MedRec: Blockchain for Medical Data Access, Permission Management and Trend Analysis

by

Ariel C. Ekblaw

B.S., Yale University (2014)

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning,
in partial fulfillment of the requirements for the degree of

Master of Science in Media Arts and Sciences

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2017

Author ... Signature redacted .....
Program in Media Arts and Sciences,
School of Architecture and Planning,
January 20, 2017

Certified by ... Signature redacted .....
Andrew Lippman
Associate Director and Senior Research Scientist, MIT Media Lab
Thesis Supervisor

Accepted by ... Signature redacted .....
Pattie Maes
Academic Head, Program in Media Arts and Sciences

MedRec: Blockchain for Medical Data Access, Permission Management and Trend Analysis

by

Ariel C. Ekblaw

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning,
on January 20, 2017, in partial fulfillment of the
requirements for the degree of
Master of Science in Media Arts and Sciences

Abstract

Years of heavy regulation and a long-standing focus on compliance have co-opted the ability of the healthcare industry to implement novel data sharing approaches. We now face a critical need for such innovation, as personalization and data science prompt patients to engage in the details of their healthcare and restore agency over their medical data. This thesis proposes MedRec: a novel, decentralized record management system to handle EHRs (Electronic Health Records), using blockchain technology. The system design gives patients a comprehensive, immutable log and access to their medical information across providers and treatment sites. Leveraging unique blockchain properties, MedRec manages authentication, data retrieval, update tracking for existing records, data entry (both for patients and providers) and data sharing. MedRec accomplishes record management without creating any centralized data repositories; a modular system design integrates with providers' existing, local data storage solutions, facilitating interoperable data exchange between data sources and the patients. We incentivize healthcare industry stakeholders (government-funded researchers, public health authorities, etc.) to participate in the network as blockchain "miners". This provides them with access to aggregate, anonymized data as mining rewards, in return for sustaining and securing the MedRec network via Proof of Work. We emphasize the flexibility and extensibility of our system components to other dimensions of the healthcare industry and to applications beyond healthcare as well. This thesis describes the MedRec technical design and early-stage prototype, our pilot with Beth Israel Deaconess Medical Center (BIDMC), and an analysis of MedRec's contribution in the context of national healthcare priorities. This work is supported by the MIT Media Lab Consortium.

Thesis Supervisor: Andrew Lippman
Title: Associate Director and Senior Research Scientist, MIT Media Lab

MedRec: Blockchain for Medical Data Access, Permission Management and Trend Analysis

by

Ariel C. Ekblaw

The following people served as readers for this thesis:

Signature redacted

Thesis Reader .....

Joseph Paradiso

Alexander W. Dreyfoos (1954) Professor of Media Arts and Science

MIT Media lab

Signature redacted

Thesis Reader .....

John D. Halamka, MD

Chief Information Officer

Beth Israel Deaconess Medical Center

Acknowledgments

The Media Lab offers students a distinct take on pedagogy—we are here to learn, not to "be educated". We are reminded that "learning is something you do for yourself; education is something done to you". I am grateful for this unique and engaging environment, a place of projects, peers, passion and play. Thank you, to the Lab and to the Hive, for inspiring my pursuit of learning in this Master's program.

When reflecting on the path to this thesis, I first and foremost thank my research partner in this endeavor, Asaph Azaria. It was truly a privilege to work with you through the early days of MedRec, and I have learned so much from your approach to code and system architecture design.

Thank you to my advisor, Andrew Lippman, for the incredible opportunity to join the Media Lab and to be part of the Viral Communications family. I will long be grateful for the many insights, advice and new perspectives you share with us all. Thank you to viral-grads, a troupe of the best partners in crime, co-conspirators, and friends a gal could hope for. Thank you to MAS, the ever-supportive Linda, Monica and Keira, who serve as such kind and thoughtful advocates throughout the academic program. We are all lucky to have you. Thank you to the man behind it all, Joi Ito, who keeps the Lab's mind, magic and mischief dials at full throttle! Thank you to the legions of dedicated staff in the Director's Office, NeCSys and Facilities who do so much to support our research work.

Thank you to my thesis readers, Joe Paradiso and John Halamka, MD, for your support and thoughtful feedback throughout the thesis project. To Joe, thank you for supporting the cross-lab collaboration for this project, and for encouraging me to explore creative applications of sensors and blockchains. To John, thank you for the pivotal opportunity to test our system with Beth Israel Deaconess Medical Center and for the many notes of encouragement and support as we sought to situate MedRec in a national healthcare context. To Dr. Larry Markson and the BIDMC Clinical Information Systems team, thank you for the incredible on-site support and design discussions that made the Beth Israel - MedRec pilot a success. It has been an honor working with you.

Thank you to the MIT Digital Currency initiative, and to Chelsea Barabas in particular, for the class out of which this project was born. I am indebted to Blockchain Technologies MAS.s65, both for the opportunity to build our first prototype, and for the foundational knowledge that made this thesis possible. Thank you to Thiago Vieira for your creative contributions to the MedRec smart contracts—it was a pleasure working with you as a project partner. Thank you to Brian Forde for encouraging us to believe in the product. MedRec's visibility beyond the lab is in large part due to your thoughtful introductions connecting our team to the healthcare industry. Thank you to Neha Narula for enlightening discussions about the philosophy of blockchains and distributed systems, and for encouraging me to embrace MedRec as my master's thesis.

And finally, last but certainly not least, thank you to my dear family and friends, at times across the country and across the globe. I so deeply appreciate your kindness,

friendship and knowledge. To Mom, Fritz and Ian, thank you for building and being part of a family that is ever eager to learn and improve the world—to infinity and beyond!

Contents

List of Figures

Chapter 1

Introduction

In a world with abundant access to the data that underpins our daily lives, from instant feedback on local weather to mobile banking, why do we still struggle with the inaccessibility and obscurity of our medical data? Perhaps the most promising dataset in its potential to teach us something about our lives, our medical data remains trapped in the many silos that mark our march through "healthcare". While advances in data science and the growing trend of "personalization" (from online ads to fingerprint-actuated devices) prompt patients to engage in and track the details of their healthcare, we lack the very infrastructure to supply this service across providers and treatment sites. It is time to responsibly, and thoughtfully, upgrade the foundational technologies that currently inhibit medical record accessibility; it is time to explore new technologies that might free us from some of this friction, while still preserving the critical layers of security and privacy around this domain of sensitive data. This thesis, while exploring a novel application of blockchain technology for data management in the medical context, is ultimately a story of how we can return data ownership to the data producers; of how we can restore personal agency over personal data to the tenants of the Information Age.

The story begins with an exploration of decentralization over centralization. Con-

sidering the growing societal backlash against corporate and government concentrations of personal data (often exploited by hackers and cyberattacks), we explore whether there might be a viable, decentralized alternative. Keeping track of data means keeping a ledger, not necessarily keeping all the data in one place. Can new architectures borrowed from cryptocurrency and years of distributed systems research be used to effectively manage access and permission rules across a network of disparate data sources? The MedRec system described in this thesis tests an application of distributed ledger technology, namely a blockchain architecture, against that goal. Our aim is to give users control over the flow of their sensitive information—from baked in rights to access their own data across multiple data stewards or "providers", to rights for data retrieval and local download, to facilitating data sharing and ultimately enabling trend analysis on their holistic medical dataset. To be able to do this with a decentralized, rather than centralized architecture, is the key. We strive to enable this functionality for patients without requiring that patients place trust in a large, centrally managed repository. This desire for a trustless, open (in that it is interoperable) and immutable architecture led us to the "blockchain" as a prototype foundation. From here, we consider the ways in which the MedRec blockchain architecture can address the challenges of institutional and personal data management and which other challenges it opens anew.

1.1 Problem Definition: Electronic Health Records (EHR) Access and Usability Challenges

EHRs were never designed to manage multi-institutional, lifetime medical records. Patients leave data scattered across various organizations as life events take them away from one provider to another. In doing so they lose easy access to past data, as the provider, not the patient, generally retains primary stewardship (either through

explicit legal means in over 21 states, or through default arrangements in the process of providing care) [4]. Through the HIPAA Privacy Rule, providers can take up to 60 days to respond (not necessarily to comply) to a request for updating or removing a record that was erroneously added [5]. Beyond the time delay, record maintenance can prove quite challenging to initiate as patients are rarely encouraged and seldom enabled to review their full record [4] [5]. Patients thus interact with records in a fractured manner that reflects the nature of how these records are managed.

Interoperability challenges between different provider and hospital systems pose additional barriers to effective data sharing. This lack of coordinated data management and exchange means health records are fragmented, rather than cohesive [6]. Patients and providers may face significant hurdles in initiating data retrieval and sharing due to economic incentives that encourage "health information blocking." A recent report from the Office of the National Coordinator for Health Information Technology (ONC) details several examples on this topic, namely health IT developers interfering with the flow of data by charging exorbitant prices for data exchange interfaces [7].

When designing new systems to overcome these barriers, we must prioritize patient agency. Patients benefit from a holistic, transparent picture of their medical history [6]. This proves crucial in establishing trust and continued participation in the medical system, as patients that doubt the confidentiality of their records may abstain from full, honest disclosures or even avoid treatment. In the age of online banking and social media, patients are increasingly willing, able and desirous of managing their data on the web and on the go [6]. However, proposed systems must also recognize that not all provider records can or should be made available to patients (i.e. provider psychotherapy notes, or physician intellectual property), and should remain flexible regarding such record-onboarding exceptions [8] [9].

Medical records also prove critical for research. The ONC's report emphasizes that

biomedical and public health researchers "require the ability to analyze information from many sources in order to identify public health risks, develop new treatments and cures, and enable precision medicine" [7]. Though data trickles through to researchers from clinical studies, surveys and teaching hospitals, we note a growing interest among patients, care providers and regulatory bodies to responsibly share more data, and thus enable better care for others [7] [10].[1]

1.2 Thesis Motivation

At the beginning of the MedRec project, we settled on three key motivations for our work. Our first priority was unifying patient access to health data across providers and treatment sites. This is the primary functionality that the MedRec prototype was built to offer, with a vision that providing patients clear and easy access to their data is the first step in improving their ability to make smart decisions about their care. As examined in the prior section, the current state of medical data access is quite antiquated. Records, when actively requested by the patient, are often delivered weeks later, in a format that does not easily transfer into other hospital systems (e.g. hard copy print outs, scanned files, et cetera), and can be charged for due to the costs of the record processing and physical media used in the transfer. We want to change this default: rather than having to actively request a data release and being at the whim of an incompatible transfer format, patient data should automatically be cataloged and available for syncing through a single, unified portal. We make no claims to who should own this portal—MedRec need not be the WebApp provider going forward. Our goal was to build a prototype, proof-of-concept infrastructure to enable holistic review of patient data, across time and across providers, without building a database that centralized this data and risked creating a target for content

[1] This section and several subsequent sections of this thesis are drawn from our previously published technical papers, in particular, the ONC Whitepaper submission [11]. Notations will be made throughout the thesis when other sections of this work are included in the thesis text.

attack.

The second motivation was to facilitate patient-initiated data sharing. Building on the notion of a "share button" for healthcare data and Steven Keating's inspiring story [12] of opening his brain tumor dataset, we hoped to enable a streamlined process by which patients could authorize a 3rd party to view their information. Perhaps a patient wants to seamlessly share data with a specialist to obtain a second opinion on their condition, or a grandmother wants to share her full medical data record with her children to establish a family health history. We hope that MedRec has provoked interest in and provided an early example of how these and other permissioning schemes might be accomplished through pointer-based accounting in a distributed ledger.

The third motivation was to empower researchers with anonymized, aggregated "big data" from an interoperable network of health records. To do this, we theorized an incentive system for healthcare industry stakeholders (government-funded researchers, public health authorities, etc.), where they would participate in the network as blockchain "miners". The research premise is to provide them with access to data as mining rewards, in return for their contribution of computational power to sustain and secure the MedRec network via Proof of Work. We believe this could enable census-level insights where institutions like the Centers for Disease Control and Prevention (CDC) might track flu trends across the MedRec network, or identify narcotics abuse. Pharmaceutical companies and large medical research institutions might rely on the MedRec network for longitudinal studies or clinical trial research. Critically, in all these cases, the research would be enabled via the MedRec distributed network rather than requiring a centralization of records.

A fundamental design guideline throughout our work was a focus on simplicity and modularity. We did not want to build a brand new EHR system, nor design an unnecessarily feature-rich user interface. Our recurring principle was to make

the MedRec architecture work with existing systems—to work with a wide range of string-queried databases that are already in use in hospitals and treatment sites across the U.S.; to work with physicians who didn't want to enter data through yet another interface; to work with existing data exchange protocols. We believe we have achieved this simplicity with our database gatekeeper module that sits on top of existing databases and provides for data retrieval post-approval from the blockchain authentication log. Notably, the MedRec system is not a "security" solution, in that it does not contribute to end-point security (for either the data sources or the end-user device), nor does it address the "DRM" problem [13] of unauthorized data copying. We discuss the security, privacy and interoperability implications of the MedRec system in more detail in the Evaluation. Fundamentally, the MedRec project is a backend system design and early prototype that hopes to provoke a new model for healthcare data management, via an access rights and permission verification log.

1.3 MedRec's Development Plan

This thesis evolved from a class project in "Blockchain Technologies" MAS.s65, a Fall 2015 class at the MIT Media Lab. The initial MedRec prototype grew out of a collaboration with fellow classmates Asaph Azaria (MIT Media Lab) and Thiago Vieira (MIT CSAIL), and was subsequently presented at the IEEE 2nd International Conference on Open and Big Data, listed on PubPub,[2] and featured in the ONC's August 2016 Blockchain in Healthcare whitepaper competition. With Dr. John Halamka, CIO of Beth Israel Deaconess Medical Center and his Clinical Infrastructure team, we completed a two day, on-site test with BIDMC in August 2016. This thesis centers on the MedRec system design and prototype code. As documented in our written materials and reflected in the MedRec Github repository, we have designed a

[2]PubPub is a "free and open tool for collaborative editing, instant publishing, continuous review, and grassroots journals", developed at the MIT Media Lab in the Viral Communications group.

novel system for decentralized record management and data sharing, using blockchain technology. This system is extensible beyond the medical record context, and the thesis will briefly address broader applications of our code in addition to our primary use case in healthcare.

Chapter 2

Background

2.1 Blockchain Background

2.1.1 Bitcoin

To understand the origin of and proliferation of blockchain technology, including the foundation of MedRec, we must turn to the original instantiation: Bitcoin. In 2008, Satoshi Nakamoto (a figure whose identity remains a public mystery at the time of writing) released what has become known as the Bitcoin whitepaper [1]. The system described therein creates a cryptocurrency, dubbed "bitcoin", and an immutable ledger "blockchain" (a term coined later, outside of the paper) that draws together many ideas from years of distributed systems, cryptography and digital payments research (Proof of Work [14] , Adam Back's HashCash [15] , Merkle roots [16] , et cetera). See Figure 2-1 for a visual representation of the Bitcoin blockchain structure.

The bitcoin currency has experienced significant volatility in the years since its initial release [17]. Without government backing or a coordinated monetary policy, the currency at times lacks stability. What it lacks in stability, it gains in independence and flexibility—its proponents often value the libertarian principles of unregulated money that can be spent across borders, without government interference and free

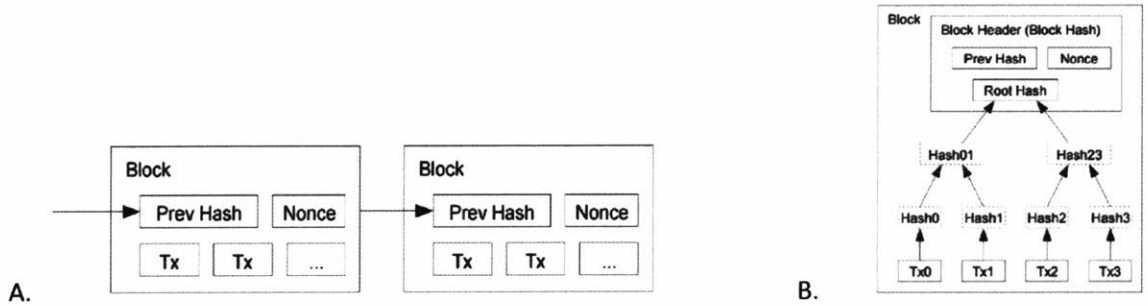


Figure 2-1: Images A and B, from the original Bitcoin whitepaper, display the "blockchain" concept of linked data, where each subsequent block includes the hash of the former. Image A shows an excerpt of such a linked chain, while image B describes the data contained within a block. [1]

from exorbitant banking fees. In the nine years since its release, Bitcoin has established a commanding presence in the digital payments space and now includes an extensive open-source code repository with volunteers working as the "core developers".

The blockchain uses public key cryptographic techniques to create an append-only, immutable, timestamped chain of content. Copies of the blockchain are distributed on each participating node in the network. The Proof of Work algorithm used to secure the content from tampering depends on a "trustless" model, where individual nodes must compete to solve computationally-intensive "puzzles" (hashing exercises) before the next block of content can be appended to the chain. These worker nodes are known as "miners," and the work required of miners to append blocks ensures that it is difficult to rewrite history on the blockchain. This "immutability" holds, provided that miners do not collude and attempt to direct collective hash rate (or mining power) at modifying a block, generally known as a "51% attack" [18].

Originally designed for keeping a financial ledger, the blockchain paradigm can be extended to provide a generalized framework for implementing decentralized compute resources [19]. Each compute resource can be thought of as a singleton state-machine that can transition between states via cryptographically-secured transactions. When

generating a new state-machine, the nodes encode logic which defines valid state transitions and upload it onto the blockchain. From there on, the blocks journal a series of valid transactions that, when incrementally executed with the state from the previous block, morph the state-machine into its current state. The Proof of Work consensus algorithm and its underlying peer-to-peer protocol secure the state-machines' state and transitioning logic from tampering, and also share this information with all nodes participating in the system. Nodes can therefore query the state-machines at any time and obtain a result which is accepted by the entire network with high certainty. [1]

This transaction-based state-machine generalization of the blockchain is informally referred to as smart contracts. Ethereum, discussed in more detail below, is the first to attempt a full implementation of this idea. It builds into the blockchain a Turing-complete instruction set to allow smart-contract programming and a storage capability to accommodate on-chain state. We regard the flexibility of its programming language as an important property in the context of EHR management. This property can enable advanced functionality (multi-party arbitration, bidding, reputation, etc.) to be coded into our proposed system, adapting to comply with differences in regulation and changes in stakeholders needs.

We utilize Ethereum's smart contracts to create intelligent representations of existing medical records that are stored within individual nodes on the network. We construct the contracts to contain metadata about the record ownership, permissions and data integrity. The blockchain transactions in our design carry cryptographically signed instructions to manage these properties. The contract's state-transition functions carry out policies, enforcing data alteration only by legitimate transactions. Such policies can be designed to implement any set of rules which govern a particular medical record, as long as it can be represented computationally. For example, a policy may enforce that separate transactions representing consent are sent from both

[1] This paragraph and the following content of this section are drawn from our previously published techincal publications [11], as explained in footnote 1.

patients and care providers, before granting viewing permissions to a third party.

2.1.2 Ethereum

Ethereum, a system first described by founder Vitalik Buterin in 2013, has been summarized as a "collection of non-localized singleton programmable data structures" [20]. This approach for encoding data on a blockchain includes two primary interaction modes: personal accounts orchestrated via public/private key pairs where the monetary token of the system (Ether) can be "held", sent and received; and "smart contracts" in a Turing complete language[2] that encode any number of real-world arrangements and require Ether to fund the activity of the smart contract. Ethereum smart contracts are "stateful", allowing updates to the status of contracts even as they are logged on a blockchain.

Unlike in Bitcoin, where the financial token is the centerpiece of the system (both in terms of accounting for its exchange between parties, and as the sole reward for computationally sustaining the network via mining), the Ether token has been described as "crypto-fuel" to run logic processes [21]. For quite some time, users were not encouraged to hold Ether as a financial investment, but rather as a tool for running Ethereum smart contracts. The Ether token, in this role, still has economic value as the means of executing code, and the ETH and ETC tokens[3] are traded on some exchanges.

Ethereum consists of a distributed system of "nodes" or processing units (personal computers, virtual machines, etc) that run one of their supported language imple-

[2]The significance of Ethereum using a Turing Complete language set for the scripting of smart contracts: a Turing-Complete programming language both enhances the flexibility of the system (compared with the very limited stack-based language of Bitcoin), but therefore also introduces a complexity for bounding the behavior of certain contracts. This has led to security vulnerabilities in prominent smart contracts on the live Ethereum network.

[3]The summer 2016 infamous DAO fork led to two competing versions of chain history, and thus two different Ethereum currencies. Ethereum "Classic" or ETC recognizes the attack on the DAO that effectively "stole" investors' money and maintains the original chain, and Ethereum or ETH revised history to a state where the funds had not been hacked.

mentations. Ethereum offers a command line interface and a series of clients through which you can manage your node, and several supported languages for writing smart contracts.[4] As an open-source project, the level of support differs for various languages, and most nodes default to the now-standard Go implementation and Geth client [22]. Ethereum includes a protocol for mining Ether (to computationally secure and sustain the network) based on Proof of Work. Their particular variant, ETHash, is memory-hard and stated by the Ethereum team to be more resistant to ASICs.[5] The Ethereum team has extensively researched a switch to Proof of Stake,[6] an alternative mining approach, though the main codebase still uses Proof of Work.

The MedRec system is architected on an Ethereum blockchain foundation. In November 2015, we forked the Ethereum Frontier release [23] (originally made public in June of that year) and modified aspects of their codebase to fit our use case scenarios. The primary components of our Ethereum test network and the modules we forked are detailed below. In our case, by forking the codebase and taking the development private again, we were not building on the live Ethereum network nor working as an application "on top of" Ethereum. MedRec is, in essence, a privatized, small-scale ethereum blockchain with extensive APIs built on top to facilitate the healthcare application. This gave us the flexibility to explore and modify blockchain parameters such as the wait time before accepting blocks, the "difficulty" of the Proof of Work algorithm, and more. Over the course of our code modifications, we discovered certain limitations in the original implementation of pyethapp (a python-based Ethereum client) and contributed patches back to the Ethereum open-source project.

Since our fork of the Ethereum Frontier repository, the Ethereum network has seen several major "hard forks" or significant, backwards-compatibility-breaking changes,

[4]Solidity is the most popular, and now primarily supported Smart Contract scripting language. Other supported languages include Serpent.

[5]"Application Specific Integrated Chips" used in this context to refer to hardware developed expressly for the purpose of solving hash functions used in Bitcoin and other cryptocurrencies.

[6]For more on Ethereum's Proof of Stake philosophy: https://github.com/ethereum/wiki/wiki/Proof-of-Stake-FAQ .

reflecting the rapidly evolving nature of blockchain architectures. This impacts our strategy in taking MedRec forward, as we would need to consider a migration to the latest Ethereum release or re-architecting on a different blockchain. I discuss these choices in greater detail in the Evaluation section.

2.2 The Evolving EHR Landscape

The other main piece of the MedRec puzzle, beyond blockchain technology, is integration with existing EHR systems. In the last several years, EPIC has gained significant market share and has become the center of debate around closed medical record systems, where the "vendor" rather than the hospital or treatment community retains certain control over the transfer and accessibility of data. Other leading EHR providers include Cerner and McKesson. Interestingly, while many hospital networks and providers accepted this intermediary role that EPIC and others provide, Beth Israel Deaconess Medical Center maintained the independence of their self-built clinical information systems, based on the InterSystems cache database [24]. BIDMC was also one of the first hospital networks to offer electronic "personal health records" to their patients, as early as 2000 [25]. Interestingly, the InterSystems database also underpins EPIC's many products [26], making their implementation an important consideration for interoperability in any future development on MedRec. That is to say, MedRec and future blockchain-based systems for healthcare would benefit from preparing for integration with cache, at least until data transfer is standardized over industry-approved APIs.

In approaching the problem of medical record fragmentation, we note a long history of related efforts, from data standards work in the 1990s on the Clinical Document Architecture [27] (as digitization of records was becoming more common) to recent tech industry efforts to unify access to data, such as Google Health (which

ceased operations in 2011) [28]. As "medical devices" proliferate in new packages, such as smartphones and wearables, we also note the rising trend in corporate managed healthcare datasets, notably Apple Health Kit [29] and Fitbit [30]. Though our MedRec prototype has only been tested with traditional record architectures (i.e. doctor-generated, stored in a hospital database), we envision a future for MedRec and EHR systems more generally, that allows inclusion of data from these non-traditional (in the healthcare context) devices. We are cognizant of the complex regulatory environment that governs medical data applications, from HIPAA to the HITECH Act, and the extensive prior policy work emphasizing interoperability [31].

2.3 Combining Healthcare and Blockchains

For a discussion of prior and related work in blockchain applications for record management, we begin by acknowledging the work of Zyskind et al. to assemble references to data and encode these as hashed pointers onto a blockchain ledger [32]. Kish proposed the blockchain for hypothetical key management in a medical context [10]. As interest in "blockchain in healthcare" has grown, we note extensive interest among the MedTech industry, pharmaceutical companies, consulting firms and others in designing and deploying blockchain applications [33]. To the best of our knowledge, we are the first to introduce a testbed prototype with our pointer architecture, blockchain smart contracts and database gatekeeper, the first to propose and implement a model for medical researchers as blockchain miners, and the first to propose that such a blockchain prototype be designed as an open API architecture for an interoperable health IT stack. Through our many meetings with the healthcare and tech industry over the past nine months we have examined why certain efforts have failed (e.g. Google Health [28], healthcare data exchange standards discarded over the years by the HL7 community, et cetera) and which others are on the rise (e.g. personal

healthcare aggregators [34]) in improving the usability and trend analysis potential of health data. We believe the blockchain architecture of distributed data offers a fundamentally new and promising approach to the problem, avoiding the pitfalls of centralized data repositories, the creation of unnecessary intermediaries and forced data migrations.

Chapter 3

MedRec System Design and Implementation

3.1 MedRec Design Overview

Our MedRec blockchain implementation addresses four major issues highlighted in the introduction: fragmented, slow access to medical data; lack of system interoperability; lack of patient agency over healthcare data; the need for improved data quality and quantity for medical research. We build on the work of Zyskind et al. [32] to assemble references to data and encode these as hashed pointers onto a blockchain ledger. We then propose to organize these references to explicitly create an accessible bread crumb trail for medical history, without storing raw medical data on the blockchain. Our system design supplements these pointers with on-chain permissioning and data integrity logic, empowering individuals with record authenticity, auditability and data sharing. We design modular APIs to integrate with existing provider databases for interoperability. A novel data-mining scheme is proposed to sustain the MedRec network and bring open, big data to medical researchers. We present MedRec not as the panacea for medical record management, but as a foray into this space to

demonstrate innovative EHR solutions with blockchain technology.

For MedRec, the block content represents data ownership and viewership permissions shared by members of a private, peer-to-peer network. Blockchain technology supports the use of "smart contracts," which allow us to automate and track certain state transitions (such as a change in viewership rights, or the birth of a new record in the system). Via smart contracts on an Ethereum blockchain, we log patient-provider relationships that associate a medical record with viewing permissions and data retrieval instructions (essentially data pointers) for execution on external databases. We design the system to include on the blockchain a cryptographic hash of the record to ensure against tampering, thus tracking data integrity. Providers can add a new record associated with a particular patient, and patients can authorize sharing of records between providers. In both cases, the party receiving new information receives an automated notification and can verify the proposed record before accepting or rejecting the data. This keeps participants informed and engaged in the evolution of their records.

MedRec prioritizes usability by also offering a designated contract which aggregates references to all of a user's patient-provider relationships, thus providing a single point of reference to check for any updates to medical history. We design an identity confirmation system via public key cryptography and propose a DNS-like implementation that maps an already existing and widely accepted form of ID (e.g. name, or social security number) to the person's Ethereum address. A syncing algorithm handles data exchange "off-chain" between a patient database and a provider database, after referencing the blockchain to confirm permissions via our database authentication server.[1]

In the following sections we present the design goals of our distributed system. Please see Figures 3-1 and 3-2 below, which show the intended data flows through

[1]The content of this section (3.1) is drawn from our previously published techincal publications [11], as explained in footnote 1.

the MedRec system.

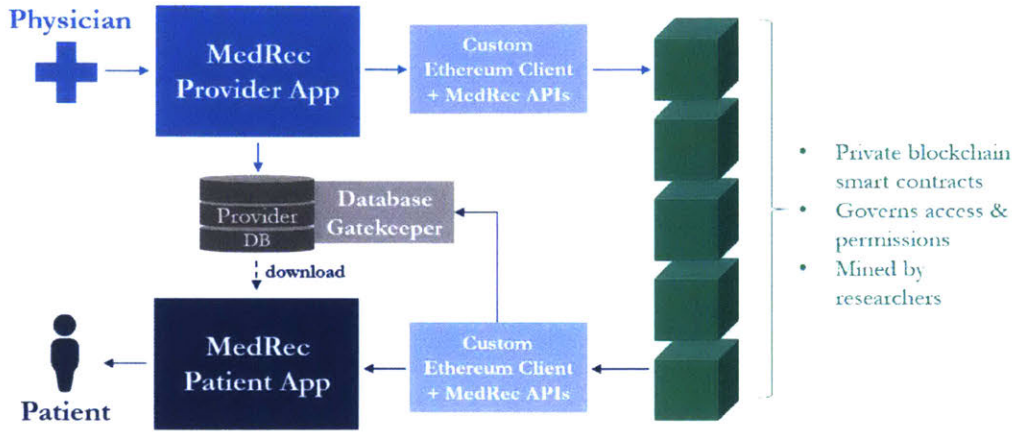


Figure 3-1: Data flow schematic for the MedRec system, showing integration between input data, blockchain directory and data retrieval.

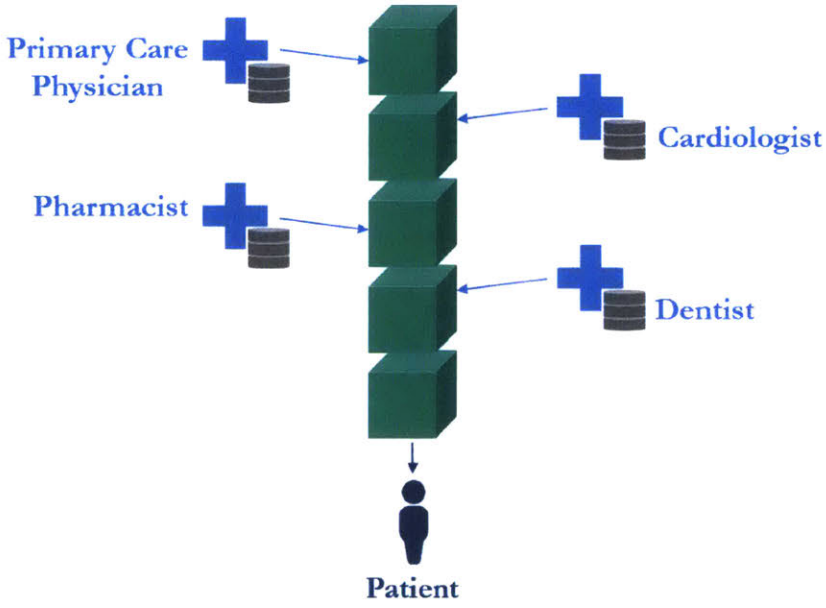


Figure 3-2: An example MedRec network, where the MedRec database keeper is installed at multiple nodes, all coordinating permission and access information via the blockchain log.

3.2 MedRec System Architecture: Design Concept

3.2.1 Smart Contract Structures

Registrar Contract (RC)

This global contract maps participant identification strings to their Ethereum address identity (equivalent to a public key). We intentionally use strings rather than the cryptographic public key identities directly, allowing the use of already existing form of ID. Policies coded into the contract can regulate registering new identities or changing the mapping of existing ones. Identity registration can thus be restricted only to certified institutions. The RC also maps identity strings to an address on the blockchain, where a special contract described below, called the Summary Contract, can be found.

Patient-Provider Relationship Contract (PPR)

A Patient-Provider Relationship Contract is issued between two nodes in the system when one node stores and manages medical records for the other. While we use the case of care provider and patient, this notion extends to any pairwise data stewardship interaction. The PPR defines an assortment of data pointers and associated access permissions that identify the records held by the care provider. Each pointer consists of a query string that, when executed on the provider's database, returns a subset of patient data. The query string should affix the hash of this data subset, to guarantee that data have not been altered at the source. Additional information indicates where the provider's database can be accessed in the network, i.e. hostname and port in a standard network topology. The data queries and their associated information are composed by the care provider and modified when new records are added. To enable patients to share records with others, a dictionary implementation (hash table) maps viewers' addresses to a list of additional query strings. Each string

can specify a portion of the patient's data to which the third party viewer is allowed access.

Our prototype demonstrates this design with SQL data queries. In a simple case, the provider references the patient's data with a simple SELECT query conditioned on the patient's address. For patients, we designed a tool which allows them to check off fields they wish to share through our graphical interface. Internally, our system formulates the appropriate SQL queries and uploads them to the PPR on the blockchain. Note that by using generic strings our design can robustly interface with similar string queried database implementations. Hence, it can conveniently integrate with existing provider data storage infrastructure. At the same time, patients are enabled with fine-grained access control of their medical records, selecting essentially any portion of it they wish to share.

Summary Contract (SC)

This contract functions as a bread crumb trail for participants in the system to locate their medical record history. It holds a list of references to Patient-Provider Relationship contracts (PPRs), representing all the participant's previous and current engagements with other nodes in the system. Patients, for instance, would have their SC populated with references to all care providers they have been engaged with. Providers, on the other hand, are likely to have references to patients they serve and third-parties with whom their patients have authorized data sharing. The SC persists in the distributed network, adding crucial backup and restore functionality. Patients can leave and rejoin the system multiple times, for arbitrary periods, and always regain access to their history by downloading the latest blockchain from the network. As long as there are nodes participating in the network, the blockchain log is maintained.

The SC also implements functionality to enable user notifications. Each relationship stores a status variable. This indicates whether the relationship is newly estab-

lished, awaiting pending updates and has or has not acknowledged patient approval. Providers in our system set the relationship status in their patients' SC whenever they update records or as part of creating a new relationship. Accordingly, the patients can poll their SC and be notified whenever a new relationship is suggested or an update is available. Patients can accept, reject or delete relationships, deciding which records in their history they acknowledge.

Our prototype ensures that accepting or rejecting relationships is done only by the patients. To avoid notification spamming from malicious participants, only providers can update the status variable. These administration principles can be extended, adding additional verifications to confirm proper actor behavior.

Please see Figure 3-3 for examples of how the Registrar Contract, Patient-Provider Relationship Contract and Summary Contract interact in the MedRec system.

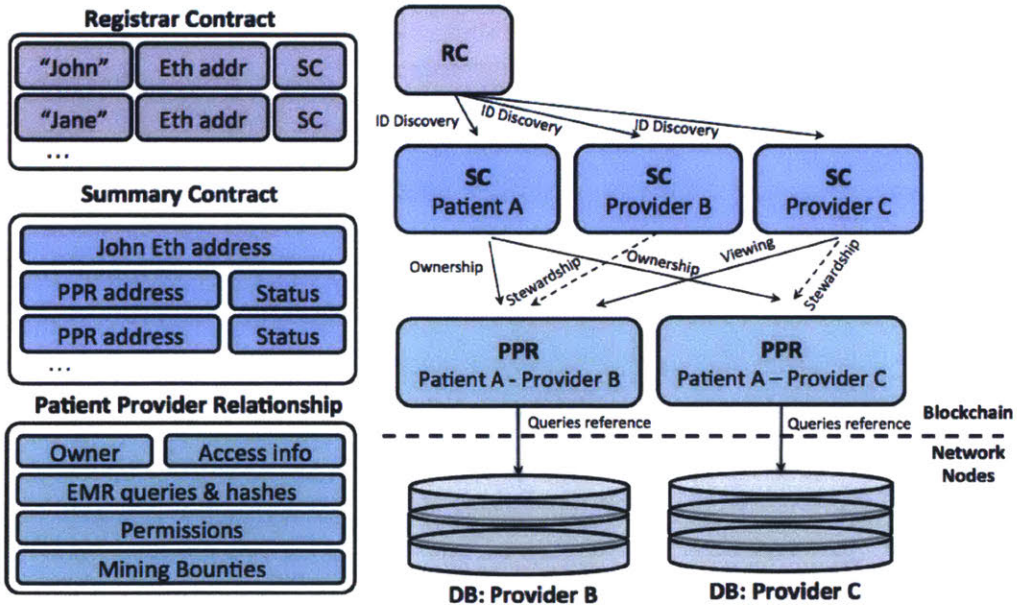


Figure 3-3: MedRec smart contracts on the left of the figure, showing data content for each contract type. Sample relationship graph between contracts and network nodes on the right.

3.2.2 System Node Description

We design the components of our system nodes to integrate with existing EHR infrastructure. We assume that many nodes, and in particular care providers, already trustfully manage databases with patient data stored on servers with network connectivity. Our design introduces four software components: Backend Library, Ethereum Client, Database Gatekeeper and EHR Manager. These can be executed on servers, combining to create a coherent, distributed system. We provide a prototype implementation of these components that integrates with a SQLite database and is managed through our web user interface. Notably, any provider backend and user interface implementations can participate in the system by employing the modular interoperability protocol as defined through our blockchain interface code.

Patient nodes in our system contain the same basic components as providers. An implementation of these can be executed on a local PC or even a mobile phone. Their local database can be one of many lightweight database implementations. The databases can function merely as cache storage of the patient's medical data. Missing data can be retrieved from the network at any time by following the node's Summary Contract. Please see Figure 3-4 below, for a step by step diagram showing orchestration of the modules that define a MedRec "node". Subsequent paragraphs in 3.2.3 Primary System Components will refer back to these steps.

3.2.3 Primary System Components

Backend API Library

We construct multiple utilities, bundled in a backend library, to facilitate the system's operation. Our library abstracts the communications with the blockchain and exports a function-call API. Record management applications and their user interfaces can thus avoid the hurdles of working directly with the blockchain. One such hurdle is verifying that each sent transaction is accepted with high confidence by

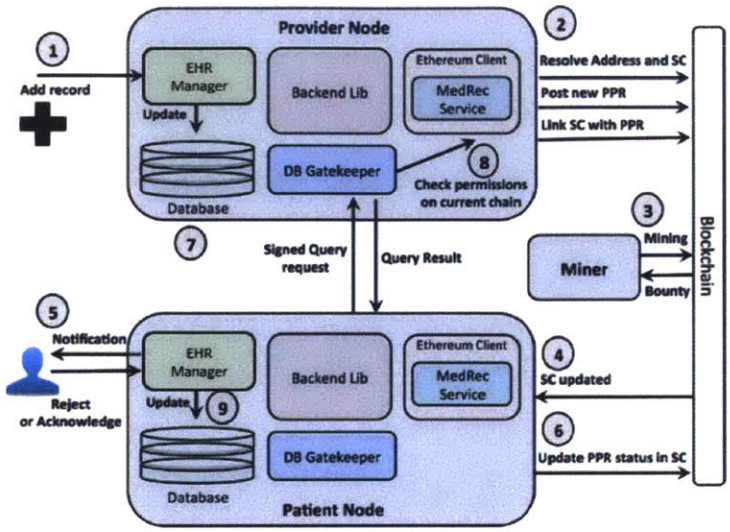


Figure 3-4: System orchestration example: provider adds a record for new patient.

the network. Our library automatically handles the uncertainty of when transactions are mined and deals with cases when they are discarded. The backend library interacts with an Ethereum client to exercise the low-level formatting and parsing of the Ethereum protocol.

Steps 1 and 2 in Figure 3-4 illustrate our backend implementation of a scenario where a provider adds a record for a new patient. Using the Registrar Contract on the blockchain, the patient's identifying information is first resolved to their matching Ethereum address and the corresponding Summary Contract is located. Next, the provider uploads a new PPR to the blockchain, indicating their stewardship of the data owned by the patient's Ethereum address. The provider node then crafts a query to reference this data and updates the PPR accordingly. Finally, the node sends a transaction which links the new PPR to the patient's Summary Contract, allowing the patient node to later locate it on the blockchain.

Ethereum Client

This component implements the full functionality required to join and participate in the Ethereum blockchain network. This handles a broad set of tasks, such

as connecting to the peer-to-peer network, encoding and sending transactions and keeping a verified local copy of the blockchain. For our prototype implementation we use PyEthereum and the PyEthApp client. We modify the client to be aware of our mapping of identity and addresses. We then implement a service to locate the node's Summary Contract (SC), via Registrar Contract address lookup. This service runs continuously within the client to monitor real-time changes to the SC. In the event of an update, the service signals the EHR Manager to issue a user notification and, if necessary, sync the local database.

Steps 4 to 6 in Figure 3-4 continue the use case described above from the patient node perspective. The patient's modified Ethereum client continuously monitors her SC. Once a new block is mined with the newly linked PPR, the client issues a signal which results in a user notification. The user can then acknowledge or decline her communication with the provider, updating the Summary Contract accordingly. If the communication is accepted, our prototype implementation automatically issues a query request to obtain the new medical data. It uses the information in the new PPR to locate the provider on the network and connect to its Database Gatekeeper server.

Database Gatekeeper

The Database Gatekeeper implements an off-chain, access interface to the node's local database, governed by permissions stored on the blockchain. The Gatekeeper runs a server listening to query requests from clients on the network. A request contains a query string, as well as a reference to the blockchain PPR that warrants permissions to run it. The request should be cryptographically signed by the issuer, allowing the gatekeeper to confirm identities. Once the issuer's signature is certified, the gatekeeper checks the blockchain contracts to verify if the address issuing the request is allowed access to the query. If the address checks out, it runs the query on the node's local database and returns the result to the client.

Steps 7 to 9 in Figure 3-4 illustrate how a patient retrieves personal data from the provider node. Note that our components similarly support third-parties retrieving patient-shared data: the patient selects data to share and updates the corresponding PPR with the third-party address and query string. If necessary, the patient's node can resolve the third party address using the Registrar Contract on the blockchain. Then, the patient node links their existing PPR with the care provider to the third-party's Summary Contract. The third party is automatically notified of new permissions, and can follow the link to discover all information needed for retrieval. The provider's Database Gatekeeper will permit access to such a request, corroborating that it was issued by the patient on the PPR they share.

EHR Manager

We tie together all the software components previously mentioned with our EHR management and user interface application. The application renders data from local SQLite databases (designed to be interchangeable with other database software) for viewing, and presents the users with update notifications, and data sharing and retrieval options. Our user interface prioritizes intuitive, crisp, and informative design, as recommended by the Department of Veteran Affairs and ONC's Blue Button design competition [35]. Our application is conveniently accessed through a web interface, built on a python micro-framework. We are especially cognizant of future redesign goals for compatibility for mobile devices, as modern users expect easy access and high quality experiences while on-the-go.

3.2.4 MedRec Blockchain Mining

We incentivize "miners" to participate in the network and contribute their computational resources to achieve a trustworthy, gradual advancement of the chain. We propose a model that engages the healthcare community in network stewardship—

MedRec brings medical researchers and health care stakeholders to mine in the network. In return, the network beneficiaries, i.e. providers and patients, release access to aggregate, anonymized medical data as mining rewards. We explore this idea in our prototype by implementing a special function in the PPR contract. It requires care providers to attach a bounty query to any transaction they send updating the PPR. For example, this bounty query can be formulated to return the average iron levels in blood tests done by the provider, across all patients, in the previous week. When the block containing the record-update transaction is mined, the mining function automatically appends the block's miner as the owner of the bounty query. The miner can then collect it by simply issuing a request for this bounty to the provider's Database gatekeeper. Because it is signed by the provider as part of the transaction, the bounty query is safe from malicious alterations.

This "bounty query" or data reward for mining could enable medical researchers to access population-level insights into medical treatment and healthcare outcomes, potentially revolutionizing how data is gathered and accessed for research purposes. We envision future updates to the mining model where miners (i.e. medical researchers) can specify preferences for demographic cohorts and features of the data they are looking for, in order to enable precision medicine and targeted research (while still preserving the privacy of the patients). This could be accomplished by faciliating a bid system, where medical researchers could propose queries they would like included as their mining "bounties" (with a sum of Ether associated as the "fuel" to win the bid auction). Such a system would necessitate vetting of the queries, to ensure that the returned data could not be used to reidentify patients (even if drawn from a future pool of anonymized data). The medical researchers can be incentivized to continue mining day-to-day, as the data accessible to the bounty query could be timebound to the time of the mined block. To continue obtaining the latest data, miners must

continue a regular pattern of computational contributions to the network. [2]

3.3 MedRec Prototype Code Review

With the MedRec system architecture concept described above, we will now go through an in-depth code review to examine the features of our envisioned design that are fully or partially implemented. The codebase detailed below is an early research prototype. The MedRec codebase lives in an MIT Enterprise Github repo, with separate branches for the main development and our test integration with Beth Israel Deaconess Medical Center. This section of the thesis will be posted online to supplement the Github repo with a detailed ReadMe, to provide helpful background information for any future developers wishing to work on the project.

3.3.1 MedRec Adaptation of Core Ethereum Functionality

Initial Setup:

Our first step in building a private, Ethereum testnet was to set up several Virtual Machines (VMs) as Ethereum nodes. To do this, we prepared the VMs with the prerequisite software packages for the full necessities of the MedRec system:

Standard, Non-Ethereum installs:

- Python 2.7
- SQLite (for our particular database implementation, though others could certainly be used)
- Flask (for our particular web framework implementation)
- Jinja (for our particular web framework implementation)

[2]The content of this section (3.2) is drawn from our previously published techincal publications [11], as explained in footnote 1.

Ethereum installs:

- Geth (the Go implementation client)
- Pyethapp (the Python implementation client)

With the VMs now properly prepared to run an Ethereum node, the next step is to initialize clients on each VM with a genesis block and starting allocations of Ether. Our testnet's genesis block was defined in JSON, in the format shown below in Figure 3-5 (but with our own unique values).

```
{
  "alloc"      : {},
  "coinbase"   : "0x0000000000000000000000000000000000000000",
  "difficulty" : "0x20000",
  "extraData"  : "",
  "gasLimit"   : "0x2fefd8",
  "nonce"      : "0x0000000000000042",
  "mixhash"    : "0x0000000000000000000000000000000000000000000000000000000000000000",
  "parentHash" : "0x0000000000000000000000000000000000000000000000000000000000000000",
  "timestamp"  : "0x00"
}

"alloc": {
  "0x0000000000000000000000000000000000000001": {"balance": "111111111"},
  "0x0000000000000000000000000000000000000002": {"balance": "222222222"}
}
```

Figure 3-5: Code excerpt showing genesis block initialization for Ethereum testnet [2].

In our implementation, we use the Geth client for the designated bootstrap node—the VM that is always turned on first and acts as the miner. Because we did not have plans to significantly alter the mining protocols, it made the most sense to use the best supported and most fully developed Ethereum node client for the mining functionality. Our Geth mining client does not run automatically on VM startup, to avoid bloating the chain with empty blocks during intermittent test development. This could be modified in the future, when a deployed MedRec network would need to be live and mining at all times.

Conversely, because we required customizability for the other nodes in our system (which would act as "provider" and "patient" nodes), we decided on pyethapp as our Ethereum client for the subsequent VMs. With pyethapp based on python, and being most familiar with development in python, we were better situated to move forward with our MedRec implementation goals. When starting up the pyethapp clients for these nodes, they connect to the bootstrap Geth node by referencing the bootstrap node's "enode" address. Figures 3-6, 3-7 and 3-8 below are virtual machine snapshots that show the Geth and Pyethapp clients running on the Miner and Patient/Provider nodes, respectively.

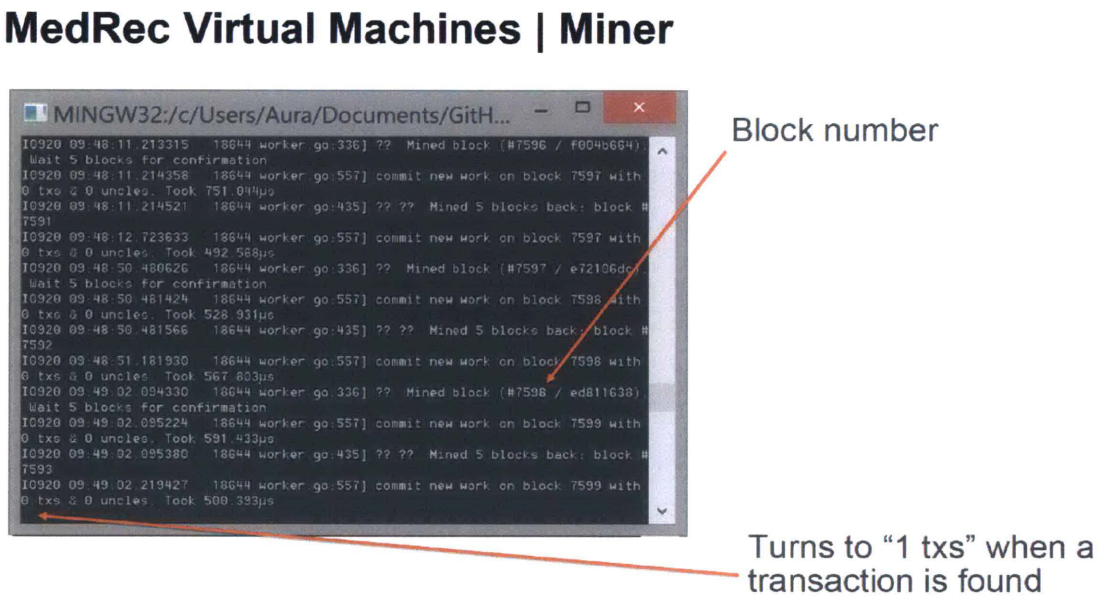


Figure 3-6: The Miner view includes metadata about the current state of the blockchain, including a rolling network timestamp, the latest "block number" (a chronological counting of blocks since block 1), whether there are any pending transactions and how long in micro-seconds it took to mine the block. For example: `I0920 09:49:02.219427 18644 worker.go:557] commit new work on block 7599 with 0txs & 0 uncles. Took 500.393 us`

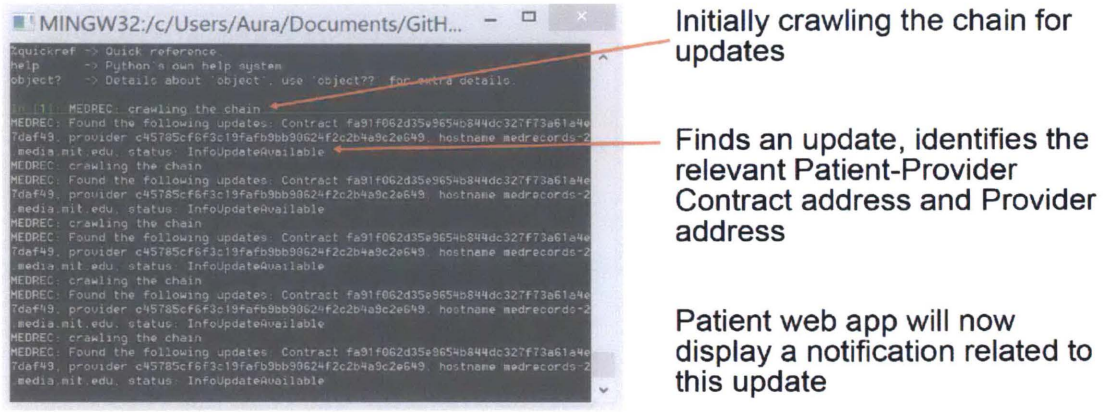


Figure 3-7: The Patient/Provider views show two different scenarios while both clients are crawling the blockchain: the Patient's client has found an update on the blockchain, and the node confirms the associated Patient Provider Relationship contract address where this update (perhaps a new medical record) can be found. For example: `MEDREC: Found the following updates: Contract #, provider #, host name medrecords-2.media.mit.edu, status: Info Update Available`

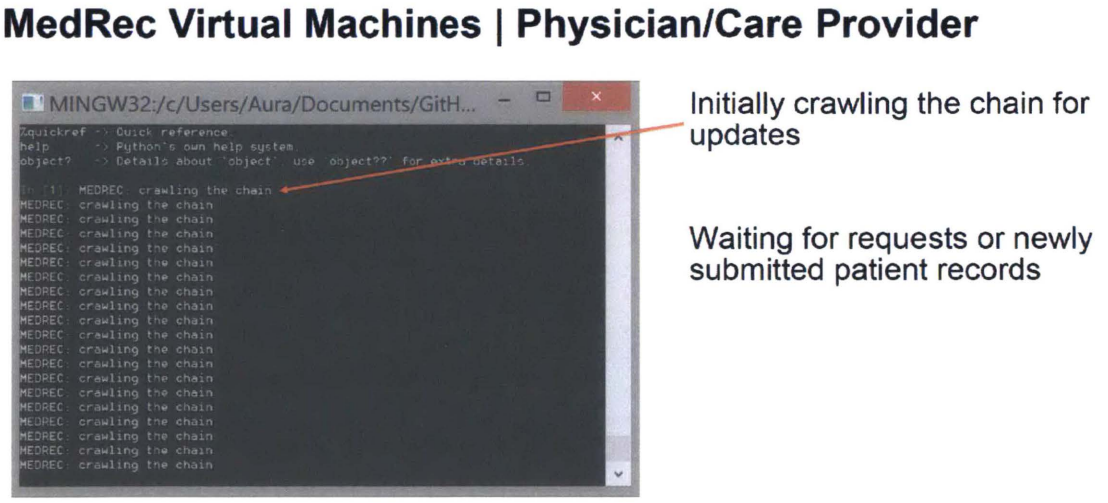


Figure 3-8: The Provider's client has yet to find an update (no change in viewership authorizations, no patient-submitted symptom reports on the horizon) and will continue crawling the chain until there is an update. For example: `MEDREC: crawling the chain`

3.3.2 MedRec Blockchain Interface Code

Scripts described in the following section, in order of appearance, include:

- BlockChainHelpers.py
- Eth_Adapter.py
- GlobalRegistrars.py
- PatientSummaryContract.py
- PatientProviderRelationship.py
- Patient_contact_record.se
- Relationship_contract.se
- Login.py

With Ethereum clients up and running on the VMs, tailored to their use cases in the MedRec system, we configure a series of blockchain helpers that control parameters of the mining and verification steps. In our code, these are found in BlockChainHelpers.py and contain functions that: set the "wait time" in number of blocks that must be mined after the current block before a transaction is "complete"; verify that a submitted transaction has been successfully logged on the blockchain; confirm that the transaction was found and that mining has not timed out. In addition to the blockchain helpers code that we have modified, we imported code from Ethereum for the "Eth" object that allows command line interfacing with the live chain and the GlobalRegistrars approach for registering the hashes of new contracts.

The meat of any Ethereum implementation is in the smart contracts. The MedRec system defines three smart contracts (described in detail below) and a set of blockchain-interfacing functions that execute logic for the creation of and updates to these smart

contracts. Our scripts "PatientProviderRelationship.py" and "PatientSummaryContract.py" read in (separate) definition files for the smart contracts, apply updates to the data in these contracts according to the definition schema, and post the either newly created or updated smart contracts (via Ethereum transactions that represent these changes) to our live blockchain. These two scripts are the core APIs that take interactions from the WebApp and run the corresponding orchestration on the blockchain. In other words, they are the heart of the MedRec implementation.

As described in the system architecture section, the MedRec prototype defines two primary smart contracts beyond the registrar orchestration code; one that orchestrates the "Patient Provider Relationship" (defining the data associated with a Patient and Provider pair) and the "Summary Contract" (listing all Patient Provider Relationships and current relationship statuses associated with a particular user, either provider or patient). Both contracts are written in serpent.

The script "relationship_contract.se" is the "Patient Provider Relationship Contract", which:

- Confirms the patient and provider's Ethereum addresses
- Defines means of adding a 3rd party, authorized "viewer" with access to retrieve data
- Defines means of adding a new record, and orchestrating the miner reward for processing the transaction associated with adding a new record

The script "patient_contract_record.se" is the "Summary Contract", which:

- confirms the owner's Ethereum address (i.e. either a patient or a provider)
- confirms the contract addresses for the associated PPRs
- confirms the "supplier's" user addresses (i.e. the other pairwise user that was the source of data in the PPR)

- prepares to accept status updates to the summarized PPRs (new records, new 3rd party viewer authorized, etc)

- confirms any 3rd party viewers that have been authorized for shared access

The final blockchain interface code is a Login.py module that associates the user name set by the global name registrar (defined in GlobalRegisters.py, mentioned above) with the public key of the user (searchable on the chain). The password should unlock the private key of said user. This code interfaces with the blockchain to create a new smart contract for new users (which is then posted to the blockchain with the scripts mentioned above), and fund it with Ether.

3.3.3 MedRec APIs and Database Gatekeeper

Our bundle of APIs translates between the MedRec WebApp and the blockchain interface code. The scripts define a layer of functions that orchestrate medical record management tasks (new and existing record updates, new relationships with doctors, authorization for data sharing, et cetera) with integrated feedback from our blockchain permissioning and data access mechanisms. Scripts described in the following section, in order of appearance, include:

- MedRecPyethappService.py

- Local Configuration.py

- PermissionsFormulator.py

- RemoteSQLiteService.py

- Setup_init.py

- SQLiteSyncingClient.py

- UserNotification.py
- UserNotifications_SerivceAPI.py

The MedRecPyethappService.py defines a method to crawl the Ethereum chain and flag updates that have been pushed to the smart contract, but have not yet been updated in the WebApp. This script underpins our user notification system, allowing the MedRec system to have a dynamic updating feature that keeps the patient's healthcare record up to date and relevant. This script integrates closely with the global registrar definitions, in order to associate the updates with the appropriate contracts, and ultimately, end-users of the WebApp.

Our LocalConfiguration.py script is a relatively standard approach for grouping passwords, database paths, server logins and more in a file that is private, and not uploaded to public repositories. When referencing objects (like our database files) in other scripts, we point to LocalConfiguration.py to grab the full paths or login information.

The Permissions Formulator.py script is a "stub function", outlining a concept for the miner bounty. As mentioned previously, instead of earning bitcoin or ether as a financial reward for successful transaction validation and "mining", our miners are rewarded with access to data. This script shows an example where a pre-approved query (in this case for the type of record posted, i.e. Bloodwork, Medication or Vaccination) is established as the miner's fee. There are a wealth of options for how to implement this stub function in live networks, depending on the incentives and regulation involved. For example, pharmaceutical companies who "mine" on the MedRec network, or large research institutions like the CDC and NIH, may want to define their desired reward queries in advance, specifying conditions on the demographic pool that their research targets. Conversely, the maintainers of the MedRec system might define a set of queries that are pre-approved and constantly changing, in order to keep the mining entities interested in the latest, unpredictable data outputs. There is much further

work to do on this script, integrating the simple goal of query-permission-granting with a data science framework that can provide for the aggregation and anonymity required to make both research-worthy and privacy-preserving queries.

The RemoteSQLiteService.py script is a key element of the "Database Gatekeeper" functionality described previously. These methods begin to define our networked syncing service, which allows transfer of content between SQLite instances mediated by the blockchain's permission information (and upon further completion of the code, mediated by the public key signatures that maintain cryptographic identities in the system). This script first checks that the viewer identity requesting information is logged on the blockchain as having access rights to the data in question. The MedRec system currently accomplishes this by confirming that the requesting identity either has ownership access to the contract (i.e. is the patient who has default rights to the data referenced by the contract) or has 3rd party viewership rights, as previously set by the patient. For the latter case, we consider the need to have a restricting query that limits the information transferred, based on particularities of what has been shared with the 3rd party. The nature of appropriate restricting queries is currently left quite open-ended, in order to support perhaps unforeseen needs for granularity in what is and isn't shared and transferred. After confirming access and permission per the Blockchain smart contracts, the syncing service returns the requested data from the source database. While this script defines a protocol for SQLite, the logic is in principle extensible to many alternative database implementations. The approach would transfer quite well for most string-queried databases, including SQL, a standard of healthcare data warehousing. While we currently use RPyC (a python library) to establish the connections and complete the data transfer, this mechanism could also be updated to a more enterprise-friendly software suite (such as the server connectivity tools that come with fully fledged SQL implementations). We chose RPyC mostly for convenience in working around the limitations of SQLite (which was in turn chosen

for its simplicity and lightweight implementation befitting a research prototype).

Working closely with RemoteSQLiteService.py, the SQLiteSyncingClient.py module constitutes the other "half" of the Database Gatekeeper implementation. Here, we invoke the methods defined above and take the returned records (post-release from the source database) and sync them into the local app's Records table. The syncing methods are separated by whether we are writing to the "owner's" (i.e. the patient's) local records or a 3rd party viewer's local records. Updates to existing records are handled, and new records are retrieved.

The Setup_init.py script defines a set of test patients, both from a simulated data and simulated account perspective. Based on the preset identities, we: initialize the database instances; generate and load sample data (fake bloodwork, medication and vaccination records) in two different databases (simulating two different provider sources); create Personal Summary Contracts for each patient; associate miner permissions for relevant contracts (i.e. tracking the miner who validated the transaction posting a new contract or data update); define command line arguments to run setup methods.

The UserNotifications.py defines methods that give fine-grained control to patients over what they pull into their personal health record. As described previously, there are three notification types: initiation of a new patient-provider relationship, updates to an existing patient-provider relationship and updates to 3rd party viewership rights. This script picks up updates in these categories, as logged in our Notifications database table, and applies the patient choice of rejecting or accepting each update. This script is called from the WebApp and determines what information is shown in the user interface. The closely related UserNotifications_ServiceAPI is called to populate the updates in the Notification database table, from which UserNotification.py reads. This latter script is called by MedRecPyethappService.py as it crawls the blockchain for updates, and pushes all current notifications as they appear

on the chain.

3.3.4 MedRec WebApp

The MedRec WebApp, as the focal point of record review for patients, is the ultimate orchestrator that sets our APIs in motion for retrieving and updating existing data, and initiating data sharing. Though our prototype implementation is far from end-user readiness (in both the fullness-of-features dimension and UI design), the WebApp has served as a useful tool for visual demonstration of the MedRec system. Current features include a means for reviewing records (sorted by record type), sharing data with an additional entity (other than the provider acting as the primary data source), and the patient control mechanisms for accepting or rejecting update notifications to their record.

Regarding our callbacks in the MedRec WebApp, we directly call only four of the previously described modules (all others are invoked further down the stack). Please see Figure 3-12 for a flow diagram of the MedRec codebase.

- PermissionsFormulator
- PatientProviderRelationship (both PatientProviderRelationship_ProviderAPI and PatientProviderRelationship_PatientAPI)
- UserNotifications
- LocalConfiguration

The WebApp establishes a home page, where all bloodwork, medication and vaccination data are loaded and rendered on separate tabs. The WebApp also displays available updates that require input from the user (accept/reject). We pull this data via SQL queries from our SQLite instances. We handle the intent to share records on

a granular basis by entry (e.g. you can share a single bloodwork result, without having to share all data in that category), and allow the creation of new records (both for patients contributing a symptom report and for providers submitting health record data). For the latter use case, we realize that providers may wish to avoid data entry through yet another interface, and rather than expecting them to always submit data through the MedRec WebApp, we anticipate pulling in new information directly from the provider's networked databases.

Built on Flask with simple HTML/CSS templates and running on our local servers, the MedRec WebApp is not intended to scale to serve multiple users. We would recommend re-architecting on a more robust web development stack, should future developers be interested in taking the project forward.

Figures 3-9 and 3-10 show two snapshots of the MedRec Version 1.0 user interface, bracketed by command terminals that are running the mining Geth client, patient/provider Pyethapp clients, and the database access interface. Figre 3-11 shows the latest WebApp user interface mockup.

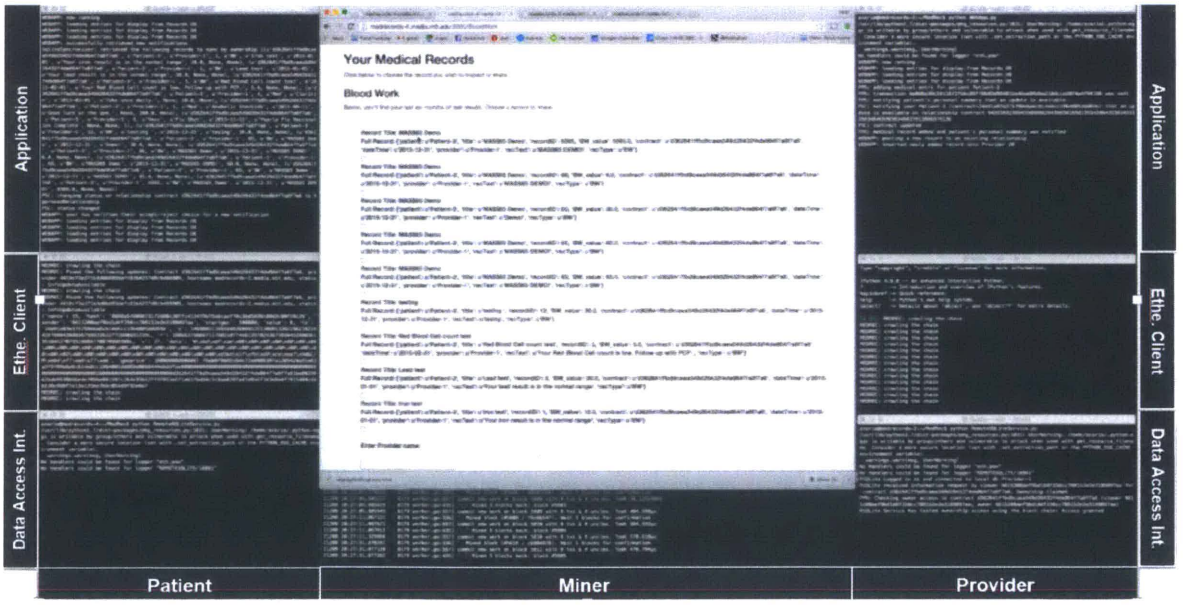


Figure 3-9: This view is taken from a demo prepared for the MAS.s65 Blockchain technologies class in December of 2015, showing a listing of bloodwork records.

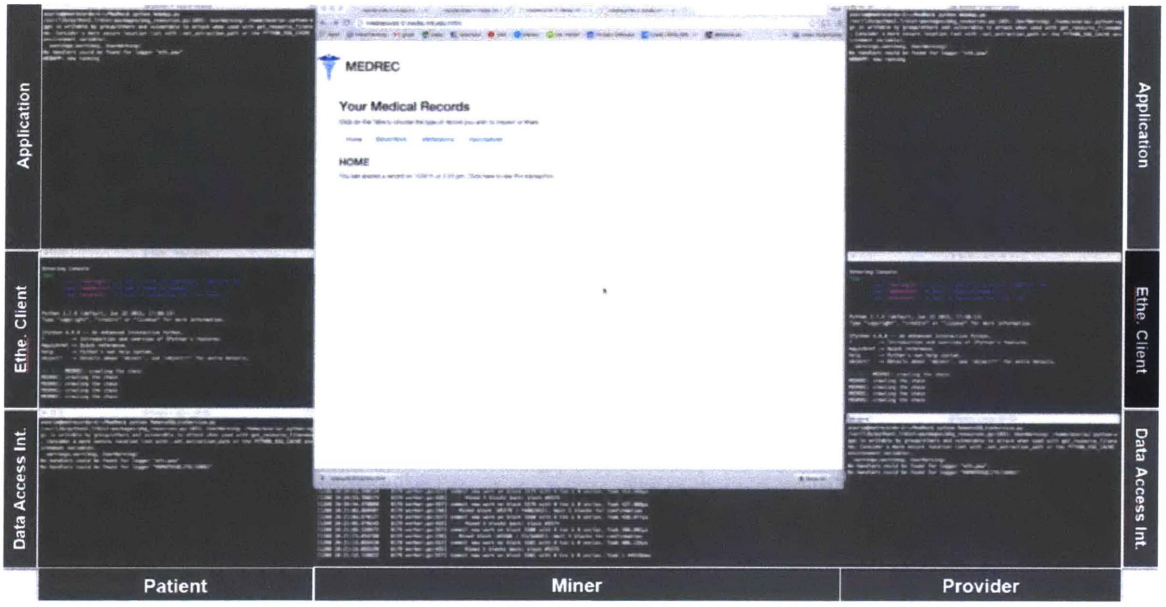


Figure 3-10: This view is taken from a demo prepared for the MAS.s65 Blockchain technologies class in December of 2015, showing the original homepage.

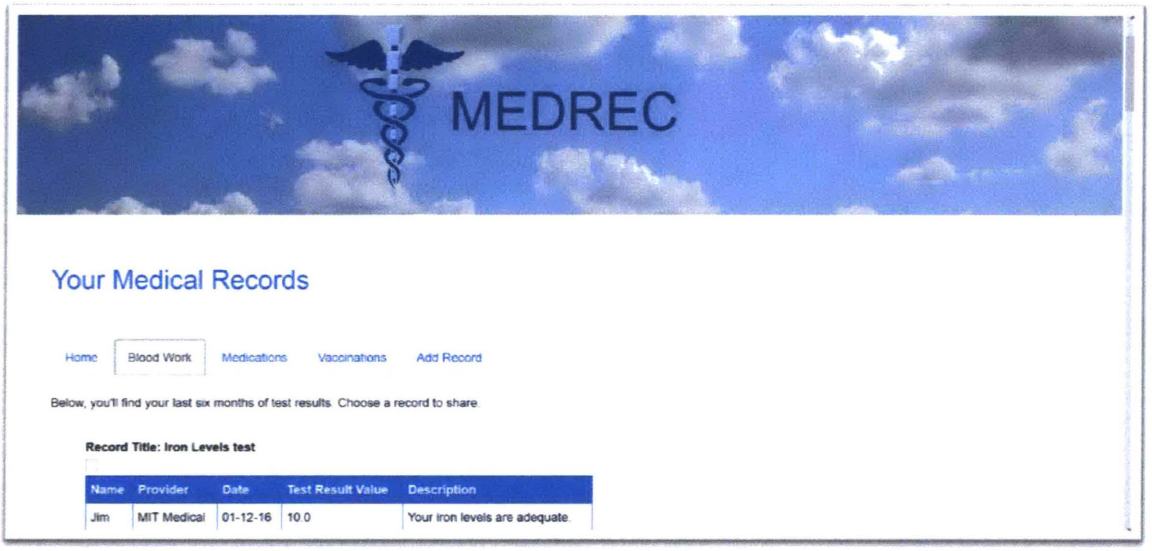


Figure 3-11: A snapshot of the MedRec Version 2.0 user interface from July 2016, with improved visual design.

3.3.5 Prototype Code Review: In Summary

The MedRec code base establishes an Ethereum blockchain test network, blockchain-interfacing APIs, an off-blockchain data syncing service, and a WebApp for patient and provider use. The interaction between modules serving these purposes is summarized in Figure 3-12. The far left column in the Figure establishes base level scripts that stand on their own, making no calls to other modules. As you progress left to right, each column establishes a layer with more complexity and more hooks into other parts of the system. In the final layer, the WebApp serves as the focal point of orchestration, driving execution of other scripts based on user input. Certain background processes must be running at all times (such as mining and the patient/provider clients), even when activity in the WebApp is dormant.

When considering all modules in the MedRec code base, we identify six scripts as the "heart of the project". These scripts execute the unique ideas in the MedRec system design, and interface with the blockchain in an innovative way. While the other modules are of course integral to the functioning of the prototype, they represent tasks that are also handled similarly in many other systems (such as login and local confirmation orchestration, basic database design, or slightly-modified standard Ethereum protocols). The key scripts are:

- PatientProviderRelationship.py
- PersonalSummaryContract.py
- patient_contact_record.se
- relationship_contract.se
- MedRecPyethAppService.py
- MedRec WebApp

While not all features of the MedRec system design and end-product vision are implemented in the current code base, the MedRec repository does serve as an effective research prototype and was tested with Beth Israel Deaconess Medical Center in Boston. Key features that merit further code development include: management of the crypto keys for signature verification before off-blockchain data release; raw data to pointer management for proper data referencing in the smart contracts; formulation of bounty queries for miners beyond our simple POC cases; end-to-end encryption for the off-blockchain data syncing (likely via integration with off-the-shelf data syncing and management services). As discussed later in the Evaluation section, a functioning MedRec network would also rely on a system of identification that would be able to map patients across hospitals and treatment sites. While we could maintain an internal directory mapping the various hospitals IDs to a single, unique patient ID in the Ethereum address context, this would concentrate the identity management inside the MedRec system–a design decision we would prefer to avoid for security and privacy reasons. MedRec does not intend to solve the "global identity" challenge, and will look to build on other blockchain-based identity solutions that are beginning to emerge in this research area. Being able to decentralize the management of identity is still a challenge under research development.

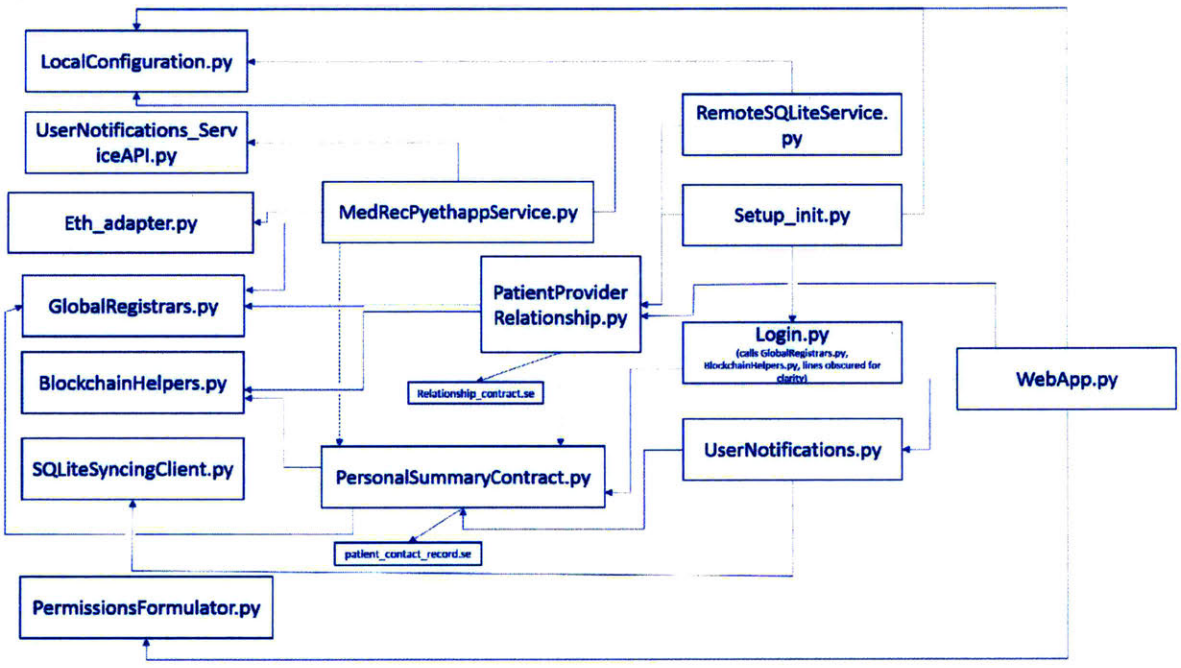


Figure 3-12: Code flow diagram showing all major MedRec scripts and the relationships between them. Arrows point from the script making the "call" to the script housing the function that is "called", i.e. PatientProviderRealtionship.py calls BlockchainHelpers.py, and PatientProviderRelationship.py is in turned called by the WebApp.

Chapter 4

Evaluation

The MedRec system design gives patients an immutable log of their medical history, which is not only comprehensive, but also accessible and credible. This restores patient agency, as participants are now more fully informed of their medical history and any modifications to it. Through permission management on the blockchain, we enable patient-vetted data exchange between medical jurisdictions and an interoperable content management system for the physicians supervising these records. The blockchain ledger keeps an auditable history of medical interactions between patients and providers, which is likely relevant for regulators and payers (e.g. insurance) in the future. Below, we discuss our in-situ deployment testing and consider the security, privacy and interoperability implications of this project. [1]

4.1 Pilot with Beth Israel Deaconess Medical Center

In August 2016, we completed a small-scale pilot with Beth Israel Deaconess Medical Center (BIDMC)[2], in collaboration with their IT team, Clinical Information Systems and Infrastructure groups. First introduced to John Halamka, CIO of Beth Israel

[1] This evaluation overview and sections 4.2-4.4 are drawn and adapted from our previously published techincal publications [11], as explained in footnote 1.

[2] BIDMC also serves a role as the Harvard Medical School Teaching Hospital.

Deaconess, in April of 2016, we proceeded through multiple stages of vetting and project scoping to prepare the codebase for an integration with test medical data from the BIDMC servers. We evaluated MedRec's ability to smoothly intake and parse a standard clinical document, link our Database Gatekeeper utility to the relevant Beth Israel endpoint and test an end-to-end system flow from the hospital's existing user interface for physicians through our backend and out to a sample patient node. The following sections detail the process of defining the parameters for the integration test, obtaining BIDMC approval, preparing the code and test dataset, completing the on-site integration test and evaluating our results.

4.1.1 Defining the Pilot Scope

To effectively test MedRec, it is important to define a scope: to identify exactly which features we would deploy, and at what scale. On one end of the spectrum would be a full-scale, multi-week, multi-institutional network test with real users retrieving test data from and sharing test data between multiple provider locations (say, both BIDMC and Massachusetts General Hospital), engaging all functionality from the WebApp user interface down to the networked databases and blockchain log with a large throughput of data. On the other end of the spectrum would be a simple 10-minute data entry test, determining whether the MedRec system could process a single record, post a relevant smart contract to the blockchain, and support data retrieval by a test patient node. In collaboration with Dr. Lawrence Markson, VP of Clinical Information Systems at Beth Israel Deaconess (our primary contact during the integration), we settled on an appropriate middle-ground, given both the current maturity of the MedRec prototype and the available resources at BIDMC to support the integration test.

The MedRec test at BIDMC included the following (drawn from our formal agreement with the team, included in Appendix A):

- The simulated "provider" databases were prefilled with test data, for batch retrieval by the MedRec system (rather than record-by-record entry via the WebApp). BIDMC supplied two DB instances with SQL Server 2014 (test systems, not in-production systems) populated with de-identified medication data (as opposed to medication, vaccination and bloodwork data), simulating two different provider data sources. BIDMC supplied the SQL Server login (username/password) and networking information. We were given read access, with a discussion that write access could be considered in the future, to allow patient nodes to contribute to their own record, as in the OpenNotes system pioneered at BIDMC.

- MedRec "Database Gatekeeper" utility tests record exchange between the two DB instances established above, via the MedRec blockchain permissioning protocol. The software was supplied on MIT VMware VMs, running on an MIT-supplied machine, to be connected to the isolated network set up at the BIDMC Lab.

- Scope of the integration: Test end-to-end flow from batch pickup of pre-filled records in the SQL server instances, to successful pointer posting to blockchain, with update notifications logged and accepted, permission information logged as appropriate per the patient/provider pair identified in the record, successful blockchain mining of the posted record transactions, and ultimately through to data retrieval by an authorized "patient" node.

- Duration of the integration: Our timeline and extent of testing included two full (8hr) days of on-site work at the BIDMC premises in downtown Boston, on August 18th and 19th, 2016.

4.1.2 BIDMC Approval Process

To agree on the project scope above, and the security parameters that would surround our on-site visit, we met several times with Dr. Lawrence Markson and his team. An introductory conversation with Dr. Markson and two developers on their Medication Reconciliation team provided an overview of their backend systems and procedure for handling in-patient and out-patient medication records. Subsequent phone calls established the subset of MedRec features that we would test, and the nature of a BIDMC-supplied test dataset with interesting "features", as described in more detail below. To establish the on-site visit procedure and obtain final approval for the integration, we met with the BIDMC Chief Information Security Officer, their server administration team, the Infrastructure Group and Dr. Markson, where we agreed upon the following:

- BIDMC would establish an isolated intranet for us, with static internal IP addresses for our VMs.
- When on this intranet, we would be able to access the local SQL Server test data instances, and would have no connection to any external internet network.
- If a second day of integration testing is required (which it was), the MIT laptop with the provided VMs will be left overnight within the closed, secured testing area.
- After completing the test of the MedRec codebase on our MIT-provided VMs, we would permanently delete the VMs and wipe the host computer before leaving the BIDMC building.

We were allowed to keep a copy of the updated test code in secure, offline storage at the BIDMC premises for future integration testing. While we could not bring a copy of the modified code back to MIT (we established a principle that the ENTIRE

integration test would stay within BIDMC data boundaries), we do have copious integration notes on improvements we would make to the MedRec software for future integrations, based on our learnings at BIDMC. Though this integration test never touched live medical records nor any PII (Personally-Identifiable Information), we established the above best-practices as a learning opportunity and kept to them strictly, as monitored and verified by the BIDMC team.

4.1.3 Preparing the Codebase and Test Dataset

To prepare the MedRec codebase for the integration, we worked on modifications to the off-blockchain syncing code and the WebApp in a new Github repository branch, created for the occasion. Because our initial prototype worked with syncing between SQLite databases, we needed to modify the code to work with fully-fledged SQL instances. We chose to use pymssql [36] as the database interface package between Microsoft SQL Server and our python orchestration scripts. The bulk of these changes were made in the "RemoteSQLiteService.py" module, which we renamed to "RemoteMSSQLService.py" on the Beth Israel integration branch of the MedRec repo. In an attempt to simplify the integration, we chose the earliest, most stable version of the WebApp for the integration test and modified the script to expect only Medication records. In addition, we updated the expected data schema to match the database table columns as provided ahead of time by Beth Israel Deaconess.

The integration also required a custom script to generate new contract addresses for the existing patient-provider pairs in the BIDMC SQL instances. This is due to the fact that we were testing MedRec against a pre-filled dataset that did not already have contract addresses (as opposed to testing on data that was entered through MedRec, where a contract address would already be provided for). This code approach will be useful in future integrations, where existing hospital data storage would need to be retro-fitted with contract addresses for use in a MedRec network.

While we made internal preparations for the MedRec codebase, we also designed features of a test dataset that would be generated by Dr. Markson. The first requirement was that the data be split between two databases, so that we could simulate data posting and retrieval from two separate "providers" A and B. Within each database, we were then interested in records where a single patient-provider pair had multiple entries (i.e. multiple prescriptions prescribed by the same doctor) so that we could test association of multiple records with the same Patient-Provider smart contract. We were also seeking records for the same patient across the two databases, so that we could test the association of data from two different sources back to the same Patient in their Personal Summary Contract. These are just a few of our considerations that would define interesting data features; please see the appendix for a full listing of the dataset design criteria. In total, we landed on an estimate of about 100 to 200 test records, representing a spread of about six months.

4.1.4 Onsite Integration Process and Results

At BIDMC, we leveraged two work stations: one in an isolated datacenter where the laptop would run the MedRec VMs, and our conference room (thoughtfully provided by Dr. Markson and his team) where we would remote-in to the VM intranet and orchestrate the integration test. Below we present a summary of the integration challenges and our solutions.

Several challenges surrounded proper networking, VM communication protocols and SQL database connectivity. The intended DHCP and Bridging mode approach did not work, and we reverted to the static IP plan for our VM intranet connectivity needs. While onsite, we developed a supplemental script to test the pymssql connect() call and wrote a class with methods to test retrieval of records from the SQL instances by indexing on Ethereum contract address, and on patient ID. Ultimately, we updated our pymssql syncing approach with FreeTDS version 4.2 (a type

of compatibility mode) for integrations between a linux environment, python scripts and SQL databases. After overcoming this hurdle, a simple escape character set as one of the SQL server's password characters led to further hours of connectivity troubleshooting.

These, and other relatively straightforward fixes impressed upon us the importance of "deployment scalability"–that if the MedRec codebase was to be deployed across many hospitals as part of a future MedRec network, the interfacing code must be exceptionally streamlined. Fortunately, due to our backend architecture, the steps necessary to integrate with BIDMC's string queried database proved relatively straightforward. As expected, rather than having to redesign the MedRec system, our design allows certain simple updates to schema files and to the WebApp. This validated our hypothesis that MedRec could be adapted to multiple database table structures, provided that they are string-queryable. Furthermore, the MedRec contract-generation script that we developed anew for this integration will prove useful in retrofitting other provider datasets for future integrations. Another option on the horizon, rather than trying to integrate with the custom specifications of each hospitals' data storage infrastructure and networking map, lies in accessing data from standard API endpoints that hospitals are being encouraged to support by 2018 as part of Meaningful Use Phase 3 [37].

To summarize the MedRec codebase integration results: we were able to retrieve records from the SQL databases, affix these with a MedRec-generated contract address (via Dr. Markson's write access to the databases), post references to this data into the appropriate blockchain smart contracts, run the MedRecPyethapp.py service to crawl the chain and find these updates, and finally, to accept these updates (which populated correctly) in the MedRec WebApp UI. While we did not ultimately make use of the full richness of the 200+ record set due to time constraints (i.e., we did not process the separate patient-provider contracts for all this data), we did succeed in

validating the core MedRec functionality. This evaluation (beyond the months-long design and approval phase) represents 16+ hours of onsite deployment time, 30 lines of pre-prepared new code, an existing script of 100 lines of code adapted to specific BIDMC syncing interfaces, and the sucessful resolution of both minor (syntactic) and major (DB interfacing and connectivity) hurdles. We expect this contribution of time and resources to be representative of integrating with test datasets at other institutions. Integration with live, HIPAA protected records across more than one institution would present a different scale of challenges, as dicussed in Section 4.5. This completes our initial testing goals for the MedRec prototype; based on the results of this successful, early-stage pilot, BIDMC welcomed the MedRec project to return in the future for further deployment testing.

4.1.5 BIDMC Integration Learnings

In evaluating the results of the integration test, we identify several key areas for improvement. The first is run-time performance; our blockchain implementation took on the order of 10s of seconds to assign new contracts and to respond with the contract information we needed for record update logging. We felt this delay directly, as we would have to wait for several moments between repeat runs of the code execution steps. We believe this can be addressed in future versions by tweaking the parameters in BlockchainHelpers.py (shorter subsequent block count for validation of transactions) and improving the search approach that crawls the blockchain looking for updates. Additionally, we would look to redesign the WebApp with an asynchronous structure, so that UI does not halt until all sent instructions are completed (which improves the usability of the interface).

The second area for improvement is the "data exchange robustness" of our integration scripts, as briefly alluded to above. While we were able to successfully connect in the end, and pull from the SQL databases as expected, the integration code re-

quired prep work to prepare for a new schema. We also faced a few idiosyncrasies of SQL database connection limitations during the onsite troubleshooting. Even without depending on the hospital rollout of standard API endpoints (which would be the cleanest solution), we believe this might be addressed in future versions by running algorithms in the MedRec Database Gatekeeper that can scrape data based on certain probabilistic patterns and pull from various schemas.

A final learning from the integration centered on the need for a better Ether-generation mechanism in the MedRec system. While we do not heavily emphasize the Ether token in the MedRec design, this token is required "fuel" for execution of the smart contracts. At one point in the integration, our progress was halted until we realized that one of our test user accounts had run out of the Ether we had pre-supplied to it. This opens an interesting opportunity to design a new chapter of the MedRec project–should there be a marketplace where MedRec Ether can be bought and sold, should insurance companies furnish the Ether to their patients, can patients mine Ether themselves, or is this an unnecessary addition to the system that might be designed-out in a future blockchain architecture?

In closing, the Beth Israel Deaconess Medical Center integration test was a key turning point in MedRec development. We took our research design and early prototype and successfully tested MedRec functionality with an in-situ system. While the current MedRec code would require significant updates (and likely a rearchitecting plan for a new blockchain foundation), this evaluation was invaluable to our understanding of what it takes to integrate a blockchain solution with existing record-keeping infrastructure. We look forward to further discussion with the BIDMC team, as they advise the project and consider their own future blockchain in healthcare efforts.

Figure 4-1: These photos show myself (right) and Asaph Azaria (left) on-site at BIDMC for the two day integration test in August 2016.

4.2 Comments on Security

First, on robustness and security: our blockchain implementation enjoys several key properties of decentralization. MedRec enjoys a strong failover model, relying on the many participating entities in the system to avoid a single point of failure. Medical records are stored locally in separate provider and patient databases; copies of authorization data are stored on each node in the network. Because both the raw medical data and global authorization log stay distributed, our system does not create a central target for content attack—a crucial consideration in an age of cyberattacks and data leaks. Though some blockchains experience robustness challenges from a scaling limit on the "block size" or storage capacity [38], these parameters can be modified to optimize for other performance requirements in a private blockchain network. Notably, MedRec does not claim to address the security of individual provider databases—this must still be managed properly by the local IT system admin. In the same vein, MedRec does not solve end-point security, in that a compromised patient computer or mobile device could potentially open a vulnerability for data theft or

snooping (just as it might for any other software installed on the device). Furthermore, MedRec does not attempt to solve the Digital Rights Management [13] problem of undesired data copying, as our system assumes provider nodes that are bound by external regulation governing data copying in the medical use case, e.g. HIPAA.

On the security of the MedRec codebase itself, we speculate that it may have many weaknesses. Our focus in developing this prototype was to test a functionality premise, and we have not obtained a formal security review of the codebase, as we would if this were to be used in a real, clinical setting. While we have taken common sense steps to avoid known issues (i.e. the use of "parameter" characters in database calls to avoid SQL-injection attacks [39]), there may be other vulnerabilities, particularly in the structure of the smart contracts. As seen with the DAO (introduced in the Ethereum Overview section), simple logic errors in these programmable rule sets can lead to unforeseen consequences. Before MedRec is taken further, we would encourage any interested developers to complete a review with a trained, system architect and security consultant.

4.3 Comments on Privacy

Regarding privacy, use of blockchain technology introduces several limitations. The pseudonymous property of transactions currently allows for data forensics, or inferring patterns of treatment from frequency analysis. Without any disclosure of name or PII, one could infer that some entity has repeatedly interacted with another network entity through analysis of network traffic. Improving obfuscation while preserving auditability on the blockchain is an ongoing area of exploration. One potential solution is to make the blockchain a "permissioned" or private structure, where only pre-approved, white-listed nodes are allowed read access to the ledger. This would prevent rogue actors from extracting frequency-based insights from the blockchain

records, and may be a necessity in the medical context where HIPAA regulations generally lead to closed, secured systems. This may necessitate additional auditing of mining nodes, where the medical researchers are required to run the MedRec mining client on secured systems. The orchestratation of onboarding and auditing these white-listed nodes begs the question of a centralized organization, created to manage these requirements. While certain blockchain proponents would consider this an undesirable compromise in an otherwise decentralized system, we note that society does still place trust in centralized institutions as they have their own merits (e.g. efficiency, simplified access controls, clear maintenance accountability, etc.).

Alternatively, one could consider keeping an open blockchain approach by integrating with or adapting a technology like ZeroCash [40] (which makes use of zero-knowledge proofs) where identities in the system could stay anonymous and return only the minimum amount of information needed to execute a transaction. Additional research currently underway at MIT, the Enigma project [41], also offers a degree of "secret sharing" and privacy preservation via a Multi-Party Computation approach. These may be useful platforms for future applications of blockchain technology, where user privacy proves a key concern. Finally, encryption could be introduced in the MedRec off-blockchain data syncing steps to safeguard against accidental or malicious content access. We note that while privacy is an important consideration for this sytem, a complete lockdown of information would also be counterproductive. A key feature of the MedRec model lies in the data sharing functionality that can ease and improve the process of designating healthcare proxies. This calls for managing contextual privacy—the facility to share with authorized parties, and the protections to keep data private outside of these scenarios.

While outside the scope of the initial prototype (but unarguably crucial for future development), a rigorous k-anonymity analysis [42] of privacy-preserving query construction is needed, for release of the aggregated research data to medical research

"miners".

4.4 Comments on Interoperability

By integrating with providers' existing data storage infrastructure, we facilitate continued use of their existing systems. We believe this will ease adoption and aid compliance with HIPAA regulations. Building on the principle of interoperability, we have designed the system with flexibility to support open standards for health data exchange in the future—be that FHIR (Fast Healthcare Interoperability Resources) or other flavors of HL7 proposals [43]. In addition, MedRec is source agnostic, i.e. able to receive data from any number of endpoints (physician offices, hospital servers, patient home computers, et cetera). MedRec does depend on the prior digitization of medical records. We have developed MedRec not as a proprietary system, but as a set of open APIs to facilitate EHR review and exchange. We hope that MedRec will prove to be a layer that can be added to existing provider backends, thanks to the design of our Database Gatekeeper utility.

4.5 Comments on Scalability

As we learned with the Beth Israel Deaconess test integration, the MedRec project would face certain hurdles in real-world deployment. Some of these are simply limitations of a research prototype that could be addressed with further development (i.e. to bring the project to Enterprise readiness), and others remain more fundamental to the task of orchestrating cross-institutional data retrieval. Below, we consider three interesting dilemmas that raise scalability and deployment challenges. While these are perhaps beyond the scope of the MedRec project to solve, we consider them worthy of consideration for a discussion of the project in a real-world context.

First, how would we envision keeping track of the networking information for

hundreds of thousands of databases, should the MedRec model go "viral"? Because our smart contracts rely on being able to associate pointers to the network location where our database gatekeeper would retrieve the data from, we are assuming that the databases we are trying to connect to will be both networked and have previously authorized us for data retrieval. This raises the need for a separate authentication scheme that would let us handle authentication and data connectivity across a large network of databases, without storing authentication information for the databases in any centralized repository. Perhaps, rather than connecting via individual logins, we would set up a module for end-provider verification of our database gatekeeper module, via a web certificate-like system.

Second, the MedRec system begs the question of a global ID system (which may, depending on philosophical leanings, seem to contradict the beauty of an otherwise decentralized system). The MedRec system needs to be able to associate data pulled from one hospital on Patient Jane Doe, with data pulled from a second hospital on Patient Jane Doe. While we could keep a mapping in our registrar contract of Hospital 1's ID string for Jane Doe and Hospital 2's ID string for Jane Doe, and do the association internal to the MedRec system, it would greatly improve efficiency to be able to recognize some aspect of the Jane Doe identity as the same across both sources. While MedRec does not attempt to solve this larger ID-assignment problem, there are blockchain solutions looking at the option of self-sovereign identity [44] and we look forward to keeping abreast of their progress. Furthermore, though MedRec attempts to simplify the user experience by abstracting away the blockchain and cryptographic functionality from the end-user perspective, there still exists an open question of best practices for private key management at the internal app level. This implicates privacy, in that loss of control over a private key undermines the privacy of the records associated with a particular "identity" in the system. We note Clark et al.'s call for better tools for blockchain key management [45], and envision that such

a tool might be adapted to improve the MedRec system.

Finally, if our MedRec system was deployed and the medical research mining scheme proved to be a large success, we could envision soon having many nodes pinging provider databases with query requests from their successfully mined blocks. This could create a significant load on certain database servers. In response to this, the MedRec system could establish a marketplace where an economic block might limit querying if threatening to rise above a certain threshold or other gatekeeping functions to place a check on the allowable query execution. Furthermore, a limit could be set on the number of whitelisted miners, or the frequency at which miners are allowed to retreive their queries.[3] This method of distributing mining to a large network of medical researchers essentially distributes the cost of MedRec network maintenance (as opposed to the alternative of a centrally managed MedRec-like system, with its own considerable maintenance costs). We remain interested to see whether the summed investment made by all distributed parties would or would not exceed the centrally-managed maintenance costs of a comparable system. If systems like MedRec grew to be public infrastructure services, then the medical researchers' distributed costs could be modeled as essentially passed on to the taxpayers, due to the fact that tax payer money often supports medical research.

[3]We also consider the need to keep internal MedRec calls to the provider databases at a minimum, to avoid similarly overloading these systems.

Chapter 5

Beyond the Lab

5.1 MedRec in the Context of National Healthcare Priorities

As mentioned in the introduction, we do not present MedRec as a panacea nor as the only blockchain-mediated solution that would be needed to achieve our stated goals of data access, patient-empowerment, interoperability and improved medical research. In the analysis below, we refer to MedRec by name to suggest how such a project might address national healthcare priorities, likely as part of a larger suite of blockchain solutions to which we hope to contribute.

Most importantly, the MedRec model restores comprehensive patient agency over healthcare information—across providers and treatment sites, empowering citizens with the data they need to make informed decisions around their care. By giving patients a long-term, trusted log of their information with data sharing functionality built-in, the MedRec system directly addresses the ONC Interoperability Roadmap's first Outcome: "Individuals have access to longitudinal electronic health information, can contribute to the information, and can direct it to any electronic location" [31]. As envisioned by the Precision Medicine Initiative (PMI), the MedRec patient record

could grow to reflect the many facets of health data, by accepting not just physician data, but also data from the patient's Fitbit, Apple HealthKit, 23andMe profile, and more. Patients can build a holistic record of their medical data and authorize others for viewership, such as physicians providing a second opinion or family members and care guardians.

MedRec data could also feed into emerging technologies for predictive analytics, allowing patients to learn from their family histories, past care and conditions to better prepare for healthcare needs in the future. By employing open APIs like MedRec, machine learning and data analysis layers could be added to repositories of healthcare data to enable a true "learning health system" [31]. Due to the linked interoperability between provider databases in a MedRec network, better-unified access to data could facilitate a wide range of trend discovery. MedRec's modularity could support an additional analytics layer for disease surveillance and epidemiological monitoring, physician alerts if patients repeatedly fill and abuse prescription access (e.g. part of the national problem with narcotics abuse [46]), personal dashboards that show patients emerging trends in their own health, etc. In this respect, the MedRec model enables a service-oriented architecture (SOA) as outlined in the ONC Roadmap's "Secure, Standard Services" [31].

MedRec's community model, where medical researchers (and potentially other regulated stakeholders in the healthcare industry) can obtain insightful, population-wide data on medical treatment offers an unprecedented opportunity to achieve goals for precision medicine and evidence-based research. Such a system would facilitate the Patient-Centered Outcomes Research Institute's goals for comparative clinical effectiveness research [47], by linking the patients within a particular clinical cohort with both granular and long-term medical history, thus enabling a better understanding of patient outcomes across treatment groups and over time. By leveraging a data orchestration system like MedRec where the records would already be gathered, organized

and available for analysis, this type of research can be achieved with significantly less overhead than traditional research trials, which often require expensive recruitment procedures and in-person access to patients. This ability to carry out longitudinal studies on MedRec user cohorts directly addresses both the ONC Interoperability Roadmap stated Outcomes [31] and the PMI's goal for a national research cohort [48].

The MedRec smart contract structure serves as one model for a "Health Care Directory and Resource Location," secured with public key cryptography and enabled with crucial properties of provenance and data integrity. This blockchain directory model supports the ability to "grow and change dramatically throughout its lifetime—adding new participants and changing organizational relationships" through stateful updates to the smart contracts [31]. A blockchain log could provide clarity for communicating authorization "across the Health IT ecosystem," and an audit log for subsequent inquiries into use of such permissions and access patterns. With this functionality, the system would serve as a "Consistent Representation of Authorization to Access Electronic Health Information" [31].

Fundamentally, the MedRec project strives to enable Precision Medicine and holistic understanding of patient medical status without creating a centralized repository of data. Centrally-stored data has often proved disastrous in our modern age of cyberattacks and data leaks. Therefore, MedRec leverages a decentralized, blockchain architecture to enable local, separate storage but coordinated viewing of the data from the patient perspective. We believe MedRec fits squarely in the White House's goals for the ONC to "support the development of interoperability standards and requirements that address privacy and enable secure exchange of data across systems" [49]. Because MedRec is a system of open APIs, we hope the project will integrate with other key layers in the healthcare IT stack of the future. [1]

[1] Chapter 5 is drawn from our previously published techincal publications [11], as explained in footnote 1.

Chapter 6

Future Work

6.1 Taking MedRec Forward

As we consider what it would take to bring MedRec from a research prototype to a meaningful tool for enterprise, government and patient use, we have identified several thrusts of future work. First, one would need to continue the process of actively engaging with healthcare stakeholders across the industry, from hospitals and provider offices, to pharmaceutical companies, to insurance companies, to healthcare startups, U.S. Government institutions and more. Throughout the summer and fall of 2016, we gathered functionality requirements and additional use-case scenarios from the Department of Veterans Affairs, Kaiser Permanente, Merck & Co., Beth Israel Deaconess Medical Center, Deloitte and others to inform future redesigns of the MedRec system.

Though the MedRec backend is already designed to be flexible with multiple database architectures, an analysis of how to meet custom integration requirements for InterSystems Cache technology [1] would also be a worthy next step. Our goal was to make MedRec an interoperability layer that can be seamlessly added to existing

[1] As mentioned above, InterSystems technology underpins many hospital backends across the nation and supports EPIC's record management platform.

EPIC, Cerner, et cetera deployments, building on the open standards development collaboration "Sync for Science" between the NIH and ONC [50].

Due to the hardforks in Ethereum since our adoption of the codebase in November 2015, MedRec's current Ethereum clients are out of date. Though the MedRec system can run on our internal prototype VMs, it would be necessary to migrate the project to the latest version of Ethereum for the code to work reliably going forward. We are in a sense waiting for a better blockchain to take the project forward—either a more stable version of Ethereum than what is currently available (as of writing, January 2016) or a side-chain technology (like the Lightning Network [51]) that would allow us to re-architect the project on Bitcoin's scripting language, but without bloating the Bitcoin blockchain. Another option might be to pursue integration with Hyperledger [52], a rapidly growing consortium of open-source blockchain technologies. We note the collaboration between Hyperledger, IBM and Walmart on a proposed supply-chain application of blockchain [53] as a project that may have worthwhile learnings for a future MedRec implementation at scale.

To map MedRec's position in the world of blockchains, we use the axes recently posited by Neha Narula as a blockchain taxonomy in Figure 6-1 below: from closed to open systems, and from systems that track exchange of data to exchange of value. Interestingly, the chart was originally empty in the Open and Data quadrant, raising a question of whether such a blockchain technology could remain truly open without having a closed-loop economic incentive system (i.e. a cryptocurrency). We place MedRec in this quadrant primarily to provoke thought on which quadrant it ought to travel to next, should the current limitations of being an Open and Data system prove infeasible in the medical context. Going forward, the MedRec project might take two trajectories: go left into the Closed and Data, regulated domain as more of a distributed ledger system than a blockchain; go up into the Open and Value quadrant with the advent of a better privacy preserving tool like ZeroCash [40] or Enigma [41].

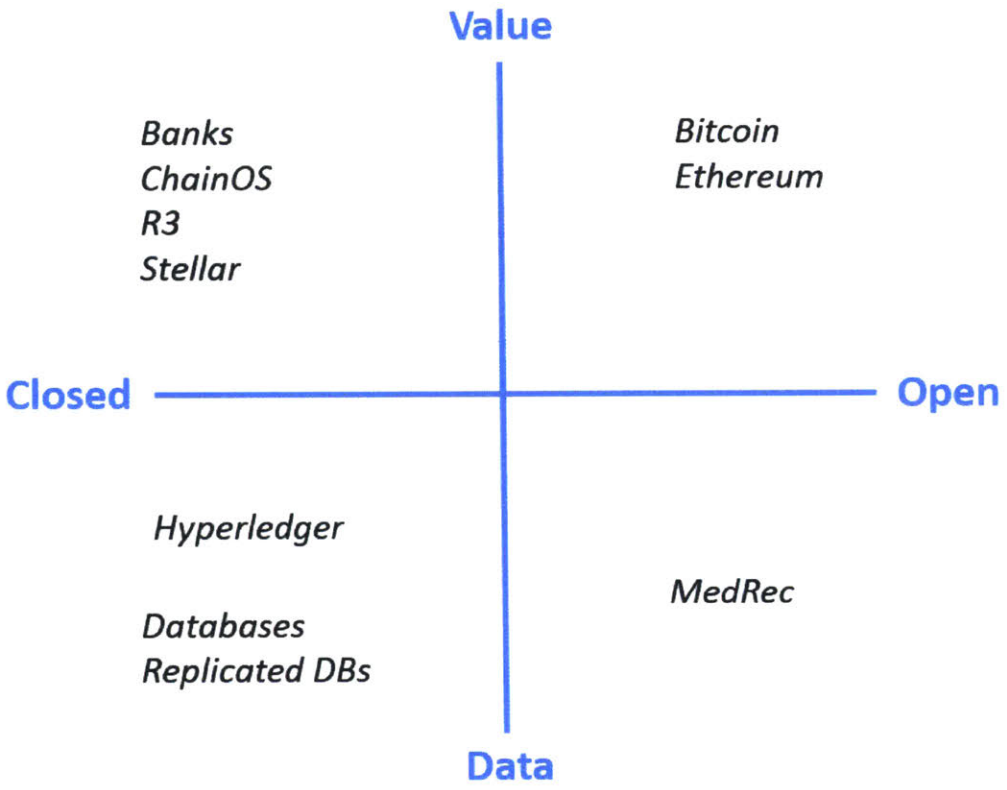


Figure 6-1: This quadrant chart attempts to place existing blockchain projects, like Bitcoin, Ethereum, HyperLedger, etc. into a taxonomy of Value vs Data, Open vs Closed. Credit to Neha Narula for the original axes [3].

6.2 Extensibility Beyond Healthcare

Though much of the use-case analysis in the MedRec project has focused on healthcare applications, we identify certain extensible elements of the MedRec codebase that could be used for other record-keeping purposes. In its simplest form, MedRec is an access and permission log, and a directory look-up tool for pointers to off-blockchain data. This immediately suggests applicability in scenarios when a certain category of data, of interest to a particular individual or research question, is naturally scattered across many disparate sources: land registries, domain names, educational records,

etc.[2] In addition, MedRec may offer a new model for social networks, where smart contracts define relationships between users, and users can retain agency over the management of their data (pictures, posts, etc.) via pointer registries mapped to their permissioning preferences. The distributed users, rather than a set of centralized corporate servers, could act as the stewards and messengers for this data.

[2]Many of these opportunities for application of blockchain tech are currently being explored by for-profit companies. The venture capital landscape around blockchain tech has flourished in recent years.

Chapter 7

Conclusion

7.1 Thesis Contributions

This thesis describes a technical design architecture, the codebase behind an early research prototype and the patient-centered approach that motivates the work. Through the course of the MedRec project, we have published our technical approach in IEEE, documented our social and technical analysis of the project in an ONC-selected whitepaper, and successfully tested the prototype with Beth Israel Deaconess Medical Center in Boston.

7.2 Reflections and Impact

The MedRec prototype provides a proof-of-concept backend system that demonstrates how principles of decentralization and blockchain architectures can contribute to secure, interoperable EHR systems. Using Ethereum smart contracts to orchestrate a content-access system across separate storage and provider sites, the MedRec authentication log governs medical record access while providing patients with comprehensive record review, care auditability and data sharing. We demonstrate an innovative approach for integrating with providers' existing systems, prioritizing open APIs and

network structure transparency. This approach was validated with an on-site pilot at Beth Israel Deaconess Medical Center. We look forward to sharing the ideas behind the MedRec project infrastructure, following the ONC's call for policy and technical components of an interoperable health IT stack. The MedRec project offers a new perspective and toolset for how we can empower patients to engage in and track the details of their healthcare, thus restoring personal agency over personal data.

Appendix A

Publications, Presentations, Patent and Awards

A.1 Publications

In order of publication:

- Azaria, Asaph, Ariel Ekblaw, Thiago Vieira, and Andrew Lippman. "MedRec: Using Blockchain for Medical Data Access and Permission Management." In Open and Big Data (OBD), International Conference on, pp. 25-30. IEEE, 2016.

- Ariel Ekblaw, Asaf Azaria, Thiago Vieira, Andrew Lippman. "MedRec: Medical Data Management on the Blockchain". PubPub, (2016). [https://www.pubpub.org/pub/medrec] version: 57e013615dbf3f3300152554

- Ekblaw, Ariel, Asaph Azaria, John D. Halamka, and Andrew Lippman. "A Case Study for Blockchain in Healthcare: 'MedRec' prototype for electronic health records and medical research data." Whitepaper. Office of the National Coordinator for Health Information Technology, Department of Health and Human

Services (2016).

A.2 Presentations

- 2nd International Conference on Open and Big Data. Vienna, Austria. August 2016.
- Life Sciences and Healthcare Blockchain Workshop, MIT Media Lab. Cambridge, Massachusetts. September 2016.
- HUBWeek @ the Federal Reserve Bank of Boston. Boston, Massachusetts. September 2016.
- ONC/NIST Use of Blockchain for Healthcare and Research Symposium. Gaithersburg, Maryland. September 2016.

A.3 Patent

- Blockchain System for Management of Electronic Medical Records. Provisional. Receipt Date 18 May 2016.

A.4 Award

- MedRec whitepaper selected a "Blockchain Challenge Winner" by U.S. Department of Health and Human Services Office of the National Coordinator for Health Information Technology.

Appendix B

Beth Israel Deaconess Integration Documents

MIT Media Lab - MEDREC
Test Deployment with Beth Israel Deaconess Medical Center

Technology: MedRec uses blockchain smart contracts to create a decentralized content-management system for EMRs, across separate storage instances and providers. The MedRec authentication log governs medical record access, while providing means for auditability and data sharing. The MedRec blockchain stores data pointers, permission rights and access logs, while sensitive EMR content remains stored in existing provider databases. A web app and backend APIs orchestrate communication between the user, blockchain and provider databases. For a full description, see pubpub.org/pub/medrec.

Parties:

MIT Media Lab student developers—Ariel Ekblaw and Asaf Azaria

Beth Israel Deaconess Medical Center—Dr. Lawrence Markson, Mike Yamamoto, Venkat Jegadessan, Steve Diorio, Steve Cinella, Ayad Shammout

Proposed scope for this deployment:

- Two instances with SQL Server 2014 (on BID test systems, not prod) populated with de-identified medication data. BID supplies SQL Server login (username/password). Minimum: Read access; Write access could be useful for record updating scenarios, if able to be securely given and separated from prod systems.
- MedRec "Database Gatekeeper" utility tests record exchange between the two instances, via blockchain permissioning protocol. The Software will be supplied on MIT VMware VMs, running on an MIT-supplied machine, to be connected to the isolated network set up at BIDMC Lab.
- Goal: Test end-to-end flow from batch pickup of existing records, into the sample "clinical data warehouses," successful pointer posting to blockchain, permission update, and retrieval by second party (aka second data warehouse instance). As time permits, test entry of individual new record and repeat end-to-end flow.
- Includes test of locally deployed web app UI (no external internet connection)
- BIDMC Data preparation: anonymized IDs will replace existing patient, physician/prescriber and record IDs, while keeping the logic between linked or repeated entries. All PII columns (Name, Age, address, etc) will be removed.

Timeline:

Two or three onsite appointments, week of August 15th. Infrastructure group will be present for initial setup and coordination.

Preparation Step:

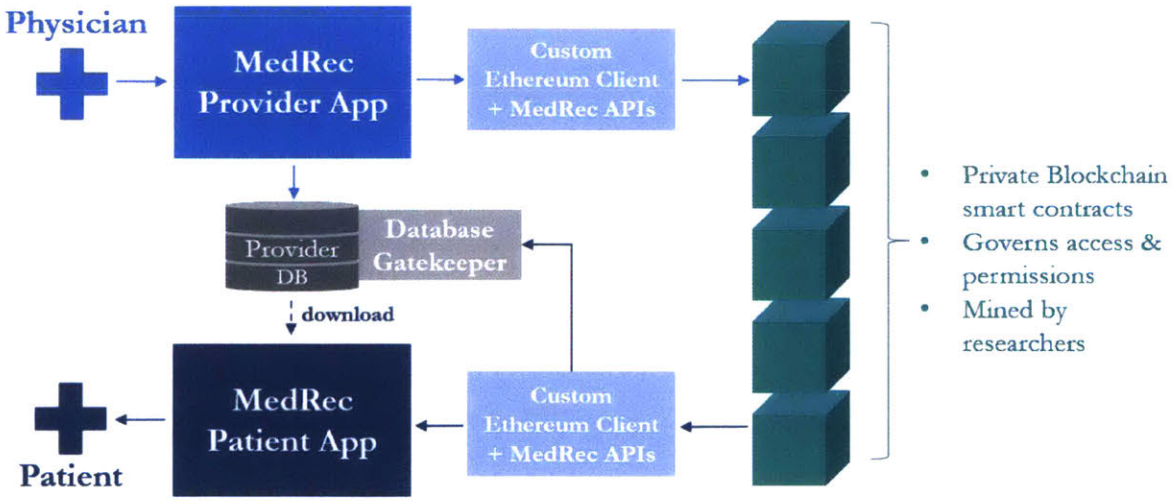
- Defining data use-cases; extracting medication data with interesting use-case features (Dr. Markson & Ariel Ekblaw)

Security Steps:

- If a second day of integration testing is required, the MIT laptop with the provided VMs will be left overnight within the closed, secured testing area.
- Upon completion of the integration testing, the MedRec project VMs will be deleted to ensure that no data leaves the boundary of the BIDMC network.

MedRec System Diagram:

Single node:



Modifications for this scope:

- Provider DB will be prefilled with data of interest (need not go through the manual physician entry step aka top left corner, unless additional testing is desired)

- Patient retrieval of data (bottom row) will be modeled by the second SQL instance, testing receipt of data after permission updates.

MIT Media Lab - MEDREC

Test Deployment with Beth Israel Deaconess Medical Center

Designing the Test Integration Dataset

Preliminary steps:

- Remove any PII information
- Replace actual Patient/Prescriber keys with new, random to us, numeric IDs
- Replace actual record keys with new, random to us, numeric IDs
- Maintain the logic of connected records (i.e. a patient ID, even though now represented by a new random number, should be repeated on any records that correlate to that same patient)
- Additional privacy consideration #1: If any patients are on particularly rare medications, where it might be possible to re-identify just based on medication name, we should likely exclude these records as well. So if the Medication dataset could be kept to patients that use relatively common prescriptions, that would help as an additional privacy check.
- Additional privacy consideration #2: Being able to test the code on a free-form text field would be helpful, but if we do keep this type of field, we should be careful that none of the material written in by the doctor would uniquely identify the patient.
- Split the selected data among the two SQL Server instances, preferably without repeats, to simulate two different provider backend repositories

Desired Dataset Features:

- Patients with a single medication record, single prescriber (basic test)
- Patients with two or more medications under the same prescriber (to test multiple med records associated with the same patient-provider relationship contract)
- Patients with multiple medications, but under different prescribers (to test association of a single patient's summary contract with two or more distinct patient-provider relationship contracts, with at least one real med record in each).
- Patients where we split their medication records across the two SQL Server instances, to simulate them having seen two different prescribers at two different institutions. So this example would feature patient A having at least one unique medication record in DB #1 and at least one different, unique medication record in DB #2 (with a different prescriber).
- Patients with updated med records, where certain fields, like dosage, may change but the patient-prescriber relationship and medication is the same
- Patients with updated med records, where the prescriber changed (to simulate change of care role) but medication remained the same, with or without other dosage changes.

[Here, we'll need to see if the unique record ID is changed, suggesting a net-new record or kept, suggested logic of just an update. Our backend will likely treat this as net-new, and potentially lose the context, since it's a new provider and our system contracts are patient-provider based. Will need to explore further.]

- Beyond the explicitly defined records designed here, could we bring the total up to 100-200 records over about 6 months for a full dataset? The rest can be semi-randomly selected, after covering for the specific cases above and meeting the privacy requirements. A lower total record count is probably fine, depending on whatever is reasonable for BIDMC!

Columns to keep from CREATE statement:

Period: January - June, will get some records that are discontinued but without a start date (i.e., started prior to January)

Add column: Transaction date will tell us the date that the transaction occurred, like timestamp from audit log ([audit_dt] [datetime])

Crossed-through columns from the original list have been removed

Bibliography

[1] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Whitepaper*, 2008.

[2] Go-ethereum. Ethereum Project Github. [Online] Available: https://github.com/ethereum/go-ethereum Accessed January 17, 2017.

[3] Neha Narula Director MIT Digital Currency Intiative. Internet & society course: Cryptocurrency slides. Lecture Notes, January 5, 2017.

[4] Who owns medical records: 50 state comparison. Health Information and the Law, George Washington University Hirsh Health Law and Policy Program, August 20 2015. [Online] Available: http://www.healthinfolaw.org/comparative-analysis/who-owns-medical-records-50-state-comparison.

[5] U.S. Department of Health and Human Services. Hipaa administrative simplification. 45 CFR Parts 160, 162, and 164, 2013. [Online] Available: http://www.hhs.gov/sites/default/files/hipaa-simplification-201303.pdf.

[6] Kenneth D. Mandl et al. Public standards and patients' control: how to keep electronic medical records accessible but private. *BMJ*, 322(7281):283–287, 2001.

[7] Office of the National Coordinator for Health Information Technology. Report on health information blocking. Report to Congress, 2015. [Online] Available: https://www.healthit.gov/sites/default/files/reports/info_blocking_040915.pdf.

[8] U.S. Department of Health and Human Services. Individuals' right under hipaa to access their health information 45 cfr 164.524. [Online] Available: http://www.hhs.gov/hipaa/for-professionals/privacy/guidance/access/ Accessed: Aug. 8, 2016.

[9] Claudia Grossmann et al. *Clinical Data as the Basic Staple of Health Learning: Creating and Protecting a Public Good.* Workshop Summary (Learning Health System Series). National Academies Press: Institute of Medicine of the National Academies, 2010.

[10] Leonard J Kish and Eric J Topol. Unpatients [mdash] why patients should own their medical data. *Nature biotechnology*, 33(9):921–924, 2015.

[11] Ariel Ekblaw, Asaph Azaria, John D Halamka, and Andrew Lippman. A case study for blockchain in healthcare:"medrec" prototype for electronic health records and medical research data. Whitepaper, 2016. [Online] Available: https://www.healthit.gov/sites/default/files/onc_blockchainchallenge_mitwhitepaper_copyrightupdated.pdf.

[12] Steve Lohr. The healing power of your own medical records. The New York Times, March 31, 2015. [Online] Available: https://www.nytimes.com/2015/04/01/technology/the-healing-power-of-your-own-medical-data.html?_r=0.

[13] Digital rights management and libraries. American Library Association. [Online] Available: http://www.ala.org/advocacy/copyright/digitalrights Accessed January 17, 2017.

[14] Markus Jakobsson and Ari Juels. Proofs of work and bread pudding protocols. In *Secure Information Networks*, pages 258–272. Springer, 1999.

[15] Adam Back et al. Hashcash-a denial of service counter-measure, 2002.

[16] Ralph C Merkle. Protocols for public key cryptosystems. In *IEEE Symposium on Security and privacy*, volume 122, 1980.

[17] Blockchain.Info. Market price (usd): Average usd market price across major bitcoin exchanges, 2017. [Online] Available: https://blockchain.info/charts/market-price?timespan=all&showDataPoints=false&daysAverageString=1&show_header=true&scale=0&address= Accessed January 16, 2017.

[18] Bitcoin Project. 51% attack, majority hash rate attack". [Online] Available: https://bitcoin.org/en/glossary/51-percent-attack Accessed January 17, 2017.

[19] Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper*, 2014.

[20] What is ethereum. Ethereum Project Github. [Online] Available: https://github.com/ethereum/wiki/wiki/What-is-Ethereum Accessed January 17, 2017.

[21] William Mougayar. The business imperative behind the ethereum vision, May 24 2015. [Online] Available: https://blog.ethereum.org/2015/05/24/the-business-imperative-behind-the-ethereum-vision/.

[22] Choosing a client. Ethereum Homestead Documentation. [Online] Available: http://www.ethdocs.org/en/latest/ethereum-clients/choosing-a-client.html Accessed January 17, 2017.

[23] Ethereum frontier release. Ethereum Gitbooks. [Online] Available: https://ethereum.gitbooks.io/frontier-guide/content/frontier.html Accessed January 17, 2017.

[24] John Halamka. On the road to rhios. *Health Manag Technol*, 27(6):8, 2006.

[25] John D Halamka, Kenneth D Mandl, and Paul C Tang. Early experiences with personal health records. *Journal of the American Medical Informatics Association*, 15(1):1–7, 2008.

[26] Intersystems unveils major new release of cache. InterSystems, February 25, 2015. [Online] Available: http://www.intersystems.com/who-we-are/newsroom/news-item/intersystems-unveils-major-new-release-cache/.

[27] Health Level Seven INTERNATIONAL. Cda release 2. [Online] Available: http://www.hl7.org/implement/standards/product_brief.cfm?product_id=7 Accessed January 17, 2017.

[28] An update on google health and powermeter. GOOGLE official blog, June 24 2011. [Online] Available: https://googleblog.blogspot.com/2011/06/update-on-google-health-and-google.html.

[29] Healthkit: Develop health and fitness apps that work together. Apple Inc. [Online] Available: https://developer.apple.com/healthkit/ Accessed January 17, 2017.

[30] Fitbit app. FitBit Inc. [Online] Available: https://www.fitbit.com/app Accessed January 17, 2017.

[31] Office of the National Coordinator for Health Information Technology. Connecting health and care for the nation: A shared nationwide interoperability roadmap, 2015. [Online] Available: https://www.healthit.gov/sites/default/files/hie-interoperability/nationwide-interoperability-roadmap-final-version-1.0.pdf.

[32] Guy Zyskind et al. Decentralizing privacy: Using blockchain to protect personal data. In *Security and Privacy Workshops (SPW), 2015 IEEE*, pages 180–184. IEEE, 2015.

[33] U.S. Department of Health and Human Services. Onc anounces blockchain challenge winners, September 1 2016. [Online] Available: http://www.hhs.gov/about/news/2016/08/29/onc-announces-blockchain-challenge-winners.html.

[34] Requesting medical records? we'll get them for you. Health Exchange Technologies, Inc. PatientBank, 2016. [Online] Available: https://www.patientbank.us/#features Accessed January 18, 2017.

[35] U.S. Department of Veterans Affairs. The patient record: health design challenge. ONC & Dept. of VA Blue Button Challenge, January 2013. [Online] Available: http://healthdesignchallenge.com/.

[36] Pymssql python package. PYMSSQL Documentation. [Online] Available: http://pymssql.org/en/stable/ Accessed January 17, 2017.

[37] U.S. Department of Health and Human Services. 2015 edition health information technology (health it) certification criteria, 2015 edition base electronic health record (ehr) definition, and onc health it certification program modifications. Regulation 80 FR 62601. [Online] Available: https://www.federalregister.gov/documents/2015/10/16/2015-25597/2015-edition-health-information-technology-health-it-certification-criteria-2015-edition-base Accessed January 18, 2017.

[38] Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, and Emin Gün. On scaling decentralized blockchains. In *Proc. 3rd Workshop on Bitcoin and Blockchain Research*, 2016.

[39] Sql injection. W3Schools. [Online] Available: http://www.w3schools.com/sql/sql_injection.asp Accessed January 17, 2017.

[40] Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474. IEEE, 2014.

[41] Guy Zyskind, Oz Nathan, and Alex Pentland. Enigma: Decentralized computation platform with guaranteed privacy. *arXiv preprint arXiv:1506.03471*, 2015.

[42] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.

[43] Fhir overview. HL7 International, October 2015. [Online] Available: https://www.hl7.org/fhir/overview.html Accessed January 17, 2017.

[44] Christopher Allen. The path to self sovereign identity. CoinDesk, April 27, 2016. [Online] Available: http://www.coindesk.com/path-self-sovereign-identity/.

[45] Shayan Eskandari, David Barrera, Elizabeth Stobert, and Jeremy Clark. A first look at the usability of bitcoin key management. In *Workshop on Usable Security (USEC)*, 2015.

[46] About the[opiod] epidemic. U.S. Department of Health and Human Services. [Online] Available: http://www.hhs.gov/opioids/about-the-epidemic/ Accessed January 17, 2017.

[47] Research we suport. Patient-Centered Outcomes Research Institute (PCORI). [Online] Available: http://www.pcori.org/research-results/research-we-support Accessed January 17, 2017.

[48] Precision medicine initiative cohort program. National Institutes of Health. [Online] Available: https://www.nih.gov/precision-medicine-initiative-cohort-program Accessed January 17, 2017.

[49] Fact sheet: President obama's precision medicine initiative. The White House Briefing Room, January 30, 2015. [Online] Available: https://www.whitehouse.gov/the-press-office/2015/01/30/fact-sheet-president-obama-s-precision-medicine-initiative.

[50] Fact sheet: Obama administration announces key actions to accelerate precision medicine initiative. The White House Breifing Room, February 25, 2016. [Online] Available: https://www.whitehouse.gov/the-press-office/2016/02/25/fact-sheet-obama-administration-announces-key-actions-accelerate.

[51] Joseph Poon and Thaddeus Dryja. The bitcoin lightning network: Scalable off-chain instant payments. Technical report, Technical Report (draft). https://lightning. network, 2015.

[52] Hyperledger: Blockchain technologies for business. Hyperledger/Linux Foundation. [Online] Available: https://www.hyperledger.org Accessed January 17, 2017.

[53] Walmart, ibm and tsinghua university explore the use of blockchain to help bring safer food to dinner tables across china. IBM News Room, 2016. [Online] Available: https://www-03.ibm.com/press/us/en/pressrelease/50816.wss Accessed January 18, 2017.