MicroChain: Transparent Philanthropic Microlending

Aksel Reiten, Allan D'Silva, Francis Chen, & Kristoffer Birkeland

Final Project - 6.857 Network and Computer Security - Spring 2016
Massachusetts Institute of Technology

Abstract. We utilize recent developments in cryptography to design and create a transparent philantrophic microlending platform. We aim for this system to be distributed, verifiable, and transparent, using blockchain technology to meet these specifications. We present our initial prototype, built on the Ethereum platform. In principle, it is usable by anyone with a computer and an Internet connection. We also perform a security analysis of our current system. Finally, we propose further design extensions to improve our application.

1 Introduction and Previous Work

Recently, cryptographic innovations have allowed for independent, distributed computers to collectively execute protocols by consensus. This execution can be both verified and transparent, without centralizing trust at a single point or small group. This paper provides a novel application of such techniques, exploring the creation of a philanthropic microlending system.[1]

1.1 Blockchain Technology

Bitcoin [17, 3] successfully uses distributed consensus to create a digital currency. The validity of the currency is determined by a *public ledger*, of which all nodes maintain a copy. As long as most of the computing power in the Bitcoin network belongs to honest nodes, the network will eventually agree upon a single correct version of history in the ledger [17].

Bitcoin relies on a mechanism called the *blockchain*, which is essentially a log of bitcoin transactions, partitioned into *blocks* [17, 3]. Nodes in the network must solve a *proof-of-work* in order to append to the blockchain and get a reward. The network is incentivized to only accept valid blocks with correct transactions, and each node can store the entire history [17]. Thus, the blockchain creates a single version of history that is both viewable by all nodes and known to be valid with high probability.

The Ethereum platform builds further functionality on the blockchain, introducing a construct called the *smart contract* [10]. This allows transactions in

[1] Our project does not have any dependent stakeholders; thus, it can be safely released immediately.

the Ethereum blockchain to (a) execute code in a Turing-complete scripting language, and (b) persist results in a data store [3]. This theoretically allows for verified, transparent execution of any traditional software program. The Ethereum website touts this as enabling "applications that run exactly as programmed without any possibility of downtime, censorship, fraud or third party interference" [10]. We explore the possibility of creating a philanthropic microlending application with these properties.

1.2 Microlending and Philanthropy

Microlending provides a fascinating domain to apply blockchain technology. In microlending, individual lenders provide small-scale loans to borrowers who cannot access traditional financial institutions [14]. For example, Kiva [15, 20] allows any individual with Internet access to make a loan to an entrepreneur in the developing world. This can provide sustainable pathways out of poverty for borrowers, if they are able to create businesses [14].

Kiva has had reasonable success, moving about $800 million in loans since 2005 [20]. It uses two approaches to provide money: (a) working with on-site Field Partners as an interface with lenders, and (b) electronic methods such as PayPal. Unfortunately, both systems have key vulnerabilities. Though Kiva uses both initial vetting processes and ongoing monitoring to approve Field Partners [15], this is generally a closed process, involving only Kiva and the Field Partner. With such a system of centralized trust, transparency is limited, and fraud and abuse become more likely when larger amounts of money are involved. Electronic payments suffer from a similar problem—since records are generally not public, it is very difficult to know exactly how one's loan is being spent.

The blockchain provides a possible solution to the issues of verifiability and transparency in this domain. In the following subsection, we explore previous work towards this goal.

1.3 Previous Work

We identify two major projects utilizing blockchain technology in the realms of microlending and philanthropy. Both are quite recent and provide an idea of the state-of-the-art.

Factom [13] is using blockchain technology to trace donated funds. Specifically, this project seeks to maintain a reliable record of financial transactions related to The Water Project, a charitable organization in western Kenya.

Rootstock [16] is a technology that seeks to enable microlending on Ethereum. A major goal is to allow banks to use smart contracts to manage loans. Furthermore, Rootstock focuses on integrating Bitcoin into the Ethereum ecosystem, leveraging desirable properties of both [16].

While Factom does leverage blockchain technology for transparency, its usage is limited to specific partner organizations. And though Rootstock is more

generic in nature by focusing on microlending, it is geared towards banking rather than philanthropy. We seek to produce a solution that incorporates elements from both, but solves a different problem.

1.4 Problem Statement and Project

Our goal is to create a philanthropic microlending system that is usable by anyone with Internet access. This system should have the following properties:

- **Distributed.** All transactions should be computed by consensus in a peer-to-peer network, to avoid centralizing trust.
- **Verified.** The system should allow users to require and provide certification regarding the completion of philanthropic projects. We aim to facilitate actual philanthropy rather than just fiscal exchange.
- **Transparent.** Users should be able to view an inalterable history of past actions for any other user, such that fraud and abuse can be easily investigated and eliminated.

Furthermore, our system should provide security guarantees that protect its users from undesired outcomes.

To address these needs, we present MicroChain, a philanthropic microlending application. We expose the overall system design by describing both a prototype application and a set of design extensions. In addition, we analyze and evaluate the system.

Section 2 describes the design and implementation of our prototype. Section 3 discusses the unique benefits of the prototype over existing systems. Section 4 performs a security analysis of the application. Finally, Section 5 presents design extensions.

2 Prototype

In this section, we first specify the context and design of our prototype. We then give key details on our implementation.

2.1 Overview, Features and Assumptions

MicroChain enables users to borrow and lend money to each other, and implements additional features to promote philanthropy and transparency. The protocol is implemented in a smart contract, which describes the system state and all legal interactions. This smart contract has been deployed on the Ethereum blockchain. This enables us to transfer trust and authority from a centralized server to a peer-to-peer (P2P) network. Users can access the smart contract and append transactions to it through a web interface. We make the following simplifying assumptions in our prototype:

- **Users must run a full Ethereum node.** We assume all users have access to a workstation that can synchronize a full Ethereum node. This implies that users have sufficient computational resources, such as storage capacity, processing power, and Internet connectivity, to download the complete history of the Ethereum blockchain.
- **Users must have an Ethereum account.** We assume all users to have sufficient knowledge to use Ethereum. This implies that users can create an account on Ethereum and are able to unlock this account on a local workstation.
- **Users must have initial capital.** We assume all users have access to some initial amount of capital. This is necessary because users need to pay ether, Ethereums's own cryptocurrency, to append transactions to the blockchain.
- **Trusted administrator.** We assume there exists a trusted administrator, that acts as an initial interface between users and the outside world. The administrator handles conversions from real currency to MicroChain tokens, and registers users. Only the administrator and registered users may interact with the system.

2.2 System Architecture

The high-level components of our prototype are (a) a web interface, and (b) a network of Ethereum nodes. The web interface is used to interact with the smart contract. A user enters a public key in the web interface to identify themselves. The web interface communicates with an Ethereum node running on the same machine. This requires the user to have a private key stored in the node, and to initially unlock their account by entering a password at the node.

Instead of using a traditional client-server architecture, with a front-end web interface connected to a back-end server, MicroChain replaces the back-end server with the Ethereum blockchain. Thus, the web interface includes only client-side code, and the "back end" is the smart contract.

2.3 Users

For clarity, we categorize the users of MicroChain into three different groups: borrowers, lenders and administrators.

- **Borrower.** A user who borrows money from another user. In microlending, this would be a person with a need for capital, such as an entrepreneur in a developing country.
- **Lender.** A user who lends money to a borrower. Normally, this would be a person with access to excess capital, motivated by philantropy and/or economic gain via interest.
- **Administrator.** A trusted authority that registers users and transfers money into and out of the system.

Note that in practice, a user can be both a lender and a borrower. All users have a *reputation*, which essentially records their credit history. Users can read from and write to the blockchain with our web interface. We support the following interactions:

All users can:

- Register as a MicroChain user. This requires the user to contact an administrator and is the pre-requisite to every other action.
- Log into and out of the system, given that the user has an active and unlocked Ethereum account.
- Get information from the blockchain. This can be information about (a) all existing loans any user has requested money for, (b) the current user's outstanding and borrowed loans, (c) any user's previous history, including reputation, or (d) a project's history.
- *Certify* other users' projects, which means that the user personally testifies that a project has been completed.

Borrowers can:

- Request money for a project, so that the request becomes publicly viewable and fulfillable by all other MicroChain users. The borrower may optionally specify a target number of specifications for the project, as an incentive for philanthropic lenders.
- Pay back money the user has borrowed from others, given that the user has enough currency on their account. This also assesses whether the user has obtained enough certifications on the project.

Lenders can:

- Fulfill loans other users have requested money for. The lender is not guaranteed to get their money back, and may therefore be incentivized through interest rates that compensate for risk.
- Extend the due date of an outstanding loan. This gives the borrower additional time to pay back the loan without causing harm to the borrower's reputation (note that this feature is implemented in the smart contract, but is currently not accessible on the front end).
- Default an outstanding loan that is past due. This permanently records the borrower's failure to repay, and decreases their reputation. This also assesses whether the borrower has obtained enough certifications on the project.

Figure 1 illustrates the interactions within the system.

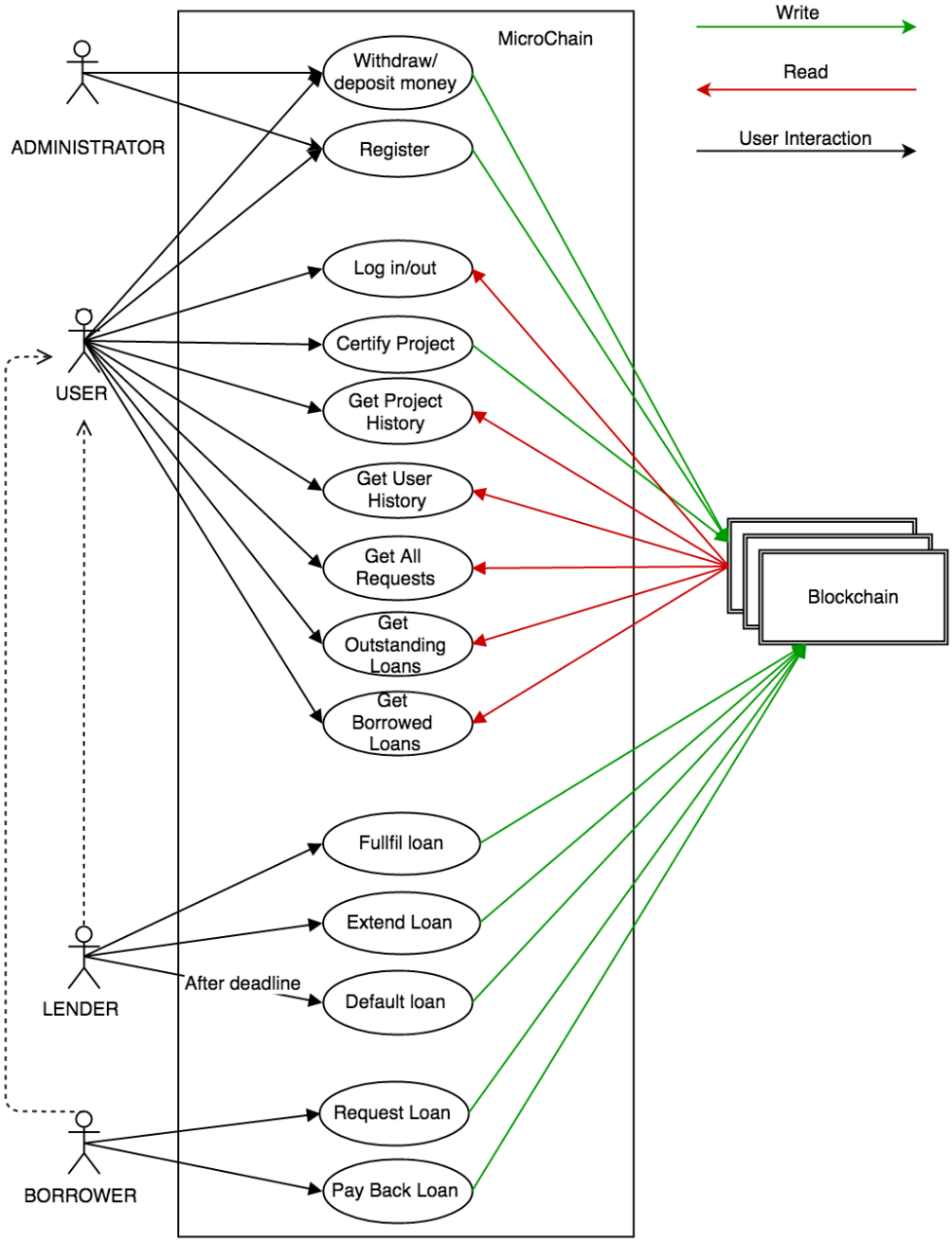

Fig. 1. MicroChain use case diagram. Dotted arrows signify that all users can be both borrowers and lenders.

2.4 Implementation

The MicroChain smart contract is written in the Solidity programming language, which is an Ethereum-specific language [8]. The smart contract defines the data and methods involved in MicroChain. Our primary challenge with Solidity was the difficulty of debugging a blockchain application—each code change required a new contract deployment (incurring latency), and errors were unexplained and difficult to diagnose.

The smart contract is currently deployed on the Morden testnet [7]. We make this choice in order to demonstrate the system without expending the resources required for a real Ethereum node.

We used Ethereum Wallet, a graphical user interface (GUI) [10] and Geth, a command line interface (CLI) [6] to download the blockchain and manage accounts. We conducted contract deployment and basic debugging through Ethereum Wallet.

We implemented a web interface using HTML, CSS, and JavaScript, to create a usable front end for MicroChain. In order to communicate with the blockchain, we used a newly developed JavaScript library called Web3 [9]. We note that Web3 is still in version 0, and that implementation in such a new environment was sometimes an uncertain and ad-hoc process.

2.5 Repository and Demos

Our source code is hosted on GitHub: *https://github.com/akselreiten/857_root*

The repository also contains a demo video, as well as a set of instructions for getting MicroChain running on your own machine.

3 Advantages of MicroChain

The main advantage of MicroChain is that it provides an unalterable history of previous transactions. This includes users' lending and credit history, and the history of all project. We believe that this will make our system attractive for both lenders and borrowers.

In addition to this general feature, MicroChain provides three specific categories of advantages over existing solutions.

3.1 Low Resource Demand

In order to enable microlending, we can simply ask an on-the-ground aid organization to run an Internet-connected Ethereum node. Locals can visit the organization to use MicroChain to borrow money. Lenders in the developed world can run their own nodes. Though the system requires non-trivial technical expertise to set up, it requires only a limited amount of additional infrastructure, while eliminating the need for manual loan management.

3.2 Unforgeable Credit Rating

Credit ratings are essential to microlending, providing lenders with a signal for borrower trustworthiness. By providing a transparent, unalterable history, MicroChain enables credit ratings that cannot be hidden or forged. This can facilitate establishment of reasonable interest rates across all loans. In particular, honest borrowers can be consistently rewarded with lower interest rates. This will incentivize repayment of loans.

Currently, our credit rating is a summary of a user's past borrowing activity. This includes a *cash reputation* (total value of repaid loans minus total value of defaulted loans), the number of defaulted and repaid loans, and the number and value of currently outstanding loans. We believe that this provides the most important information that lenders need.

3.3 Certifications

Even if a borrower has a solid credit rating, they may divert borrowed money from its stated purpose. It is therefore key that the lender can verify how their loan was used.

Our certification system solves this problem by allowing borrowers to set an amount of certifications needed for the project. For example, a borrower may request funds for a water pump, promising to have at least 10 users testify that the pump was built. If this goal is not achieved, the reputation of a borrower will be permanently damaged.

Though users may collude to commit fraud, MicroChain's inalterable history disincentivizes this behavior. This is because all conspirators would be immediately implicated if the fraud were ever discovered.

In addition, aid organizations can register as users and certify projects. This could potentially increase the credibility of projects and borrowers.

4 Security Analysis

Our prototype relies on both cryptographic and real-world security practices. In this section we define our notion of security, explore some of these practices, and evaluate the system against potential attacks.

4.1 Security Policy

Here, we define the ideal security policy of MicroChain. There are three roles in our system, each with a unique set of permissible actions.

Administrator

- Should be able to register users, defining who can and cannot use the system.
- Should be able to register self as a user.
- Should not be able to register any user more than once.

User

- Should be able to convert real currency to MicroChain currency.
- Should be able to both read from and write to the blockchain, as defined in the user model in Section 2.
- Should not be able to otherwise alter MicroChain data.
- Should not be able to have more than one account.
- Should not be able to fabricate real currency or MicroChain currency.

Non-User

- Should not be able to alter MicroChain data in any way.

To summarize, MicroChain should provide *both authentication and integrity* for registered users, but *neither anonymity nor confidentiality*. This is because the reputation and history system relies on a person being permanently associated with one account only, and on all history being public.

In addition, MicroChain should incentivize good loan behavior, and provide ways for users to reduce the risk of loan abuse. Finally, the system should be highly available.

4.2 Attacks and Vulnerabilities

Here, we explore various breaches of our security policy, focusing on our prototype. Section 5 contains potential remedies for some of these.

Computer Access It is not reasonable to assume that all developing world users have the resources to run their own Ethereum nodes. Therefore, the likely deployment setting of the prototype would be to have an aid organization provide a computer that runs an Ethereum node and allows use of the system. Unfortunately, when an Ethereum account is unlocked on a computer, this allows any user of the computer to access the account and use it for MicroChain.

In order to prevent users from posing as other users, the owner of the node would have to carefully restrict physical access to the computer. In addition, users would have to always end sessions properly, which is not entirely intuitive [18]. This process would be error-prone, especially if users are not technically trained, leading to potential security breaches.

We discuss addressing this problem in Section 5, by enabling simple payment verification (SPV) [17].

Trusted Authority In the prototype, we rely on a trusted authority to register users and handle currency transfer into and out of MicroChain. Importantly, this allows a malicious administrator to fabricate as much MicroChain currency as they want—there is no real-world grounding at all.

In addition, there is currently no verification mechanism that guarantees that each user only ever has one account. This leads to several additional attacks, described below.

We propose to address these problems using sidechaining for currency exchange [1], and by using a "chain of trust" for verification, as described in Section 5.

Multiple Borrowers If users can create arbitrarily many acccounts, a malicious user can borrow and default, then create a new account with a fresh reputation, repeatedly. We characterize this as a "multiple borrower attack." User verification is essential to assure lenders that their loans are going to uniquely identified people, thus preventing a market failure.

In order to fix this problem at scale, we need the means to authenticate multiple administrators (one for each locale), and to verify individual users, in a way that prohibits this attack. We discuss solutions for this in Section 5.

Smart Contract Flaws and Attacks A subtle issue with our prototype is the safety of the implementation. Although Ethereum provides a cryptographically secure blockchain, there are unique challenges intrinsically present in the implementation of a smart contract. Previous research has shown that even creating a simple smart contract is quite demanding [5]. Specifically, when university students were tasked with implementing a smart contract, some common stumbling blocks included (a) poor conceptual design at the state-machine level, (b) misunderstanding and mishandling of stakeholder incentives, and (c) insufficient understanding of Ethereum [5]. At least some of these issues may also apply to our system. Additionally, we consider that Solidity is extremely new (version 0.2.0) [8]. Thus, it likely to be under rapid development and not completely tested. Thus, we cannot provide a reliable guarantee of correctness in our implementation. We address this issue in Section 5.

Borrow-and-Run Another vulnerability of our system is that lenders have few ways to force the borrower to pay back their loans. Instead, we rely on a borrower's desire to build a good reputation. This is the purpose of the credit ratings, as discussed in Section 3. The assumption is reasonable if the system gains a substantial market share, but is less credible otherwise. In particular, a borrower could create a single account, request and obtain a substantial loan, and quit the system, making a one-time profit.

Our credit system attempts to address this by listing the number of loans a borrower is currently holding, to limit the ability of a new borrower to obtain many loans without establishing trust. Also, to spread the risk for lenders, we consider the option of having multiple lenders to fund a single borrower in Section 5.

5 Design Extensions

In this section, we propose design extensions that aim to address key vulnerabilities and add important features in MicroChain. We conclude the section by identifying potential areas of future work, where unanswered questions remain.

5.1 Features

We list a few high-level features would be relatively easy to implement, and would increase the usability of the system.

- **Multiple Lenders and Borrowers**. Larger-scale philanthropic projects might require many-to-many relationships between lenders and borrowers.
- **Additional Documentation**. It may be helpful to allow additional documentation to be persisted on the blockchain, e.g. government-issued IDs of users, photo and textual documentation for projects, etc. This would help to provide additional details in our inalterable history.
- **Required Certifiers**. In the case that there is a trusted aid organization on-the-ground, lenders may wish to require this organization to certify a project.

5.2 Dedicated Blockchain

It would be feasible to re-implement MicroChain in a dedicated blockchain, based on Bitcoin's design [17]. As a first pass, we could use the same P2P protocol and proof-of-work. To replace the data store offered by Ethereum [10], we would use a *statefile*, which is a feature offered by Bitcoin [3]. In the case of Bitcoin, this is essentially a table of unspent coins used to verify transactions. It is populated by scanning the downloaded blockchain [3]. The analog in MicroChain would keep track of all system state, including loan requests, outstanding loans, projects, reputations, and user histories, providing the same functionality as our prototype. Actions that mutate state would simply add transactions to the blockchain. This could have some distinct advantages:

- Limited functionality could be easier to implement than smart contracts.
- We could focus on adding features to increase usability for target audiences.
- Full control over core implementation to fix vulnerabilities and apply upgrades.

However, the obvious disadvantage is that this altcoin would be difficult to secure, if the mining capacity is low [3].

In the following sections, we discuss both potential features and advantages enabled by a dedicated blockchain, and ways to secure such a system.

Key Management and Simple Payment Verification. Currently, Ethereum [10] seems to store private keys in encrypted form on a user's computer. The user must enter a password to unlock them. This approach can be inconvenient—the user needs a separate account for each device. In addition, they must download the entire blockchain on every device.

It might be more convenient for a user to use a *password-derived wallet* [3], e.g. in a web or mobile application. This would allow access to MicroChain on any device. This method allows a user to generate private keys deterministically upon sign-in, using a password. However, this is vulnerable to any attack that targets passwords, e.g. rainbow tables.

Another solution would be to enable some form of *simplified payment verification* (SPV) [17, 3]. SPV, first proposed in the Bitcoin protocol, is a way for a client to verify blockchain information with high certainty, without storing all of the blocks locally and without trusting any node. Though Ethereum is in the process of developing such a protocol [11], our own blockchain would give us control over the implementation of SPV. This would allow us prioritize this feature, and also to add additional verification mechanisms. For example, SPV clients could be made to resist attacks by downloading full blocks upon receiving invalid headers, as mentioned in [17].

Given SPV and local encrypted key storage, we would envision deploying MicroChain by (a) having on-site aid organizations run full nodes near user populations, and (b) having users access MicroChain using a smartphone application that stores encrypted keys locally, runs SPV, and can post transactions via full nodes. This would allow for a reasonable balance of security and convenience.

Sidechaining. At present, we do not have a solution for verifiably transferring currency into and out of our system, instead relying on a trusted authority. Sidechaining [1] is a proposal that would allow transfers of currency between Bitcoin and an altcoin. This could theoretically allow users to use Bitcoin for MicroChain transactions. Notably, Rootstock [16] proposes to use sidechains as well.

Though sidechaining is currently unimplemented on Bitcoin [3], it could become a feature in the future. Creating our own altcoin might maximize our ease of adoption. Bitcoin is an excellent means of exchange for our application, because it operates without regard to international borders and banking infrastructures, assuming Internet access [4].

Incentivizing Mining. The security of an altcoin depends on its mining capacity—it must be prohibitively difficult for more than 50% of the hashing power to be malicious [17]. Thus, we propose ways to increase the number of miners on the network.

Using a technique called *merged mining*, we can allow Bitcoin miners to also provide proofs-of-work for MicroChain [1]. This technique is currently used in practice, and is one of the pillars of Rootstock's approach [16]. However, as noted in the Rootstock article, miners must opt into merged mining.

To promote miner adoption, we propose to use transaction fees. These can be extracted from both loaned currency and from promises of future debt payment. We would seek to set transaction fees at a level that creates an adequate incentive, but incurs substantially less overhead than a physical lending organization on-the-ground. Given the potentially high interest rates in microlending [14], promises of future repayment could be an interesting form of compensation, playing the role of an investment with high risk and high return.

5.3 Identification and Authentication

The security of MicroChain depends on the guarantee that each user can only create one account. We propose to enforce this by creating a "chain of trust", as follows:

1. Aid organizations on the ground run MicroChain nodes and have the power to register users. Their identities are verified by a CA like Symantec [19].
2. Aid organizations register users using some combination of biometric security and government-issued identification. Only registered users may interact with MicroChain. By only allowing user registrations that come from verified sources, and requiring organizations to do verification, we form the "chain of trust."
3. For further verification, identity information may be stored in inalterable history on the chain, for future auditing purposes.
4. In the case of fraud or abuse, depending the context, aid organizations may work with local or international law enforcement in pursuit of justice. At the very worst, all users have an inalterable record of fraud, including all implicated parties, making it difficult for fraudulent borrowers to continue such practices.

5.4 Future Work

In both our prototype and extended designs, we assume the presence of aid organizations on the ground to perform verification. This is a lighter burden of infrastructure and trust compared to fully administering loans. However, we observe that this makes the system *decentralized* rather than *distributed* (see Figure 2). This means that there is still a (reduced) degree of centralized trust, and availability can be compromised [2]. We would like to explore ways to move towards a more distributed design, without requiring users to all run full network nodes.

Additionally, block times tend to create latency, leading to a sub-optimal user experience. We would seek to pursue methods for eliminating this latency as much as possible. It seems that 12 second block times are possible in Ethereum [12], meaning that we may be able to achieve a desirable number of confirmations in minutes. If this remains sub-optimal, we can explore UI-level approaches for reducing the impact of latency.

Finally, as the most important area of future work, we would seek to complete a full conceptual evaluation of Ethereum against the approach of using our own blockchain. This would allow us to determine which method to pursue, to most effectively fulfill our goal of producing a distributed, verified, and transparent philanthropic microlending system.

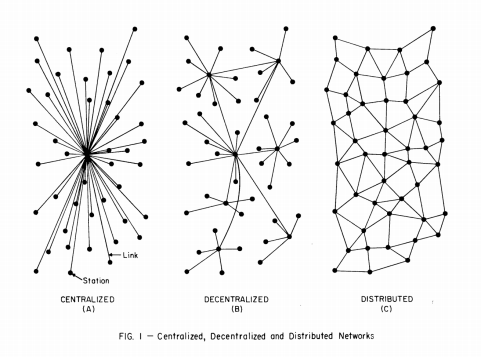

Fig. 2. Centralized, decentralized, and distributed networks. If we assume that all users run their own nodes, MicroChain is distributed. However, our final design uses a pattern more like a decentralized network, with aid organizations as the "hub" nodes. This means that trust is somewhat centralized, and availability is dependent on the few "hub" nodes, rather than a single central node. From [2].

6 Conclusion

We have designed and implemented a prototype of a philanthropic microlending platform that is both distributed and transparent. By storing all transactions on a public blockchain, it can connect borrowers and lenders from opposite sides of the world, producing an inalterable history of transactions. Anyone with a connection to the Internet can use the system, after being registered by an administrator.

We have also discussed potential attacks on our current system, and proposed designs to address some of these. In conclusion, we believe that this demonstrates the applicability of blockchain technology to philanthropic microlending.

Acknowledgements

Thanks to Professor Ron Rivest and the 6.857 TAs, especially Conner Fromknecht and Cheng Chen, for instruction and guidance. This project was enabled by our learnings from the class, and the course materials.

References

1. Back, A., Corallo, M., Dashjr, L., Friedenbach, M., Maxwell, G., A, M., Poelstra, A., J, T., and Wuille, P. Enabling blockchain innovations with pegged sidechains. *https://blockstream.com/* (10/22/2014).
2. Baran, P. On distributed communications: Introduction to distributed communications networks. *United States Air Force Project Rand* (August 1964).
3. Bonneau, J., Miller, A., Clark, J., Narayanan, A., Kroll, J. A., and Felten, E. W. Research perspectives and challenges for bitcoin and cryptocurrencies. *IEEE Symposium on Security and Privacy* (2015), 104–121.
4. CAF. Giving a bit(coin): Cryptocurrency and philanthropy. *Charities Aid Foundation - Giving Thought* (5/2015).
5. Delmolino, K., Arnett, M., Kosba, A., Miller, A., and Shi, E. Step by step towards creating a safe smart contract: Lessons and insights from a cryptocurrency lab. *International Association for Cryptologic Research* (11/18/2015).
6. Ethereum. Ethereum wiki - geth. *https://github.com/ethereum/go-ethereum/wiki/Geth* (Accessed 5/11/2016).
7. Ethereum. Ethereum wiki - morden. *https://github.com/ethereum/wiki/wiki/Morden* (Accessed 5/11/2016).
8. Ethereum. Solidity documentation. *https://solidity.readthedocs.io/* (Accessed 5/11/2016).
9. Ethereum. Web3 - ethereum compatible javascript api. *https://github.com/ethereum/web3.js* (Accessed 5/11/2016).
10. Ethereum. Official website. *https://www.ethereum.org/* (Accessed 5/8/2016).
11. Ethereum. Light client protocol. *https://github.com/ethereum/wiki/wiki/Light-client-protocol* (Accessed 5/9/2016).
12. Ethereum. Blog: Toward a 12-second block time. *https://blog.ethereum.org/2014/07/11/toward-a-12-second-block-time/* (Accessed 7/11/2014).
13. Factom. Transparency tech seeks to solve non-profit accountability. *https://www.factom.com/transparency-tech-seeks-to-solve-non-profit/* (9/1/2015).
14. Hayes, A. Investopedia: What is microlending and how does it work? *http://www.investopedia.com/articles/personal-finance/040715/what-microlending-and-how-does-it-work.asp* (Accessed 5/8/2016).
15. Kiva. Official website. *https://www.kiva.org/* (Accessed 5/8/2016).
16. Maras, E. Rootstock merges mitcoin and ethereum to help world bank with micro-lending. *https://www.cryptocoinsnews.com/rootstock-merges-bitcoin-ethereum-help-world-bank-micro-lending/* (11/16/2015).
17. Nakamoto, S. Bitcoin: A peer-to-peer electronic cash system. *www.bitcoin.org* (2008).
18. Reddit. Reddit post on re-locking accounts in geth. *https://www.reddit.com/r/ethereum/comments/3gngw4/do_i_need_t_reloc_my_account_after_sending_a/* (Accessed 5/11/2016).
19. Wikipedia. Symantec. *https://en.wikipedia.org/wiki/Symantec* (Accessed 5/11/2016).
20. Wikipedia. Kiva (organization). *https://en.wikipedia.org/wiki/Kiva_(organization)* (Accessed 5/8/2016).