Distributed Ledgers for Data Privacy: Student Data and Afterschool Programs in Chicago

Alden Golab, Paul Mack, Emily Webber

University of Chicago

May 30, 2017

Abstract

Current methods for obtaining data consent from legal guardians are limited to paper forms and human data entry. While this classic method of granting consent is familiar to the parties involved, it involves little accountability or ongoing access oversight. Parents are asked to sign multiple permission forms, but are not given an efficient manner by which they can differentiate between fields for consent nor keep track of usage. Blockchain technology represents a promising approach for developing "smart contracts," digital promises to be executed without reliance upon a legal intermediary. Utilizing Hyperledger Fabric and Composer, we propose a framework of nested authorization, allowing a public school data administrator the ability to grant authorization rights to a third party after-school intervention program and their volunteers, who meet with parents and obtain consent for data access rights relying on smart contracts. These data consent signatures allow researchers the ability to design targeted inventions, meeting the needs of the community.

1 Introduction

When it comes to social science research using administrative data, there is often a substantial barrier imposed by issues of informed consent from research participants. In particular, recent academic discussion has suggested one-time consent agreements for data use are often insufficient means of obtaining *informed consent* overtime, where consent is obtained from an individual who is (a) in possession of adequate reasoning capacity and (b) in possession of all relevant facts surrounding the use of their data.[1]

Institutional Review Boards (IRB) responsible for the ensuring ethical behavior for research involving human subjects generally require that researchers document participants' informed consent. Typically, paper forms that outline the purpose of the study and how a participant's data will be used are signed by the participants and a record of their consent is digitally recorded based on receipt of the paper form.

We believe introducing a digital ledger into this collection process could substantially improve how authorization information is collected and stored. A digital process could benefit researchers by reducing staff time spent digitally transcribing paper records, automatically creating an immutable log of consent agreements with timestamps, and enforcing IRB required data expirations. Furthermore, a digital process could benefit research participants by providing a digital reference for them to monitor their consent over time, a way to control their consent on a field by field basis, and a means to easily revoke their consent. By distributing the system between multiple nodes in a trustless system, we are able to concurrently improve the overall trust of participants and researchers in the process.

[1] The Belmont Report (1979).

1.2 Related Work

Similar work in this field has been developed by MIT's Media Lab[2], where researchers built a decentralized record management system to handle electronic medical records utilizing distributed blockchain technology. Throughout the course of a medical patient's life, he or she will leave medical records scattered throughout the ecosystem of various health care provider's data silos. Requests for data from these providers can take up to 60 days to comply, according to the HIPPA Privacy Rule.

In this work, they explore the application of public key cryptography to create append-only, immutable, timestamped chains of consent. Copies of these consent blocks are distributed to each node that is participating in the network. The researchers develop a Proof of Work algorithm that is used to protect the contracts from tampering relies upon a "trustless" model, whereby individual nodes compete with each other to solve computationally-intensive hashing exercises, also known as "mining." Relying on the Ethereum development environment, the researchers demonstrate the successful use of smart contracts in a medical application.

They deploy three contracts: Registrar Contract (RC), a Patient-Provider Relationship Contract (PPR), and a Summary Contract (SC). The RC maps participant identification strings to their Ethereum address identity, which is the same as their public key. The PPR contract is issued between two nodes in the system, where one nodes stores and manages the medical records for the other node through SQL queries. Finally the SC functions as a trace of participants in the system to locate their medical record history. The SC holds a list of references to PPRs, which represents all participants' previous and current engagements with other nodes in the system.

While providing a trail of contract history, our implementation is slightly different in that it

[2] Azaria et. al (2016)

does not rely on Summary Contracts. Instead, we keep records of transactions with the appropriate party, for example a student record holds all of its associations and consent submissions. This is similar to a Summary Contract, but because we rely on Hyperledger instead of Ethereum, the record is considered an asset, which is unique to Hyperledger.

Furthermore, our implementation solves the problems of nested authorization and granular consent for data release.

1.3 Overview

In Section 2, we outline the problem and motivating example from Thrive Chicago and our overall approach. In Section 3, we discuss the system architecture at a high level then discuss the particulars of this application of the Hyperledger system and a high level overview of the algorithms developed. In Section 4, we go through a security evaluation of the current approach. Finally, we discuss our conclusions and next steps in Section 5.

2 Application

Thrive Chicago is part of a national network of afterschool program providers for public school students in large urban centers. Data collection and program evaluation are an essential part of Thrive Chicago's work, but the current methods for obtaining informed parent consent for the use of children's are laborious, relying on volunteers armed with paper consent forms and pens every year. Signed forms are then entered by hand into digital form and sent to the school's data manager when data are requested for an IRB-approved study. Unsurprisingly, Thrive Chicago tends to see approval rates under 50% for parental data consent.

2.1 Approach

We implement a smart contract solution that would allow for Thrive Chicago's process of approval to be implemented digitally on distributed ledger, or blockchain. This provides a variety of benefits, notably:

- An immutable ledger of how data access has been approved, by whom, and for whom, easily accessible by the parent.
- Parental control over revocation of access in the event of student un-enrollment or worry about misuse of the data by third parties.
- Potentially more granular parental control over which fields are shared.
- Automatically enforced restrictions on time horizons, e.g. for time-restricted IRB approvals.
- High level of security for student data access outside of the system, with corresponding guarantees, using RSA Public Key encryption.
- A user-friendly digital interface for all data access controls.

In particular, replication of the current paper form based system requires thoughtful attention to the way in which the unspoken authorization networks are created by having volunteers (a) confirm the parent/guardian's identity, usually through checking an ID and (b) giving the form for the parent to sign on behalf of the IRB-approved study investigators.

3 System Architecture

The Linux Hyperledger Project[3] is a collection of projects that implement private, distributed ledger networks. The system is relatively robust, despite its youth, with extensive documentation and an active community of developers. Recently, Hyperledger Composer was added to the Hyperledger ecosystem, which automates a large portion of the chain-code necessary to implement a Hyperledger Fabric instance, their version of a

[3] See: https://www.hyperledger.org/. Accessed 6/1/17.

blockchain.[4][5][6] We use the Composer system to construct this private, distributed network via their Playground browser GUI, which runs a stubbed test network.

Uniquely, Hyperledger has three types of peer nodes within a given network: consenters, endorsers, and committers. This allows for the creation of channels, or unique peer networks, that can limit a given transaction's execution to a subset of the network, then farm out privacy-preserving consensus endorsement mechanisms to the broader network. Fundamentally, this allows for the broader idea that not all nodes need to or should store all the transactions on the chain. The resulting system includes relatively granular membership controls for things like Read/Update access that are ideally suited for this implementation.

3.1 Implementation

The Hyperledger Composer system requires three inputs: a model file, a functions file, and a permissions specification. While the functions file is relatively standard javascript, the business model and permissions file are written in Composer specific languages.

3.2.1 Model

Our use case is designed for acquiring data consent in a distributed manner by relying on a network of volunteers to meet with the data owners in person and complete the consent process. This approach mirrors how data consent is collected via paper forms today and replicates the trust that is established through face to face interaction.

We have designed a model with several types of entities to facilitate consent collection via a hierarchy where entities can act on behalf of one another. Figure 1 demonstrates the authorization process and access process which allow volunteers to act on behalf of other entities, both to associate a parent with a student record and to initiate a parental data use consent contract.

A note on terminology, a Data Manager (DM) is an entity that is the steward for the off chain resource. In our case this is the database administrator at Chicago Public Schools who manages student records. A Data User (DU) is an entity that needs consent agreements to access the off chain resource such as a nonprofit running an after school program. An Authorized Agent (AA) is an entity who is deputized by either another AA with administrative abilities to create other volunteers, or by a DU who needs them to request consent agreements on their behalf with an Associated Entity (AE). An AE owns the rights to the off chain resource managed by the DM, in this case a legal guardian responsible for deciding how a student's records may be used.

Figure 1 demonstrates how a DM's authority to associate parents with particular student records flows through one or more nonprofit administrators and the volunteers these nonprofit administrators create who perform the student to parent associations in the field. It also demonstrates how a DU has the ability to request data access from a parent by outsourcing consent requests responsibilities to volunteers.

3.2.2 Permissions Structure

Limited Access is enforced via a permissions file in Fabric Composer. This limits read, create, update, and delete access to entities and assets.

In our implementation, a parent (AE) only has read access to student records (the asset) to which they have been associated. A volunteer (AA) can only create other volunteers (AA) if they have been given administrator privileges by Chicago Public Schools (DM). A nonprofit (DU) can only access volunteers that are affiliated with their organization.

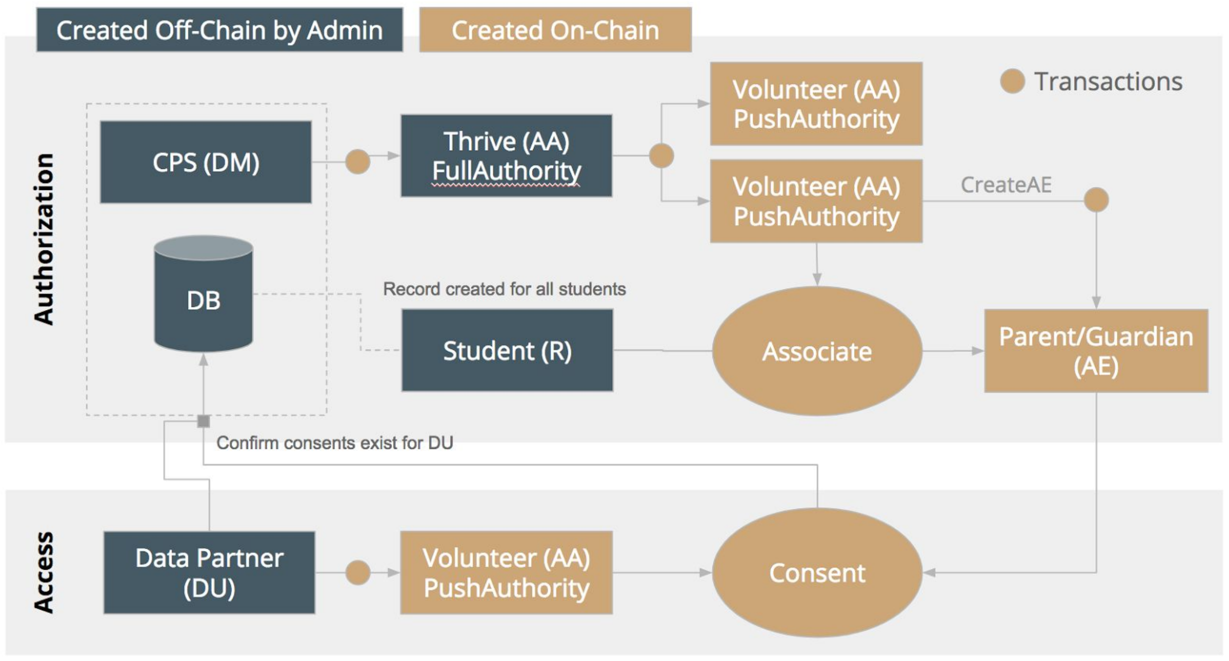
By limiting the access to resources different entities have, we limit the transactions they can perform and enforce the security of the system.

[4] For more on Hyperledger Composer, see: https://www.hyperledger.org/projects/composer Accessed 6/1/17.

[5] Chaincode refers to code written directly on the Fabric instance, generally in Go language.

[6] For more on the Hyperledger project, see: https://hyperledger-fabric.readthedocs.io/en/latest/. Accessed 6/1/17.

Figure 1: Visualization of the Thrive Chicago data authorization and access systems


Created Off-Chain by Admin
Created On-Chain
Transactions
Authorization
CPS (DM)
Thrive (AA) FullAuthority
Volunteer (AA) PushAuthority
Volunteer (AA) PushAuthority
CreateAE
DB
Record created for all students
Student (R)
Associate
Parent/Guardian (AE)
Confirm consents exist for DU
Access
Data Partner (DU)
Volunteer (AA) PushAuthority
Consent

3.2 Function Algorithms

There are three primary algorithms within our implementation that have some variation between specific instances. Here, we outline their general structure and operation.

In Hyperledger, each transaction function takes a transaction object as a parameter; this transaction object is immutable and, once the function is completed, will be stored on the ledger. The primary purpose of functions are to check that the transaction initiator is specified in the transaction and then check the chain for previous transactions that allow for the current transaction to occur. Transactions are designed to be 'all or nothing', such that only all actions or no actions are completed; this is not reflected in the algorithm structure here, but is reflected in the Appendix Code by nesting all chain updates within returns.

Each function returns to the system a promise, which is a cryptographic guarantee that the updates, additions, or other chain work has been completed.

3.2.1 General Authorizations

We outline a general version of the Authorization function below in Algorithm 1.

Algorithm 1 Authorization

```
procedure AUTHORIZE($authorization$)                         ▷ Takes transaction
    if $authorization$ has $initiator$ then
        $initiator \leftarrow authorization.initiator$
    if $authorization$ has $manager$ then
        $initiator \leftarrow authorization.manager$
    $agent \leftarrow authorization.agent$
    $authRoot \leftarrow authorization.authRoot$
    if $initiator \neq currentUser$ then throw $Error$
    if $initiator.authRoot \neq authRoot$ then throw $Error$
    append $authorization$ to $agent.authorizations$
    $promise \leftarrow$ update $agent$ to $agentRegistry$
    $promise \leftarrow$ write $authorization$ to $authorizationRegistry$
    return $promise$                                        ▷ generated by updates
```

There are two important guarantees in this algorithm: (a) the current user, that is, the user initiating the function, must be listed as 'initiator' or 'manager' in the transaction; (b) the initiator must have the same 'authRoot' as the transaction. This prevents someone other than an AuthorizedAgent or DataManager carrying out this transaction and prevents the initiator from giving another user any authorization other than one they current have, thus guaranteeing a chain of authorizations from the same root.

Algorithm 2 Association

```
procedure ASSOCIATE($association$)                                  ▷ Takes transaction
    $parent \leftarrow association.associatedEntity$
    $student \leftarrow association.record$
    $associator \leftarrow association.authorizedAgent$
    if $associator \neq currentUser$ then throw $Error$
    $verifiedRoot \leftarrow student.authRoot$
    $authorizations \leftarrow associator.authorizations$           ▷ Transaction list
    for all $authorizations$ do
        $authRoot \leftarrow authorization.authRoot$
        if $action = associate \land authRoot = verifiedRoot \land unexpired$ then
            $authorized \leftarrow true$
            $break$
    if $authorized = true$ then
        append $association$ to $student.associations$
        append $student.id$ to $parent.students$
        $promise \leftarrow$ write $association$ to $associationRegistry$
        $promise \leftarrow$ update $parent$ to $parentRegistry$
        $promise \leftarrow$ update $student$ to $studentRegistry$
    return $promise$                                                ▷ generated by updates
```

Algorithm 3 Consent

```
procedure REQUESTCONSENT($consent$)                                 ▷ Takes transaction
    $agent \leftarrow consent.agent$
    if $agent \neq currentUser$ then throw $Error$
    $dataUser \leftarrow consent.user$
    $student \leftarrow consent.student$
    $guardian \leftarrow consent.guardian$
    $authorizations \leftarrow agent.authorizations$
    $associations \leftarrow student.associations$
    for all $associations$ do
        if $association.guardian = guardian \land unexpired$ then
            $authorizedGuardian \leftarrow true$
            $break$
    if not $authorizedGuardian$ then throw $Error$
    for all $authorizations$ do
        $authRoot \leftarrow authorization.authRoot$
        $action \leftarrow authorization.action$
        if $action = consent \land authRoot = dataUser \land unexpired$ then
            $authorized \leftarrow true$
            $break$
    if $authorized = true$ then
        append $consent$ to $dataUser.consents$
        append $consent$ to $student.consents$
        $promise \leftarrow$ write $consent$ to $consentRegistry$
        $promise \leftarrow$ update $dataUser$ to $dataUserRegistry$
        $promise \leftarrow$ update $student$ to $studentRegistry$
    return $promise$                                                ▷ generated by updates
```

3.2.2 Associations

We outline the Association function in Algorithm 2. The algorithm checks to make sure that the user initiating the function is correctly specified in the transaction. It also guarantees that the associator has an unexpired authorization rooted in the same Data Manager that created the student record. As part of the user interface, the Authorized Agent will be asked to check the guardian ID before continuing to execute the transaction, similar to how they would when handing the parent a form.

3.2.3 Consent

We outline the Consent function below in Algorithm 3, which takes advantage of both the previous functions' outputs.

The Consent function guarantees both that the specified guardian has an unexpired association with the specified record and that the agent carrying out the consent contract is part of the authorization network rooted in the specified Data User.

3.3 Scalability

Our peer network prototype was built in Hyperledger Composer, which uses a local machine with a stubbed test network.

The system can be easily scaled, however. Our implementation assumes that there is only one Data Manager; however, the ACL protocol is easily adapted to multiple Data Managers and chains of authority be restricting Data Manager access to other Data Manager's assets, as Data Managers currently are given total admin access. Other permissions within an individual peer network are already handled.

Hypothetically, the system is infinitely scalable as a blockchain solution, with the addition of nodes serving to increase overall capacity for computation. We have not thought through the roles of various nodes or how those would be executed and hosted as they were beyond the scope of our efforts.

4 Security Evaluation

In this context our primary assets to be protected are the student data records, themselves sensitive data with a low-to-moderate value to an malevolent agent.

4.1 Threat Models

We consider four key threat models as essential to the protection of those data records, and implement key asset design elements to address each.

4.1.1 Nosy Parents

First, we consider the case of a “nosy parent,” or an Associated Entity who considers viewing the grades and status profile of other students. This parent would be motivated through curiosity to examine the records of another student. We evaluate to have a high likelihood, though the parent will likely have little-to-no technical sophistication and limited resources. To mitigate, parents as Associated Entities in our system have READ access only to students who have been associated as theirs by Authorized Agents.

4.1.2 Nosy Volunteers

Second, we examine a “nosy volunteer,” or an Authorized Agent with minimal technical resources, but who has been granted push authority such that they are able to Associate guardians. Authorized Agents with push authority are actually quite powerful: if sufficiently motivated, they could create a parent for themselves, associate the parent with a student, then view the student’s records, in do doing, viewing the records of students when not explicitly given permission to do so by the guardian. We consider this to be moderately likely and mitigate this threat by limiting Associated Agent authority to short periods of time through an authorization expiration date.

4.2.3 Lazy Data Users

Third, we examine the threat of a “lazy data user,” or a third party researcher who wishes to access the student records for either intervention design on behalf of the after school program or general social science research, the results of which can oftentimes have positive social impact.

Even with the best of intentions, data researchers can introduce system insecurities in one of two important ways. While the mitigation of both solutions we consider outside the scope of our work, it is nonetheless instructive consid possible

First, being human, it is always possible that a researcher will access the wrong data file, or will execute a query that was not granted express permission. The database and access to it will need to be designed such that it is robust to these errors, and does not result in leakage of data records for which the parents did not give consent.

The second insecurity is in regards to the researchers’ method of accessing, aggregating, and disseminating sensitive data records. Even though a parent may grant access for a researcher to read a student’s data file in order to design a more effective program, the parent is certainly anticipating that the researcher will preserve the anonymity of the child. It is then imperative that any summary statistics, reports, or data sub-releases do not provide insights that can be easily linked with third party data sets to render identification of a student observable. Differential privacy represents a potential solution to this issue, whereby the preservation of the identity of each student within the data set is preserved up to a given budget level.

4.2.4 Hacktivist

The final threat model we evaluate is that of a “Hacktivist,” or a malevolent and highly technical agent who seeks to obtain access via nefarious means. We consider the likelihood of this attacker low, and mitigate the concern of this threat through the standard method of a single trusted database administrator, here the Data Manager of Chicago Public Schools, who in turn grants access selectively to Data Users via off-chain methods outside the scope of this work.

5 Summary

We’ve outlined briefly a system that can easily be implemented within a Hyperledger Fabric instance using Composer to manage the necessary authentication networks and consents for access to an off-chain resource in a way that is easily scalable and trustless---providing immutable guarantees of accuracy and transparency as a function of the system, not oracles---all with clear process advantages over existing analog systems.

Our codebase is available online and use is governed under an MIT license for easy reproducibility: https://github.com/aldengolab/blockchain-for-data-consent. .

5.1 Future Work

This work is step one in a larger process toward implementing a fully operating system. Scaling the blockchain system to multiple organizations was outside the scope of this stage of the project.

Importantly, the interface will need to abstract away much of the blockchain interactions, so that individuals without technical expertise can easily understand and use the system. To that end, further work in usability testing in field will be necessarily. Additional stress testing security vulnerabilities in field will also be important.

Scaling the Fabric to multiple nodes will be a large undertaking that will require additional thought around the various node roles and how the nodes will be hosted. Our initial suggestion would be to make Data Managers and Data Users each responsible for a node, along with the partners like Thrive who are in the field associating guardians and obtaining consent. There will likely be important considerations around who ought to have transactions stored: while there exist cryptographic guarantees for privacy of the transaction when it is stored, there is always a risk of some data leakage (e.g. via usage, frequent updates, etc) that would worry some policy makers. Additional thought around how to leverage Hyperledger Fabric to prevent these kinds of leakages would be fruitful.

5.2 Acknowledgements

We are greatly indebted to Ian Foster, Matt Gee, Adam Hecktman, Brian Howe, Julia Stoyanovich, Daniel Yao, Soren Spicknall, Dev Bharel, and Kelvin Fitcher for their feedback and advice at various points throughout the development of this project.

5.3 References

1. Office of the Secretary Ethical Principles and Guidelines for the Protection of Human Subjects of Research. The Belmont Report. 1979. Accessed 6/1/17: https://www.hhs.gov/ohrp/regulations-and-policy/belmont-report/index.html.
2. Eckblaw, Ariel, Asaph Azaria, Thiago Viera and Andrew Lippman. "A Case Study for Blockchain in Healthcare: 'MedRec' prototype for electronic health records and medical research data" (2016). Reprinted from 2nd International Conference on Open & Big Data. Accessed 6/1/17: http://dci.mit.edu/assets/papers/eckblaw.pdf
3. Zyskind, Guy, Oz Nathan, and Alex Pentland. "Decentralizing Privacy: Using Blockchain to Protect Personal Data." IEEE Security and Privacy Workshops, 2015.