Blockchain-based Bidirectional Updates on Fine-grained Medical Data

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

[1] National Institute of Informatics, Japan [2] Kyoto University, Japan
[3] SOKENDAI (The Graduate University for Advanced Studies), Japan [4] University of Tokyo, Japan

***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 should only be accessed by authorized users. Given a total medical data, users may care about parts of them and other unrelated information might interfere with 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 fined-grained pieces shared with different peers then synchronize full data and these pieces 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 enables users to trace data updates history. This paper can provide a new perspective to view full 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, blockchain, update, bidirectional transformations**

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 goes to see a doctor and often resides on the hospital's database. Medical records usually 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 is hard to make sure that all medical institutes would follow these rules because they may expose patient privacy for profit. Hence, transparency in medical data management system is important. Moreover, patients might visit different hospitals and leave their records scattered [2] in different places, so that it is hard for them to manage records efficiently. Patients should better to be provided with a platform to manage and review their historical medical data that are from different hospitals in case of exposure or being tampered. In addition to providing data to doctors, sharing medical data could also benefit other stakeholders such as researchers or policy makers. For example, researchers can identify public health risks and then develop better treatment by analyzing large-scale medical data [3].

To protect patients' data from being exposed or tampered, shared medical data could reside in encrypted formats on a trusted cloud storage server and can only be accessed by authorized users. However, in that case, centralized access control might lead to a single point of failure and become the bottleneck of the sharing system. Some decentralized medical data sharing systems [4]–[6] have been proposed to manage authentication based on blockchain [7] technology, which achieves consensus among distributed nodes. The access control logic of medical data on smart contracts [4] or Chaincode [8] can be used as a criterion to judge whether a user can be allowed to access medical data. Only a user satisfies that permission information can his access be agreed by the majority of nodes, which means he is authorized from the blockchain side.

Still, we identified a limitation on current works. Consider that a full medical record can have many attributes, while different parties may be only interested in specific parts of them. For example, given a medical record, researchers are interested in the attribute of the mechanism of medicine action, whereas patients care more about the medicine dosage standard attribute. Moreover, additional but unnecessary information might influence or even mislead users' judgment. Imagine that doctors may add some symptom description on records which might put patients in confusion and fear [9]. Besides, treatment steps are exclusive to a hospital and should not be directly accessed by other users.

To fill this gap, we propose an idea that a full medical data (i.e., data source) could be split into lots of smaller pieces (i.e., data views). From the perspective of relational databases, these kinds of smaller pieces can be seen as view tables derived by querying a few but not all attributes on the base (source) table. A user can share different fine-grained data pieces with different users based on predefined protocols. Imagine a doctor can share dosage usage with a patient and medicine mechanism with a researcher respectively. In this way, only user-concerned data are exposed to them which can avoid additional data interference and protect private proprietary data of data providers from being leaked.

However, if we adopt this idea, we have to dispose of two issues as follow.

Firstly, we need to handle the synchronization between

2473-3490/19/$31.00 ©2019 IEEE
DOI 10.1109/ICDEW.2019.00-40

IEEE computer society

source and multiple views after updates on those fine-grained views occur. Consider this scenario: a researcher updates the medicine mechanism on shared data with a doctor. Note here the shared data is actually a view from the full medical data (source) on the doctor side. Thus the doctor needs to synchronize this change on view to create a new source. To solve this problem, we apply bidirectional transformations [10] to synchronize them after updates on either one side. For example, we can invoke *put* direction of a BX program to reflect modifications on shared data in the full data and *get* to produce shared data from full data. Because different views produced from the same source might have overlapping data. After the updates on shared data are reflected to doctor's source, the doctor still has to judge whether he needs to modify the shared data with patients by reproducing a new view.

Secondly, we need to conduct access control on fine-grained view data. Existing blockchain-based access control solutions, such as [4], focused on permission control on the full record. However, we can adapt it to work with the fine-grained views. For example, we can encode permission information of views into smart contracts. Shared data management should be conducted after the peer has been authorized.

In this paper, our contributions are as follows.

1) We proposed an idea that a full medical record can be partitioned to multiple fine-grained data pieces shared with different peers and all data should reside on each user's local database.
2) We solved the data synchronization between a full record and fine-grained data views after updates on a view by bidirectional transformation technology.
3) We designed a decentralized medical data sharing architecture and applied blockchain to control permission for fine-grained data views.

The remainder is organized like this. Section II gives some preliminaries about blockchain and bidirectional transformations. Section III sketches our system design and provides an implementation architecture. Section IV discuss security threats of our system and potential countermeasures as our future work. Section V compares our work with existing ones to clarify our improvement over them. Section VI concludes and directs our future work.

II. Preliminaries

A. Blockchain

Proposed with Bitcoin [7] 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 valid historical transactions in chronologically chained blocks. The nodes which generate new blocks by solving a computational puzzle (the proof-of-work problem) are called miners.

Not only can support the platform of cryptocurrency, but blockchain can also be applied to other scenes. Ethereum [11] extends Bitcoin with a built-in Turing-Complete programming language so that one can use this scripts (i.e., Ethereum Virtual Machine (EVM) bytecodes) to write programs (i.e., smart contracts[1]) on the blockchain. For example, we can write a smart contract in Solidity[2] and then compile it 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 consist 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 to execute the programs 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 transformations

Maintaining consistency between different data representations having overlapping contents is important [12]. For example, in databases, a view table can be produced by querying a base source table; this view table can be modified, in which case we will want to "restore consistency", i.e., we need to change the source such that the modified view coincides with the result of the query on the changed source — this is the well-known view update problem [13]. To achieve this, one may consider providing two separate programs to represent the two directions to propagate updates from one side to the other. But it is hard to prove that the source and view can still be kept consistent after updates. Bidirectional transformations (BXs) were proposed [14] to solve this.

BX programs[3] can be invoked in two ways as forward and backward transformations. A forward transformation (denoted as *get*) extracts some information from the source to build an abstract view, and the backward transformation (denoted as *put*[4]) embeds information of the view back into the source and produces an updated source. This pair of transformations should satisfy the *round-tripping* laws (also referred to as *well-behaveness*) called *PutGet* and *GetPut*.

$$get(put(\mathbf{source}, \mathbf{view})) = \mathbf{view} \qquad (\mathit{PutGet})$$
$$put(\mathbf{source}, get(\mathbf{source})) = \mathbf{source} \qquad (\mathit{GetPut})$$

Intuitively, *GetPut* states that no update should be performed on the source when there is no change on the view, while *PutGet* hints that *put* should take all updates on the view into account so that the view can be regenerated from the updated source by *get*. The most distinguished point of BX is that a view can contain only a part of a source. With respect to some consistency between a source and a view, BX programs can synchronize the source and view, and their well-behavedness guarantees that the source and view are

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

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

[3]The BXs we refer to in this paper are asymmetric lenses [15], one of the synchronization models studied by the BX community.

[4]*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.

kept consistent after updates on either side. There are some languages for constructing well-behaved BX programs, such as Boomerang [16], BiGUL [17] and HOBiT [18].

III. System Architecture

In this section, we first describe a scenario of distributed medical data sharing among doctor, patient, and researcher in Section III-A and propose a system architecture to model this scenario in Section III-B. Then we present the solutions for synchronization between full medical record and views using BXs and access control on views applying blockchain in Section III-C. Our system can not only allow updates on shared data but other operations such as creating data, which are described in Section III-D and explained by a concrete case in Section III-E.

A. Scenario description

Table "Full medical records" in Figure 1 shows patients' full medical records which includes seven attributes: a0. patient ID, a1. medication name, a2. clinical Data, a3. address, a4. dosage, a5. mechanism of action, a6. mode of action. Suppose there are three users: Patient, Researcher, and Doctor. Each user only stores some attributes of full records on their local databases. For instance, Patient only keeps the attributes from a0 to a4 on Table D1 and Researcher retains attribute a1, a5 and a6 on Table D2. Doctor manages attributes a0-a2, a4, and a5 on Table D3 and generates two view tables D31 and D32 by querying special attributes from D3. D31 and D32 are used to be shared with Patient and Researcher respectively. Note that D13 and D31 are identical tables but D13 is stored in Patient's database and D31 resides in Doctor's database. Similarly, attributes a1 and a5 are shared between Researcher and Doctor, which are shown in the view Table D23 on Researcher's database and Table D32 on Doctor's storage. We note that the formats and contents of shared data are predefined by sharing peers.

B. System architecture

Our system architecture is shown in Fig. 2, which contains following components:

- *Client side* controls the interaction between users and other components.
- *Server App* acts as a mediator to interact with other components.
- *Blockchain* keeps the manage permission of shared data on smart contracts and notifies sharing peers the change on them.
- *Database manager* disposes of the synchronization between shared data and local data according to consistency logic relations. These synchronizations are implemented by executing BX programs.
- *Database*: each user has a full database and many data pieces shared with other users. The latter (seen as a view) can always be reproduced from the former (seen as a source).

Now we discuss details about this architecture.

a0. Patient ID	a1. Medication Name	a2. Clinical Data	a3. Address	a4. Dosage
188	Ibuprofen	CliD1	Sapporo	one tablet every 4h

D1 (Patient)

a0. Patient ID	a1. Medication Name	a2. Clinical Data	a5. Mechanism of Action	a4. Dosage
188	Ibuprofen	CliD1	MeA1	one tablet every 4h
189	Wellbutrin	CliD2	MeA2	100 mg twice daily

D3 (Doctor)

a0. Patient ID	a1. Medication Name	a2. Clinical Data	a4. Dosage
188	Ibuprofen	CliD1	one tablet every 4h

D13 (also D31)

a1. Medication Name	a5. Mechanism of Action	a6. Mode of Action
Ibuprofen	MeA1	MoA1
Wellbutrin	MeA2	MoA2

D2 (Researcher)

a1. Medication Name	a5. Mechanism of Action
Ibuprofen	MeA1
Wellbutrin	MeA2

D23 (also D32)

a0. Patient ID	a1. Medication Name	a2. Clinical Data	a3. Address	a4. Dosage	a5. Mechanism of Action	a6. Mode of Action
188	Ibuprofen	CliD1	Sapporo	one tablet every 4h	MeA1	MoA1
189	Wellbutrin	CliD2	Osaka	100 mg twice daily	MeA2	MoA2

Full medical records

Fig. 1. Data distribution

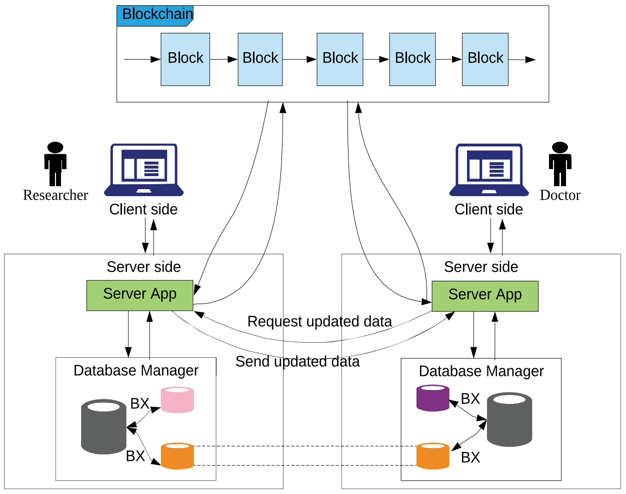

Fig. 2. System Architecture

Firstly, medical data always stay in each peer's local database and data transfer only exists between sharing peers, which avoid data being leaked to the third party so as to keep shared data security. The data can not only be provided by doctors. Instead, each node can be a shared data provider. As referred in [19], 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.

Moreover, blockchain's consensus protocol scheme keeps the shared data between sharing peers the same after updates since each peer will receive the notification from contracts and request new shared data from other sharing peers. Additionally, any modification on shared data can be recorded on the blockchain. Blockchain properties such as immutability, auditablility, and transparency enable nodes to check and review update history on shared data. Still, simultaneously updates to the same shared data by multiple peers are forbidden. Smart contracts dispose of 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 the newest shared data can they execute further operations.

C. System design

1) Synchronization between source and views: As we said before, BX programs are used to synchronize the full data and shared data. Shared data can be seen as views which can be produced from full records named as sources. For example in Fig. 1, Table D13 is shared by Patient and Doctor and can be produced from D1 by BX_{13}-*get* (i.e., applying the get direction of the BX program between D1 and D13). If D13 is modified, then D1 need to be updated from original D1 and D13 by using BX_{13}-*put*(i.e., invoking put the direction of the BX program between D1 and D13) to ensure that the modified D13 can be regenerated from the updated D1.

In our design, shared data between any two peers are not exposed to the third party, which can keep data privacy between them to some degree. For example, any operations on D23 or D32 can only be known by Researcher and Doctor and Patient has no information about this. However, If D31 and D32 have some overlapping data, after D32 is modified, D31 might need to be regenerated by using BX_{31}-*get* on modified D3 which can be produced by applying BX_{32}-*put*.

2) Access control on views: Figure 3 presents a metadata collection table which dictates the update permission on each attribute of the shared data. These kinds of tables reside in smart contracts on the blockchain. Each metadata entry corresponds to a shared table. For example, the entry for D13 or D31 declares that it is shared by Patient and Doctor and Doctor can update all attributes value but Patient can only change the clinical data. The "Latest Update Time" shows when the metadata was modified most recently. The value on the attribute "Authority to Change Permission" indicates the user who can modify other users' permission. For instance, because D13 and D31 are initially produced by Doctor and shared with Patient and then Doctor can change Patient's access permission. Doctor can change the permission for updating "Dosage" from "Doctor" to "Doctor, Patient" so that Patient can also update the "Dosage" later.

Metadata ID	Sharing peers	Write permission			Last Update Time	Authority to change permission
D13 & D31	Patient, Doctor	Medication Name	Dosage	Clinical Data	2018.12.22	Doctor
		Doctor	Doctor	Patient, Doctor		
D23 & D32	Doctor, Researcher	Medication Name	Mechanism of Action		2018.12.23	Researcher
		Doctor, Researcher	Researcher			

Fig. 3. Metadata collection in smart contract

If users want to share data, they need to form an agreement on the structure of the shared table and register the corresponding metadata on smart contracts. Suppose Doctor initiates the data sharing with Patient. According to their agreement, he will deploy a smart contract on blockchain which stipulates the metadata about the shared data, such as sharing peers (i.e., Patient and Doctor) and so on.

D. Data Management

In Fig. 4, we briefly sketch the procedures for CRUD (i.e., Create, Read, Update, Delete) operations on shared data considering entry level or table level. For Read operation, since shared data stay in users' local databases, they can just execute a query to get the shared data. We describe the workflow for updating an item on Section III-E.

	Entry Level/ Table Level
Create, Update, Delete	1. A user tries to execute operation locally and send a request (a transaction) to smart contract. 2. Smart contracts verify permission. 3. If this user is authorized then execute following steps. (If permission denied, then this request failed) 4. Smart contracts notify sharing peers of modification. 5. Sharing peers fetch this update on shared data. 6. Update metadata on contract. 7. Sharing peers conduct BX programs to reflect the change on shared data in complete data.
Read	Query local database directly

Fig. 4. CRUD operations on shared data

E. Case analysis for updating shared data

Figure 5 depicts a scenario where the researcher initiates the update the shared data. We use notations in Fig. 1. The numbers indicate the corresponding operations sequence. The red and blue circles indicate the updated places. Steps 1 - 5 and Steps 7 - 11 perform procedures for an operation on Fig. 4. Notably, Step 6 checks whether D32 and D31 have some dependencies since they might have some overlapping data. Our work leaves the initialization of shared data to future work and we only consider management on existing shared data.

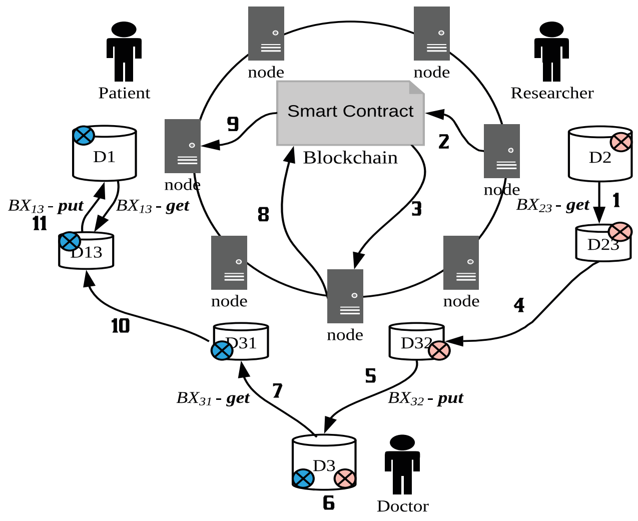

Fig. 5. A workflow for updating data fields of shared data

Let us try to describe this workflow using the data in Fig. 1. After updating the "MeA1" on D2, the Researcher wants to propagate the update to the shared data D23 so he uses the BX_{23}-*get* to regenerate D23 (Step 1). Then he will call a smart contract via a trusted node connected to blockchain by sending the request for updates to the D23 (Step 2). Note that the smart contract in Fig. 3 records all permission info about that D23. The smart contract will be executed until all nodes form a consensus on this update request, which means Researcher is permitted to update D23. Each node will conduct the smart contract locally. The entry related D23 & D32 of the contract is modified and the Doctor will receive notification that D32 needs to be modified (Step 3). Then he will ask for data from Researcher by sending a data request message and use the latest shared data to refresh D32 (Step 4). After that, Doctor will use BX_{32}-*put* to reflect the change on D32 in D3, i.e., update the "MeA1" to a new name (Step 5). Since Doctor shared some data (D31) with Patient, he needs to check whether D31 needs to be reproduced (Step 6). (If there is no need to reproduce, Step 6 - 11 will not be performed.) For example, Doctor may want to modify the "Dosage" on D31. 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 a notification about the change on "Dosage" (D31) (Step 9) and ask Doctor to send the updated D31. After Patient gets the modified D31 (Step 10), he will use this to update D1 via BX_{13}-*put* (Step 11).

IV. Threats and Countermeasures

In this section, we identify some threats to our system and propose relating countermeasures.

1) Throughput: We might employ smart contracts to control access to shared data. Usually, 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 and then send requests to contracts. It is not so urgent for a patient or doctor to 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 [20] to verify the correctness of smart contracts to prevent these attacks.

3) Public blockchain: Once deployed to the public Ethereum blockchain, transactions related 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 [21], we do not 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 illegal access or updates.

V. Related Work

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

The idea of introducing Blockchain technology to healthcare was presented firstly in [22] where they use blockchain for data storage to guarantee that medical data cannot be modified by anyone. Also, they designed a Healthcare Data Gateway (HDG) to control access to the shared data. However, the medical data size can become huge so that the data become burdens for blockchain nodes' storage since each node has the same copy of blockchain. Usually, the size of metadata is smaller than data. (It also depends on the structure of metadata and data.) We store metadata on smart contracts so as to reduce the storage pressure for each blockchain node.

MedRec [4] chose to store raw medical data on providers' database and patients can download the data from it after authorized by smart contracts on the blockchain. They aimed to enable patients 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 [23], [24] so that they do not claim to manage contents automatically from physician's output. Instead, we allow each node to 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 keeps 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 et al. gave an architecture for managing and sharing medical data for cancer patient care [8]. They stored encrypted categorized shared data on the cloud and related metadata in blockchain and implemented the prototype on Hyperledger [25]. The access control policy is 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 blockchain.

Notably, these three solutions and others [5], [6], [21], [26], [27] mostly targeted the access problem on shared data but did not pay much attention to updates on them. Additionally, they presumed that different parties can share the same data. Unlike them, we aim to solve the update issues on the shared data and allow one party to split full data (i.e., source) 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 is necessary and important, which allows stakeholders on medical scenarios to contribute their knowledge to better the medical treatment. Users may have different interests in the same full medical record. Some peers might update some values of fields in the existing data. These updates need to be propagated to sharing peers. Our architecture divides a record into pieces that 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 full records by bidirectional transformations. Moreover, based on smart contracts on the blockchain, we can promise that only authorized users can update the existing shared data and only when all peers have updated to the latest data contents can they continue the operations on shared data.

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

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

[4] 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.

[5] 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.

[6] 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.

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

[8] 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.

[9] T. Delbanco, J. Walker, J. D. Darer, J. G. Elmore, H. J. Feldman, S. G. Leveille, J. D. Ralston, S. E. Ross, E. Vodicka, and V. D. Weber, "Open notes: doctors and patients signing on," *Annals of internal medicine*, vol. 153, no. 2, pp. 121–125, 2010.

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

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

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

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

[14] 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.

[15] J. N. Foster, M. B. Greenwald, J. T. Moore, B. C. Pierce, and A. Schmitt, "Combinators for bidirectional tree transformations: A linguistic approach to the view-update problem," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 29, no. 3, p. 17, 2007.

[16] A. Bohannon, J. N. Foster, B. C. Pierce, A. Pilkiewicz, and A. Schmitt, "Boomerang: resourceful lenses for string data," in *ACM SIGPLAN Notices*, vol. 43, no. 1. ACM, 2008, pp. 407–419.

[17] 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

[18] K. Matsuda and M. Wang, "Hobit: Programming lenses without using lens combinators," in *European Symposium on Programming*. Springer, 2018, pp. 31–59.

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

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

[21] 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.

[22] 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.

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

[24] 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.

[25] Hyperledger, "Hyperledger," 2017.

[26] 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.

[27] 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.