DeepLinQ: Distributed Multi-Layer Ledgers for Privacy-Preserving Data Sharing

Shih-Wei Liao*, Edward Y. Chang†, Chun-Ting Liu†, Wei-Chen Lin*
Pin-Wei Liao*, Wei-Kang Fu*, Chung-Huan Mei† and Emily J. Chang†
*Department of Computer Science & Information Engineering
National Taiwan University
liao@csie.ntu.edu.tw
†Research & Healthcare, HTC DeepQ
edward_chang@htc.com

***Abstract*—This paper presents requirements to DeepLinQ and its architecture. DeepLinQ proposes a multi-layer blockchain architecture to improve flexibility, accountability, and scalability through on-demand queries, proxy appointment, subgroup signatures, granular access control, and smart contracts in order to support privacy-preserving distributed data sharing. In this data-driven AI era where big data is the prerequisite for training an effective deep learning model, DeepLinQ provides a trusted infrastructure to enable training data collection in a privacy-preserved way. This paper uses healthcare data sharing as an application example to illustrate the key properties and design of DeepLinQ.**

I. Introduction

It has been widely discussed that artificial intelligence can be applied to improving the quality of service in many application domains [1]. However, privacy is a major concern of an increasing number of users, which may discourage users from sharing their personal data. Since all state-of-the-art AI algorithms (e.g., deep learning and reinforcement learning) demand big data to train an effective model [2], DeepLinQ aims to support privacy-preserving data sharing through a multi-layer blockchain (to be defined shortly).

This paper specifically uses healthcare data sharing as an example to illustrate the requirements and our implementation of DeepLinQ. To permit secure and convenient sharing of Electronic Health Records (EHRs) in support of AI initiatives such as disease diagnosis and precision medication [3, 4, 5], DeepLinQ must support the following *POET* assurances:

- *Privacy compliance*. The distributed healthcare ledgers[1] (DHLs) must comply with HIPAA privacy/security rules.
- *Ownership and accessibility*. Stakeholders such as patients, hospitals, insurance companies, and governments can access DHLs only with proper permissions through a revocable contractual agreement.
- *Efficiency*. Protocols related to security, storage, validation, and access must satisfy latency and throughput requirements.
- *Transparency and accountability*. All data and their accesses are logged and immutable so that compliance audits can be performed.

DHL differs from blockchain, or more generally distributed ledgers, in that DHL must safeguard privacy and inter-operate with existing centralized databases managed by hospitals. The debate between centralized versus decentralized databases does not exist in the healthcare industry as EHRs have already been centralized. As competitors, the "owners" of EHRs may not trust one another. DeepLinQ intends to design a distributed model with proper degrees of centralization (decentralized and centralized layers can inter-operate in our multi-layer design) and access control, with *smart contracts* [7] that suit all stakeholders' requirements.

One contentious debate is that the current bitcoin blockchain is entirely decentralized, but DeepLinQ is constrained by centralized databases. Another well known issue is that the bitcoin blockchain suffers from low *transactions per second* (tps) due to its computationally intensive proof-of-work protocol, whereas general applications demand much higher tps in the order of thousands. The third issue is that the original spirit of bitcoin is to keep users anonymous, but to improve tps, DeepLinQ may want to introduce the concept of a jury or validation committee to expedite new block creation and validation. To address theses concerns, DeepLinQ employs a two-layer (or multi-layer) blockchain architecture. DeepLinQ's *base-layer* blockchain may maintain the current properties of bitcoin blockchain, whereas its *branch layer* (or branch layers) employs designs and features that can meet the aforementioned requirements.

DeepLinQ enables the POET properties with five design considerations.

1) *Shared ledgers among silo systems*: Since all healthcare providers must be HIPAA-compliant [8], their database systems will continue to be trusted silos. DeepLinQ inter-operates with HIPAA-compliant silos through *smart contracts*. Once a contract has been validated, the contract is executed to fetch EHRs. A limited number of *trusted actors and validators* would jointly validate a given contract.

[1]Terms such as blockchain and distributed ledgers are used with different definitions in different contexts [6]. We use the term *distributed healthcare ledgers* to refer to the ledgers used in the healthcare domain.

2) *Storage*: DeepLinQ does not physically store EHRs. Each DeepLinQ ledger stores smart contracts to access EHRs stored in hospitals.
3) *Smart contracts*: Interactions between stakeholders would be coordinated by smart contracts (cryptocontracts), which are computer programs controlling data flow under certain conditions. For instance, a cryptocontract written in code could enforce which EHRs can be released to whom and under what permissions and terms. Once a contractual term has expired, the access to the document should be revoked. However, unlike currencies, a document, once accessed, may be copied or distributed and its integrity compromised. To avoid this, DeepLinQ embeds the contract signature via *digital watermarking*[2] on the document. The structure of smart contracts is discussed in Section III.
4) *Data anonymity*: Since an EHR may be unlawfully copied and distributed, DeepLinQ must ensure that protected healthcare information (PHI) will not be released even if a DHL is compromised. This is achieved by preventing the association of any PHI with EHRs to be written onto DHLs. Since each hospital must be HIPAA-compliant, DeepLinQ can assume that no PHI would be handed over from hospitals to write onto DHLsand DeepLinQ itself does not write PHI onto DHLs.
5) *Multi-layer Blockchain*: Whereas a branch-layer blockchain can implement specialized protocols to meet specific requirements (e.g., a DeepLinQ branch can implement features to meet the POET properties), the base-layer blockchain serves all branches to validate their tokens. Section II details this multi-layer design.

We enumerate some typical workflows that DeepLinQ supports, and use them to understand issues of being HIPAA compliant and test the adequacy of the aforementioned design:

- Workflow #1: Patients' distributed access to EHRs. Patients who visit different providers would like to access their distributed EHRs. A contract between patient p and hospital i, $i = 1, \ldots, n$ is written to a ledger. Once the ledger has been verified and approved, patient p can use the contract to access her EHRs. Later when the patient visits hospital $n + 1$, a revised contract replaces the previous contract.
  - *Privacy note*: A smart contract must not be used to infer the PHI of patient p.
  - *Ownership note*: An EHR transfer between hospitals must go through the patient. A direct EHR transfer between hospitals is prohibited.
  - *Validation note*: Validation can be done by majority vote of a committee (i.e., via subgroup signatures) to expedite the process [10].
  - *Fraud prevention*: We were informed that EHRs were modified by providers or patients to claim for higher insurance payments. DeepLinQ can detect any unlawful changes to an EHR.
- Workflow #2: Urgent care access on behalf of a patient. A hospital or relative may want to access a patient's full EHRs when the patient is incapacitated (e.g., due to severe medical conditions). A power-of-attorney to grant access can be written as a smart contract. Any proxy on the power-of-attorney can use his/her private key to access the patient's EHRs.
  - *Security note*: If such a contract has been executed, DeepLinQ records an alert on a ledger and all parties on the contract are notified.
- Workflow #3: Granular access. Research institutes may want to collect medical images to conduct research. An insurance company may want to validate a payment. Such accesses can be enforced via a contract with granular access control (GAC). DeepLinQ permits a GAC level to be specified in a smart contract. GAC levels are carefully reviewed and each maps to a protection mask (the same as what [11] proposes) to perform access control.
- Workflow #4: Transaction fees and bidding. Unlike on the base-layer blockchain, the validation performed on DeepLinQ's branch-layer blockchain employs a trust committee. The trust committee can also include patients. A patient can be given incentives to validate smart contracts and/or share his/her EHRs at a proper GAC level with a transaction fee. Such rewards can be written into a smart contract so that actions such as bidding can be supported.

The rest of this paper is organized into four sections. Section II describes the architecture and key components of DeepLinQ. Section III presents DeepLinQ's smart contracts and examples. Section IV discusses consensus protocols. Section V offers our concluding remarks.

II. DeepLinQ Architecture & Components

DeepLinQ employs a multi-layer blockchain[3] The CAP theorem [13] motivates the need for multiple layers. Based on CAP, a distributed system can achieve at most two out of the three properties: consistency, availability, and partition tolerance. The design of a decentralized blockchain can be either AP (availability and partition tolerance) or CP (consistency and partition tolerance), given that decentralization requires being partition tolerant. Bitcoin prioritizes AP over consistency. (An AP blockchain may still be able to guarantee eventual consistency.) A transaction is validated and some bitcoins become available only after a consensus has been reached. As such, a client is advised to wait for six more blocks to be created before considering the transaction valid. This waiting time is the reason for long latency.

In the healthcare domain, since centralized databases can be partition intolerant (i.e., the access to the databases is denied when a partition takes place) both consistency and availability

[2]Watermarks can be attacked [9]. Ensuring that a watermarking scheme is robust in probability is beyond the scope of this paper.

[3]The Multi-layer Blockchain Architecture (MOAC) [12] was proposed at about the same time as this work started.

can be simultaneously ensured. Therefore, the validation or consensus protocol can be light weight to support high transaction throughput, measured in the number of transactions per second (tps).

The current DeepLinQ prototype employs two blockchain layers, where the base layer preserves AP and the branch layer ensures AC. The base layer serves as the validator of the branch-layers' ledgers, whereas a branch layer implements the features to support the POET properties (note that a base layer can serve multiple branch layers with different implementations).

The base layer that we currently use is Ethereum (please refer to the Bitcoin and Ethereum white papers [7, 14] for details). DeepLinQ's branch layer consists of a ledger with some special design considerations.

A. Branch Layer Design

For the privacy requirement, DeepLinQ's branch layer blockchain dials back the full transparency[4] of bitcoin and instead has the following three choices in DHLs: *adding trusted senior validators*, *using subgroup signatures*, *creating trusted branches*, and *employing efficient consensus protocols*.

- *Adding trusted senior validators*. A group of trusted users establish a committee to grant permissions. Only a limited number of trusted users operate by the principle of universal data diffusion. If the number can be kept small (on the order of hundreds), data is replicated fewer times and scalability is improved.
- *Using subgroup signatures*. The trusted committee can act to permit data access to EHRs via multi-signature as long as a subset of the committee can validate a transaction and grant access.
- *Creating trusted branches*. To mitigate the problem that everyone has to process everything, branches can be created for specialized purposes. Since most EHR accesses may only involve some regional hospitals, each hospital group (e.g., a hospital and its affiliates) may be able to handle its channels and branch ledgers. When a set of transactions involve some fixed set of actors, using branch ledgers improves both privacy and scalability [16]. A branch ledger can limit who receives a transaction into that particular branch ledger. Thus, it requires consensus (discussed next) only on the state of that branch ledger to the parties within a branch.
- *Employing efficient consensus protocols*. Table I lists consensus mechanisms. To fulfill our performance objectives of high throughput and low latency, we will experiment with both FBA [17] and hashgraph [18]. (Our preliminary implementation of hashgraph has achieved more than $3,000$ tps.)

B. Transactions

DeepLinQ defines a transaction to be an authorized attempt. A transaction consists of three fields:

[4]Anyone with access to the Internet has access to every transaction that has ever taken place. [15] calls this model universal data diffusion.

TABLE I
CONSENSUS MECHANISMS.

Mechanisms	Decentralized	Hi TPS	Trust	Security
Proof of Work	✓			✓
Proof of Stake	✓	maybe		maybe
Byzantine FT		✓	✓	✓
FBA	✓	✓	✓	✓
Hashgraph	✓	✓	✓	✓

- Initiators. The parties involve in the transaction. Initiators are identified by their public keys.
- Actions. A cryptographically signed smart contract, which is a piece of code to carry out actions.
- Incentives. DeepLinQ is flexible and can incorporate various incentive models.

Section I presents some representative workflows. Their involved initiators and actions (verbs) are as follows:

- Data owners (patients and hospitals). Data owners know when and what data are being collected and used. Owners have granular control over how their data can be accessed. Using the notation by [11], data owners can assign access privileges on a piece of data via T_{access} transaction.
- Data guests (owners, government, insurance, and e.g., research institutes). All accesses (both read and write) must be permitted and logged. A guest can issue a query via T_{data} transaction.

C. Off-ledger Storage

In the work of [11], an off-chain storage is used to store data, and a pointer to the data is kept on the public ledger. DeepLinQ can regard all hospital databases as off-chain storage and use smart contracts to access the off-chain storage. Once a contract has been executed and data has been fetched, the data can be stored in a secured wallet.

D. Smart Contracts

In contrast to the stateless model of bitcoin, DeepLinQ's stateful model permits participants to create any functionality directly on a DHL. We detail DeepLinQ's smart contracts in the next section.

III. SMART CONTRACTS

Smart contracts are programs that exist on the blockchain network [7]. Once a contract has been deployed, it is given an account (address) to record its binary code and cannot be modified. An authorized user can invoke contracts with an incentive (e.g., known as gas in Ethereum) for consuming the computing power and storage of the network.

By interacting with smart contracts in DeepLinQ, patients can prove the existence of their own EHRs and manage the relationship with hospitals or third parties. The functionalities are constructed by three types of smart contracts, *entity contracts (EC)*, *relationship contracts (RLC)*, and *data contracts (DC)*. The concept of these contracts is motivated by MedRec [19], but DeepLinQ makes the contract organization more

general—entity, relationship, and data—which can be used beyond just supporting the healthcare domain.

- *Entity Contract* (EC). There are two types of EC contracts, which are patient EC and hospital EC. A patient EC records the patient's encrypted data, which includes a pre-processed ID (the hash value of the citizen ID and patient name), the patient's DeepLinQ address, and relevant RLC addresses. A hospital EC records the hospital's name, its RLC address, and the relevant ID that is given by the government institution for identifying that hospital. Patients can check a hospital's identity by calling an EC's smart contract methods, and vice versa. In order to guard against malicious users, all participants are certificated by the system administrator.
- *Relationship Contract* (RLC). There are two types of RLC, patient and hospital RLCs. No Matter what type an RLC is, it records medical data contract (MDC) addresses that it owns. A hospital can send an MDC to a patient; likewise, a patient can share their medical records with selected hospitals.
- *Data Contract* (DC). A DC, or in the healthcare domain, a medical data contract (MDC) represents data stored in off-ledger storage. An MDC records metadata including hospital ID, division, doctor name, time stamp, database name (for binding and access), and the fingerprint of the record. Authorized users of an MDC can access EHRs by executing the database-access code written in the MDC.

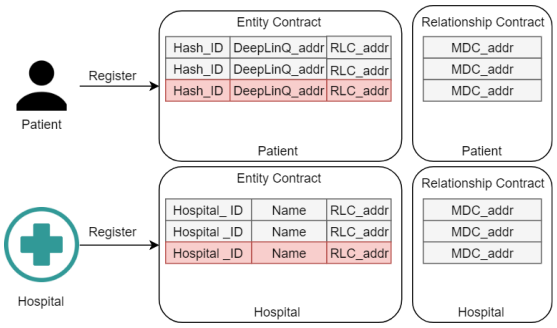
A. Access Control

The most distinct feature that differentiates DeepLinQ from MedRec is that we define two roles, patient and hospital, of EC and RLC to achieve role-based access control such that transactions are more efficient and patients are better able to ensure ownership of data. We do not allow MDCs to be sent directly between hospitals. Any such an exchange must go through the data owner, the patient. This differentiator is crucial not only because of efficiency (no need to deal with the potentially complicated $N \times N$ hospital relationships) but also because the core of privacy lies on the fact that a patient owns their EHRs.

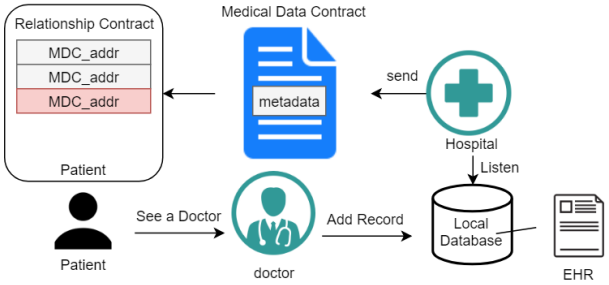
B. Contract Architecture

To explain how smart contracts work together, our current prototype has implemented three workflows, *registration*, *EHR creation*, and *EHR transfer*. DeepLinQ is flexible in adding new actors or entities and supports new relationships and workflows. We present the three prototype workflows as follows:

- Patient & hospital registration. All participants, including patients and hospitals, have to apply for a key pair from the administrator (CDC). The administrator is responsible for logging the information on patient or hospital ECs and generating relevant RLCs for the participants. When an RLC is created between a patient and a hospital, the access control on the EHR access contract is established.



- EHR creation. After a diagnosis session with a patient, the doctor adds a new medical record with pertinent lab tests, diagnoses, and prescriptions for the patient to the hospital's local database. When the hospital server is notified with the new EHR, it creates an MDC and sends the MDC to the patient-hospital RLC address, which can be mapped by checking EC with the hash value of the patient's private data. Once the transaction is on chain, the patient is notified.



- EHR transfer. If a patient wants to see a doctor at another hospital (e.g., hospital B), she can share her EHRs in hospital A with hospital B by adding hospital B to the viewer list of the MDC. Subsequently, the contract will send the MDC address to hospital B's RLC address, which can be obtained via the EC corresponding to the hospital ID. Once the new medical record is added to hospital B's RLC, the hospital B's server sends the request to the user. The user then forwards the query to hospital A to fetch her EHRs. After a series of verifications performed by hospital A, the user receives her EHRs and then forwards them to hospital B. DeepLinQ's access control scheme avoids the $N \times N$ deployment complexity between hospitals and enforces the notion that patients own and handle their EHRs.

IV. Consensus Protocols

As we discussed in Section I, DeepLinQ designs a multi-layer blockchain to achieve decentralization as well as high throughput and low latency. The base-layer blockchain can be bitcoin or Ethereum. We use the base layer to perform batch transaction validation for the second layer(s). DeepLinQ's transactions in the second layer is organized into Merkle trees. A Merkle tree is a structure that allows for efficient and secure verification of content in a large body of data. DeepLinQ stores only the root signature of a Merkle tree in the base layer. This

root signature ensures that the entire tree stored in the second layer is immutable. If the tree has somehow been tampered, the root signature stored in the base layer would not match the tampered tree root, and therefore, validation fails. DeepLinQ's base layer design makes sure that the number of transactions involved with either e.g., bitcoin or Ethereum is light-weight. Thus, the latency/throughout issue of the base-layer blockchain is a non-issue for the latency/throughput of the overall ledger system.

In the branch layers, DeepLinQ currently considers using either Federated Byzantine agreement (FBA) and hashgraph to achieve high throughput and low latency. By no means are these the only protocols that can help achieve the performance objectives. DeepLinQ looks into FBA and hashgraph because they are considered to be permissionless. This section describes both protocols, and analyzes their pros and cons.

A. FBA

In a federated Byzantine agreement system (FBA) [17], each full node maintains its own list of trust nodes, which is called a *quorum slice*. Each trusted node further maintains its own quorum slice. As stated in the white paper of Stellar [17], each node waits for the vast majority of the other nodes to agree on a transaction before considering the transaction settled. In turn, important participants do not agree to the transaction until the participants they consider important agree as well. Eventually, enough participants of the network accept a transaction, and that transaction becomes infeasible for an attacker to roll it back. Only then do all participants consider the transaction settled. Stellar claims that the FBA is permissionless. However, joining a quorum slice is by invitation only and not free, as [20] argues.

The FBAs consensus scheme can ensure the integrity of a financial network. Its decentralized control can spur organic growth. The FBA provides a flexible mechanism to let users choose their trusted parties. In the FBA's white paper [17], the hierarchical structure demonstrates that different trust tiers are played by different organizations. For example, the trusted banks can be the first tier to provide the global ledger.

The federated voting algorithm consists of two voting steps. First, each node sends a voting message to the others for a statement at time t according to their inputs and ledgers. Second, the quorum slices of node v will convince v to accept the statement at time t, if v votes contradictory statement against its quorum slices. Therefore, to guarantee liveness of node v, it needs sufficiently overlap with the quorum slices of other nodes [20]. (FBA can trade liveness for safety and vice versa.) But the problem is that it is hard to maintain v's quorum slices since there may exist a different threshold for each group that v connects to. According to [20], the FBA avoids such problems by forming a hierarchy of quorum slices and forcing each to overlap with slices at the top level as originally proposed by [21].

B. Hashgraph

The rudimentary data element of hashgraph [22] is called *gossip event* or event. An event is created and signed by a node v and it contains a timestamp, transactions (zero, one or multiple), and two hash pointers where one points to v's last event (self-parent), and the other points to the last event of the node that v syncs with (other-parent). Hash pointers can be compared to block hash in blockchain; thus, hashgraphs can always grow in one direction with an immutable order.

The hashgraph consensus protocol consists of three steps: syncing, voting, and time-stamping. At the syncing step, which is called gossip about gossip [22], each node may randomly choose a node to sync information with. A node and its chosen node exchange events that are unknown to each other which include the newest event created by the choosing node.

Each node records and validates the communication history with the other nodes. In essence, hashgraph records the entire communication history between the nodes. At a given time, two nodes may contain different subsets. However, two nodes enjoy consistency if their subsets both contain an event e and all ancestors of e. In other words, as [22] states, "as the hashgraph grows upward, the different members may have slightly different subsets of the new events near the top, but they will quickly converge to having exactly the same events lower down in the hashgraph." In the syncing step, the number of messages sent could be substantial (quadratic of the number of nodes if every node sends to all the other nodes) but the amount of data sent (the size of an event) can be quite small.

After a node has synced with the other nodes, it enters the voting step. In the hashgraph consensus protocol, nodes do not exchange messages. Instead, they decide which events are going to be confirmed by conducting virtual voting.

Virtual voting can be divided into two steps described as follows:

- *Decision round*. Each event is assigned a round number. The first events by each node in each round are witnesses. If an event e can *strongly see* more than $2/3$ witnesses, its round number is equal to $r+1$ where r is the maximum of the round numbers of e's self-parent and other-parent; else its round number is equal to r. An event e can *strongly see* a witness w if there exists a path or multiple paths from e to w going through $2/3$ of nodes.
- *Decide famous witness*. The witnesses in round r will vote for each witness in $r-1$ to decide whether each of them is famous. A witness of node i in $r-1$ round, written w^i_{r-1} ,will get a *Yes* vote if a witness in the next round w^j_r is a descendant of w^i_{r-1}. Here, *Yes* means w^j_r votes w^i_{r-1} is famous. Each witness in round $r+1$, w^k_{r+1}, will collect ballots from witnesses in r that can be strongly see by w^k_{r+1}. If $2/3$ nodes vote *Yes* for w^i_{r-1}, w^i_{r-1} is a famous witness in round $r-1$. However, if less than $2/3$ nodes vote for the same result, w^k_{r+1} will vote as same as the majority and postpone the decision to round $r+2$ and so on. However, if the following witnesses still hesitate, the decision process can fall into an infinite

loop. Therefore, in order to prevent this situation, every 10^{th} round of voting, the witness votes pseudo-randomly instead of voting the same as the majority.

If an event e is an ancestor of all of the famous witnesses in round $r-1$, we regard e as confirmed and will give e a timestamp. To calculate e's timestamp, we find a list l where each element in l is the timestamp of e's earliest descendant by node i which i has a famous witness in round $r-1$. e's timestamp is the median of l. Finally, through the three aforementioned steps, every honest node can reach a consensus in the total order of events through the aforementioned steps.

However, the hashgraph consensus algorithm still faces limitations. First, although the voting step costs no network resources, bandwidth is still a concern for the gossip about gossip protocol. The priority of a transaction can be affected by network bandwidth, i.e., the larger the bandwidth is, the faster a transaction can be confirmed [22]. Second, the hashgraph consensus algorithm is more suitable for operating on a private or permissioned chain since the network needs full awareness of all authoritative participants [23].

V. Conclusion

We propose DeepLinQ, a multi-layer blockchain for privacy-preserving data sharing, which is essential to alleviate privacy concerns of users to share their data. Though we use healthcare to illustrate DeepLinQ's design requirements and implementation, DeepLinQ can be deployed in other application domains with domain-specific branch layer implementation. We are currently working with partner universities and hospitals to deploy DeepLinQ and to enhance its cryptography modules to support flexible multi-signature and other schemes.

We believe DeepLinQ's multiple layer design to be effective in satisfying the POET properties. One challenge that we found echos the findings of [20] that the claims made by some white papers require rigorous verification. We are developing a benchmark to evaluate the performance of and trade-offs between protocols.

Acknowledgements

We would like to thank Professors Monica Lam and David Mazires at Stanford University for their helpful comments toward our research. We would also like to thank Shan-Yi Yu and Yi-Fan Chung at HTC DeepQ for their managerial and engineering support.

References

[1] UK Government Chief Scientific Advisor, "Distributed ledger technology: Beyond block chain," 2016.

[2] E. Y. Chang, *Foundations of Large-Scale Multimedia Information Management and Retrieval*. Springer, February 2011.

[3] H.-C. Kao, K.-F. Tang, and E. Y. Chang, "Context-aware symptom checking for disease diagnosis using hierarchical reinforcement learning," *AAAI*, 2018.

[4] Y.-S. Peng, K.-F. Tang, H.-T. Lin, and E. Y. Chang, "Refuel: Exploring sparse features in deep reinforcement learning for fast disease diagnosis," *NIPS*, 2018.

[5] E. Y. Chang, M.-H. Wu, K.-F. T. Tang, H.-C. Kao, and C.-N. Chou, "Artificial intelligence in XPRIZE DeepQ Tricorder," *Workshop on Multimedia for Personal Health and Health Care*, 2017.

[6] R. G. Brown, "On distributed databases and distributed ledgers," 2016.

[7] J. Ray, "A next-generation smart contract and decentralized application platform," *Ethereum Wiki*.

[8] U. D. of Health and H. Services, "Hipaa security rule." [Online]. Available: https://www.hhs.gov/ hipaa/ for-professionals/ security/ laws-regulations/ index.html

[9] S. Voloshynovskiy, S. Pereira, T. Pun, J. Eggers, and J. K. Su, "Attacks on digital watermarks: classification, estimation based attacks, and benchmarks," *IEEE Communications Magazine*, vol. 39, pp. 118–126, 2001.

[10] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "A survey of two signature aggregation techniques," *CryptoBytes*, vol. 6, no. 2, 2003.

[11] G. Zyskind, O. Nathan, and A. Pentland, "Decentralizing privacy: Using blockchain to protect personal data," *IEEE Security and Privacy Workshops*, pp. 180–184, 2015.

[12] MOAC, "Multi-layer blockchain architecture," 2018.

[13] E. A. Brewer, "Towards robust distributed systems," *PODC*, 2010.

[14] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008. [Online]. Available: https:// bitcoin.org/ bitcoin.pdf

[15] M. Rauchs, A. Glidden, B. Gordon, G. Pieters, M. Recanatini, F. Rostand, K. Vagneur, and B. Zhang, "Distributed ledger technology systems, a conceptual framework," *University of Cambridge Judge Business School*, 2018.

[16] A. Lewis, "A gentle introduction to blockchain technology."

[17] D. Mazires, "The stellar consensus protocol: A federated model for internet-level consensus," *Stellar Development Foundation*, 2016.

[18] L. Baird, "The swirlds hashgraph consensus algorithm: Fair, fast, byzantine fault tolerance," *Swirlds Tech Report Swirlds-TR-2016-01*, 2016.

[19] A. Ekblaw, A. Azaria, J. D. Halamka, and A. Lippman, "Medrec prototype for electronic health records and medical research data," *MIT Media Lab and Beth Israel Deaconess Medical Center*, 2016.

[20] C. Cachin and M. Vukolić, "Blockchain Consensus Protocols in the Wild," *ArXiv e-prints*, Jul. 2017.

[21] D. Malkhi, M. K. Reiter, and A. Wool, "The load and availability of byzantine quorum systems," *SIAM Journal on Computing*, vol. 29, no. 6, 2000.

[22] L. Baird, "The swirlds hashgraph consensus algorithm: Fair, fast, byzantine fault tolerance," 2016.

[23] S.-M. Choi, J. Park, Q. Nguyen, A. Cronje, K. Jang, H. Cheon, Y.-S. Han, and B.-I. Ahn, "Opera: Reasoning about continuous common knowledge in asynchronous distributed systems," *ArXiv e-prints*, October 2018.