacy-preserving framework for access control and interoperability of electronic health records using blockchain technology," *Sustainable Cities and Society*, vol. 39, pp. 283–297, 2018.

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

[24] X. Yue, H. Wang, D. Jin, M. Li, and W. Jiang, "Healthcare data gateways: found healthcare intelligence on blockchain with novel privacy risk control," *Journal of medical systems*, vol. 40, no. 10, p. 218, 2016.

[25] U. D. of Health, H. Services *et al.*, "Individuals' right under hipaa to access their health information 45 cfr 164.524," 2017.

[26] C. Grossman, W. A. Goolsby, L. Olsen, and J. M. McGinnis, "Clinical data as the basic staple of health learning: creating and protecting a public good," *Washington, DC: Institute of Medicine*, 2011.

[27] Hyperledger, "Hyperledger," 2017.

[28] J. Liu, X. Li, L. Ye, H. Zhang, X. Du, and M. Guizani, "Bpds: A blockchain based privacy-preserving data sharing for electronic medical records," *arXiv preprint arXiv:1811.03223*, 2018.

[29] S. Amofa, E. B. Sifah, O.-B. Kwame, S. Abla, Q. Xia, J. C. Gee, and J. Gao, "A blockchain-based architecture framework for secure sharing of personal health data," in *2018 IEEE 20th International Conference on e-Health Networking, Applications and Services (Healthcom)*. IEEE, 2018, pp. 1–6.