Blockchain for the Common Good: A Digital Currency for Citizen Philanthropy and Social Entrepreneurship

Rahul Simha
Department of Computer Science
George Washington University
simha@gwu.edu

Shweta Jain
Department of Mathematics & Computer Science
John Jay College of Criminal Justice
sjain@jjay.cuny.edu

March 22, 2018

Abstract

Popular cryptocurrencies and their attendant ledger technologies have generated considerable interest because of properties that empower *individuals* such as secrecy, anonymity, disintermediation, the lack of central authority, and, in the case of smart contracts, the ability to efficiently specify transactional conditions. At the same time, there is growing skepticism about whether private companies will willingly give up their asymmetric informational advantage, whether individuals can rely on cryptocurrencies that are not backed by governments, or whether such systems reduce trust in institutions.

In contrast, we present an application of distributed ledgers for the world of citizen philanthropy and social entrepreneurship, with stakeholder incentives designed to increase *social good* through accountability, transparency, and flexibility. In this system, which we call Directed Cash, individual donors specify conditions (the "Directed" part) attached to their donation or investment ("Cash"), that are then paired with interested recipients or aggregators of recipients (charities, social entrepreneurs) using distributed consensus so that the intent and pairing are open while maintaining donor anonymity. Furthermore, Directed Cash flows both ways to promote accountability and transparency: after receipt, a validation flows backwards to return to the donor so that the donor receives a report of how their donation was spent.

While some elements of the system borrow from existing cryptocurrency and blockchain technologies, we propose alternative incentives for distributed consensus that are better aligned with the application and promote social good through the stakeholders. This paper describes the goals and concepts, and a system design to achieve these goals, a primary feature of which is a simple SQL-like language to enable specifying conditions, pairing, aggregation and publicly-verifiable reporting.

1 Introduction

Citizen donors or investors concerned about the impact of their contributions face a daunting gamut of choices. For charitable donations, for example, they must either trust a well-established charity by contributing to its general fund or spend time scouring crowd-funding sites for particular projects of interest. Not only is this donor-to-cause *pairing* problem time-consuming for donors[1], it forces charities to spend precious resources competing for eyeballs. But most importantly, there are few opportunities for ordinary donors to specify in much detail how they want their contributions spent, and nearly impossible to verify that their donations were spent as intended. These shortcomings raise a useful question: can a marketplace of philanthropic transations be infused with the right mix of technologies to improve efficiency and create a virtuous cycle that steers stakeholders towards the greater common good?

[1]We will use *donor* interchangeably with *investor* because the underlying system goals are similar: both want to specify conditions attached to their contribution, seek projects that meet their interests, and desire accountability

In this paper, we argue that the following combination of technologies is well-suited to addressing the above question. First, a digital currency (called Directed Cash here) that allows donors to attach conditions to a donation such as the type and location of aid, a cap on organizational overhead, and other such expressions of donor intent. Second, an algorithm (algorithmic pairing) to connect donations to charities and causes. Third, a distributed ledger so that the system functions without a trusted third party. Fourth, a linked structure within the distributed ledger that allows the backward flow of transactions, connecting expenditures to the original donors so that they receive confirmation that their stated intent was satisfied. And finally, a high-level query language called DCL (Directed Cash Language) to facilitate clear expression of donor intent and recipient action. We explain why these components help us design a system that will be effective in reaching the overall goal of directing donations to causes that matter to donors.

1. Directed Cash (DC): The new digital currency will be linked to government backed currency through a central bank. Unlike Bitcoin, our goal is not to gain independence from the government and we do not want the currency itself to become an investment vehicle. Instead DC is merely a transactional convenience that allows a donor to *direct* spending to favorite causes and with conditions important to them. In addition, we build into the currency a backward flow of accounting so that every dollar spent automatically carries with it a token that flows back to the original donation.

2. Algorithmic Pairing: With algorithmic pairing, donations are not pledged to an organization but to a cause. Using a matching algorithm, a program can match donors' will with the needs of potential recipients of the donation. Analogous to program trading in stock markets, algorithmic pairing has the potential to mitigate human bias and result in socially more optimal allocation of resources. For example, algorithmic pairing of donors with causes helps address the problem of *irrational herding* [AM16], in which popular projects tend to attract new investors even if other deserving projects are better suited to investor intent. Both the popularity score and the webpage placement within crowdfunding websites can adversely influence investor choice; this is what an algorithmic approach helps mitigate. Just as importantly, our system organically establishes an open verifiable system which enables donors motivated by receiving public recognition [Ves06] to tout their contribution but also to stimulate others into providing "matching contributions."

3. Distributed Ledger: Anything without an incentive is bound to fail when the parties involved pull out their support due to adverse political, economic and other circumstances. A consensus based distributed and decentralized system can be run by stakeholders who stand to gain and thus have a vested interest in keeping the system honest and operational. Therefore, a distributed ledger based system without a trusted third party appears to be a promising approach. A blockchain based consensus has a low barrier to entry (and exit): anyone with the software can join or leave as they please. As long as there is sufficient interest in the application, we presume there will be sufficient honest nodes to keep the application functioning with integrity. However, a proof-of-work or similar blockchain based system are unnecessary in the target application, even burdensome for the types of stakeholders in our realm of interest. Instead a general consensus scheme is more suitable to address the somewhat more benign threat model, as explained later.

4. Directed Cash Language: Since a working system needs a compact and precise way to specify donations as well as causes, we design a high-level SQL-like language called DCL (Directed Cash Language). This language will facilitate pairing, reporting or accounting, and verification.

Our proposed system can be viewed as a practical and secure simplification of more general enterprises in distributed autonomous governance [DuP17]. Such futuristic systems envision an economy of interacting software programs that represent end users where the rules of governance are voted on electronically and autonomously by the participating programs. DuPont describes a failed attempt at fully decentralized so-called algorithmic governance in which Turing-complete programs representing users engage in transactions accomplished via a distributed ledger [DuP17]. The experiment, which initially attracted significant funding, was eventually shut down (manually) because of several potential attacks. Our goal is to avoid such issues entirely with three features. First, actual transactions involving money in our system use the standard banking system; the Directed cash is simply a token that attaches a dollar amount to be drawn from an actual bank in accordance with a donor's intent. When a donation is spent by a recipient, a donor's reference to the funds is fulfilled by a central bank that holds the money in reserve. Second, the programming language is not only not Turing-complete, it has only the specific interactions needed for its goals; this prevents the system from being attacked via the arbitrary complexity of general-purpose code. Lastly, the only algorithmic component is pairing. Thus, the only potential security loopholes are of two kinds: the standard ones with any existing digital payment system, and any "gaming" of the incentives for the stakeholders. We address the latter in a forthcoming section.

We begin by describing each of the system stakeholders or roles, and their needs. Then, in Section 3 we outline the system, followed by an explanation of the DCL language in Section 3.3. We present our threat model in Section 4 and related work is discussed in Section 5 before concluding in Section 6.

2 Stakeholders and their Needs

We begin by identifying the stakeholders:

- *Donors* are typically individuals or foundations who invest in a project, a cause, or an organization that supports their cause.
- A *recipient* is the ultimate target of a donation, typically an individual. A *project* is a fixed-duration enterprise with a specified outcome; thus, a project can be a collection of recipients along with spending goals, or a collection of physical items (such as habitat materials).
- A *vendor* is a commercial entity that provides goods and services to projects or recipients; examples include grocery stores, landlords, social workers. Most importantly, a vendor's legally-bound receipts form the basis of validating expenditures.
- An *aggregator* brings together donors, projects, or recipients under one theme or organization. For example, a *donor aggregator* can offer to match contributions from other donors, whereas a *recipient aggregator* more closely resembles a charity that features projects and member recipients. Many foundations are often a mix of recipient and donor aggregator. In our system, such organizations would have to separately list themselves in both categories if they wish to offer both functionalities.
- A *regulator* represents a third party solely interested in ensuring the system is functioning as designed; most commonly, a regulator is likely to be a government oversight agency or a nonprofit performing a public service.

2.1 What donors seek

Economic analyses of donor motivations demonstrate a variety of intent [Ves06] much of it deeply personal [RB16], arguing for a flexible system that accommodates all types of donor motivation. What is equally clear is the desire among donors to monitor and evaluate outcomes from their contributions [A.B15, Cau13, McG97], to find the right opportunity, and to use public announcements for their own satisfaction or stimulate others [Ves06]. Thus, for our purposes, we seek a design that enables the following:

- Donors should be able to easily attach directions with their contributions, such as "I want my donations to go only to food costs, through charities that have lower than 10% overhead and who operate in my state."
- Donors should receive as complete an accounting as possible of how their money was spent, with some type of proof that it in fact met the conditions attached to their donations.
- Donors should not have to spend much time hunting for projects that meet their interest, nor to check how their contributions get spent even if they have full accounting. Thus, the system needs to have public accountability where third-party public-interest groups can verify any path from donor to final expenditure.
- Beyond the transfer of funds at the time of purchasing Directed Cash, donors should have the option of maintaining anonymity. Yet, donors who wish public recognition or to stimulate others via matching contributions should be able to do so easily.
- For donors who prefer to browse, the system should enable search queries, the results of which can be used by donor websites for convenient display during browsing.
- Donors should be able to rate their experience for the benefit of all participants.
- Donors should be able to, where appropriate, properly document tax-deductible contributions.

2.2 What aggregators seek

Aggregator features center around automation and efficiency:

- The system should facilitate both donors that wish to announce a matching-contribution scheme, and donors who seek to leverage their contributions through matching.
- It should be easy for recipient aggregators to define projects and their attributes in ways that promote optimal pairing or large-scale aggregation.
- The system should enable effective stakeholders to stand out. Thus, attributes such as low-overhead or low-cost should be easily available as conditions.
- Search queries should include ways of identifying current projects to allow aggregators to avoid unnecessary overlap with other aggregators.

2.3 Features for vendors

We assume that vendors are commercial entities ultimately interested in profit. However, we exploit the fact that indirect outcomes that lead to profit, such as advertising and reputation, are just as desirable. One of the most important system features is a reputational incentive for vendors to report on spending from recipients. This creates a virtuous cycle whereby responsive vendors are rewarded by having donated funds reach their products. Also, through the systems public verifiability, a vendor can acquire positive reputation that impacts its other customers, in much the same way that commerical entities that support social causes enhance their standing in society.

2.4 Features for recipients

Beyond actual resources, a recipient can benefit from efficiency. For example, purchases made by recipients directly from vendors through our system should minimize the reporting burden for recipients. A smartphone app achieves both point-of-sale and also lets recipients understand their past spending, their future allowance and the range of vendor options.

2.5 Other desirable system properties

The overriding goal of our proposed system is to incentivize all stakeholders to fulfill their role in making the system work, and in doing so, achieve better outcomes for all. Thus, we need mechanisms that align the common good with each stakeholder's goals to advance their own interests. Since aggregators and vendors seek recognition, we propose a points-based reward system to enable these stakeholders to rise in rankings by gaining points for participation in distributed consensus. This, together with a mechanism to prevent domination by any particular stakeholder, helps us avoid the burden of proof-of-work blockchains. Finally, stakeholders should be able to make changes to the system through the consensus approach. This is particularly relevant for sanctifying nomenclature (what constitutes food, for example) and rankings (of vendors, for example).

3 How Directed Cash Works

3.1 Components in the system

We assume that donor aggregators or crowd-funding platforms will offer donors a user-friendly interface and will connect through the banking system to enable donors to both make contributions and specify their conditions in a convenient manner. Aggregators, vendors, regulators, and any member of the public may operate a *Directed Cash* server (DC-server) that participates in the computations of the system. The computational goals of the system (algorithmic pairing, validation, reporting, recognition) are achieved through the distributed computation that occurs amongst these servers. In particular, servers jointly agree through distributed consensus on the public ledger that includes anonymized donations, their pairings with recipients or projects, vendor validation, rankings, and reports sent back to donors.

In the following subsections we describe various aspects of the overall system:

- *The attribute-hierarchy specification.* Because donors' conditions can greatly vary, there needs to be a simple way to let donors specify their conditions. One approach would be to allow arbitrary text but that creates two problems: (1) donors do not necessarily know ahead of time what their options are ("I didn't know I could limit this to food"); and (2) pairing can be error-prone as we try to align one text description with another. This type of problem has been addressed to some extent knowledge-representation systems using *ontologies* [WPH06], which are predefined categories and hierarchies of accepted terms. We merely enable the use of any tree-structured ontology (such as those described in [Bri, Hep08]), along with a distributed mechanism to alter the ontology as needed.
- *The language syntax and expression of queries.* Rather than describe the language formally, we instead show by example a few typical queries written for each of the stakeholders.
- *Distributed consensus and ledger maintenance.* The core of the system is the maintenance of the public ledger so that the servers agree on the resolved parts (the blocks in the chain), and so that anyone can examine the ledger.
- *Algorithmic pairing.* Our goal is to ensure that pairing occurs efficiently and fairly so that donations are distributed evenly amongst candidate recipients or projects that satisfy the conditions attached.
- *Validation and reporting.* Finally, we describe how vendor receipts are used as the starting point for the flow of reports back to donors.

3.2 Why another smart contract language?

The notion that digital transactions should be "programmable" is not new, arising out a desire to go beyond pre-formatted message exchanges as in the Electronic Data Interchange standards [KB96]. The central idea is to accompany transactions with executable code that provides dynamicity and agency to both buyer and seller, so that richly complex transactions can meet desired terms and conditions automatically. Seijas et al [STM16] survey existing and proposed *smart contract* languages, whereas Bordini et al [BBD$^+$06] survey languages for multi-agent systems (one application of which is financial transacting).

While a full-fledged programming language allows maximum expressibility (Turing-completeness), automated verification is theoretically impossible, practically infeasible as emphasized in [STM16], and vulnerable [DAK$^+$16]. Also, if a conflict should arise, it is humanly difficult to examine two programs to determine if their contract should succeed. And high expressibility also leads to complexity, and gaming the system through complexity, analogous to fine print in legalese. The other fully-flexible and human-understandable alternative, a text-based approach, in which stakeholders merely write their terms in plain text is also problematic because plain text specifications naturally invite conflict based on ambiguity. At the same time, a fixed-format predetermined list of allowable donation categories cannot anticipate future needs, and restricts the creativity of projects and aggregators.

We propose a compromise by using a carefully constrained non-Turing complete language along the lines of SQL. For databases, SQL combines enough expressibility for simple querying with simplicity in syntax, and most importantly, can be fully validated (and also optimized for execution speed) at runtime. SQL also features the minimum needed for updates, reports, and aggregation. Thus, our goal is to strike a reasonable balance between expressibility and verifiability.

3.3 The Directed Cash Language (DCL)

Rather than present a formal description of the language that might be more suited to a technical report, we outline the language's features through examples typical for the stakeholders.

3.3.1 Identity and authentication

Because every DCL transaction (query and statement) will feature the writer's identity, and will itself have a unique ID, we first address the question of how stakeholders are identified and authenticated. The identity and authentication part of the system is not central to the contributions of the paper and thus we see any of several approaches as roughly equal, with competing advantages.

The first approach is to have the system require a single central bank or a digital payment system such as PayPal which is responsible for issuing both types of IDs: stakeholder identities as well as unique transaction (or DCL statement) serial numbers. Clearly, since our goal is to have our system tied to standard government-backed banking, it is convenient to have all stakeholders possess accounts linked to the digital payment system which generates IDs and transaction numbers. The advantages are simplicity and convenience in transaction ID generation, but its disadvantages include a single point of failure and reliance on a single source for trust. Note that computational efficiency is less of a concern because most DCL statements and queries feature a human-in-the-loop that automatically limits the rate at which transactions are likely to be generated. Thus, the maximum needed computational capacity needed is no worse than at any major ecommerce website.

The second approach is a slight variation: each stakeholder is affiliated to some bank or credit card account, and these are at mutually-distrusting banks. These accounts are responsible for issuing IDs. This has the advantages, again, of tying the proposed system to regular banking but, through multiple banks that audit each other, can address both the single point of failure and single source of trust.

In both the first and second approaches, we assume that banks will post to the distributed ledger every request for an ID. This is a critical part of the infrastructure since authenticated IDs are an essential part of the security in the system. Also, for donors that do not wish to be identified, the IDs anonymize the donor but can be "opened" via public-key cryptography in case of a legal challenge.

The third approach is to use a distributed identity system similar to that of Bitcoin and others. Since this is well studied [FM13] we do not discuss this further. We note that one compelling advantage of using standard banks is that they combine four useful features for the stakeholders: (1) the banks are already legally bound to operate under audit; (2) they have an incentive to be efficient since they benefit from the transaction flow; (3) the banks can serve as a policing function by blocking malicious parties from gaming the system; (4) and perhaps the most important one is that banks have the legal authority to authenticate stakeholders as individuals and as organizations based on their physical tax identifiers. This last one is critical in preventing denial-of-service or flooding attacks that create multiple fake identities.

3.3.2 Queries for donors and donor aggregators

We envision the following common scenarios for donors. Donors should have the ability to: (1) search existing proposals, projects and recipients; (2) make isolated donations with constraints and reporting requirements; (3) issue continued donations spread over time; (4) stimulate or join matched contributions; (5) rate a recipient aggregator; (6) receive reports on their spending; (7) perform validation by operating a server in the ecosystem.

Let us walk through the steps with a simple example of a donor that wishes to make a monthly donation for a year for either food or clothing. The donor first uses their account with the central digital bank to convert actual currency into a token that can be sent with the DCL statement into the public ledger. The DCL statement includes the user's ID, a transaction ID, and a signed digest of the entire statement:

```
FROM ANON (USERID=VO8t05RV9NrZYtrT1wL4QLcjp)
DCL ID y70koDcJx4lUKk6tAgvwSRGr0uIm
TIMESTAMP 2018-02-12T20:53:00+00:00:00001
TOKEN 1fAPdDCxJmqGHZnPhtRqC96IXW1wiDE
DONATE $100 MONTHLY 3-10-17 TO 3-10-18
DECIDE FCFS
WHERE (SCHEMA=1.1) AND (CATEGORY=food) OR (CATEGORY=clothing)
REPORT ONCE
DIGEST 51oJ34Kfijcplswe1fvurN4
```

Here, we see that a donor who wishes to be anonymous is donating $100 per month over the period of a year, with the stated aim of allocating the funds on a first-come-first-served (FCFS) basis in the categories of food and clothing. The schema refers to the product ontology that is being applied (for which options include `schema.org` and other such ontologies [Bri, WPH06, Hep08]). The donor also seeks a single report at the end of spending. Next, we see that the bank issues both a timestamp that's sufficient to serialize such statements, and an encrypted token to be used as the reference so that later, when the amounts are cashed by vendors, a vendor uses token to reference this donation. Finally, the bank also includes a signed digest to detect whether the statement itself has been tampered with. The statement is then written to the public distributed ledger, awaiting further action when a node in the system performs pairing as part of writing the next block in the blockchain.

Eventually, when algorithmic pairing occurs, such "buy orders" are paired with projects or recipients and eventually spent, with a report that aggregates over time using the token reference number so that it can be returned to the donor.

A simpler version allows a donor or a representative to query the system for existing projects that might suit their interests, as in: Query existing proposals:

```
FROM PSEUDONYM NightHawk (ID= ...)
FIND PROJECTS
WHERE (SCHEMA=1.1) AND (CATEGORY=food) OR (CATEGORY=clothing)
```

(For brevity, we will omit the ID and timestamp detail.) Since these carry no obligation, no security measures are necessary. A donor may wish to change the `DECIDE` clause by seeking to match existing contributions: `DECIDE MATCH name=...` names a particular pseudonym whereas leaving the name out results in the first match that meets all the criteria. Thus, a matching contribution merely adds the requirement that another contribution has already been made of at least the same amount. And lastly, a donor can issue a rating, as in:

```
FROM ANON (ID= ...)
RATE Bright Ray Charity AS 4.3/5
```

3.3.3 Queries from recipient aggregators

One of the most important features in enabling algorithmic pairing is to allow recipient aggregators to create a statement of intent – that is, to define a project. Here is an example:

```
FROM Bright Ray Charity (ID=...)
DEFINE PROJECT Food Drive 2018
GOAL $10,000
WHERE (SCHEMA=1.1) AND (CATEGORY=food)
```

Clearly, this is a project that can be paired with the donation example from the previous section. Here is an example of issuing a call to vendors for a particular food item:

```
FROM Bright Ray Charity (ID=...)
PROJECT Food Drive 2018
VENDOR RFP $1000
WHERE (SCHEMA=1.1) AND (CATEGORY=food.canned.soup)
DESCRIPTION URL http://brightraycharity/fooddrive2018.html
```

Note the use of a hierarchical ontology (in Schema 1.1). Recipient aggregators may also query existing donations in the ledger, rate vendors, and are expected to report on their spending as in:

```
FROM Bright Ray Charity (ID=...)
EXPENSE=$550.00 TO VENDOR Fresh House Grocery
WHERE (SCHEMA=1.1) AND PROJECT=Food Drive 2018
      AND (CATEGORY=food.canned.soup)
```

Once a vendor verifies this by posting a receipt to the ledger, such a claim can be verified. The more verified claims that a charity has, the higher its rating.

3.3.4 Queries from vendors

A vendor needs to be able to query existing calls, make bids, and rate recipient aggregators. As an example of making a bid, consider:

```
FROM Fresh House Grocery (ID=...)
BID $550 TO Bright Ray Charity
WHERE (SCHEMA=1.1) AND PROJECT=Food Drive 2018
      AND (CATEGORY=food.canned.soup)
DESCRIPTION URL http://freshhousegrocery/search.html?prodid=98735
```

Then, it's up to a recipient to accept the bid and complete the transaction. A completed transaction then result in the vendor posting the receipt on the ledger.

3.3.5 Other types of queries, self-governance

Stakeholders include regulators who seek to examine the ledger and to independently score vendors and aggregators. Since these are straightforward and mundane, we do not describe these here.

We envisions two types of actions in self-governance, both of which will use a DCL-driven mechanism for voting. The first includes voting on mundane changes to the information used by the system, such as schemas. Anybody can propose a change of schema or an alternative, and if it receives a majority vote, can expect to use the new schema. The second is more controversial: the ability to vote that a stakeholder be temporarily blocked out of the system for perceived malicious behavior. For example, consider a charity that is too aggressive in flooding the system with too many tiny projects aimed at gaming the pairing algorithm. However, this somewhat drastic measure itself needs to be limited to avoid collusion dynamics, a topic for future work.

Instead of prescribing the DCL syntax (which would be straightforward), we merely list two options for self-governance that are orthogonal to the rest of the system. The first is merely to appoint actual people onto a governing board, who would then examine proposals and oversee a voting procedure and follow up action. The second is to formulate a detailed automated mechanism for voting with rules for scheduling votes, counting, follow up actions and the like. To propose something for a vote, any stakeholder can issue a statement to the ledger along the lines of

```
FROM ID=...
PROPOSE ...
VOTE DEADLINE ...
DESCRIPTION URL http://brightraycharity/petition5.html
```

However, one needs rules to limit the number from each stakeholder and enough time for humans in the loop to digest and act.

3.4 Distributed-Ledger and Consensus

A ledger entry is merely a DCL statement. Recall that a DCL statement contains a stakeholder ID as well as a unique statement ID issued by the central bank. Once a DCL statement is in a resolved block of the ledger it will also have a serial number so that all resolved DCL statements are uniquely ordered, as are the larger blocks in which they are contained. This is the purpose of operating a blockchain: to achieve distributed consensus by offering participation incentives and yet prevent domination by any one player.

Any registered (an account with a central bank) stakeholder can operate a computational node in the distributed ledger system. We assume that regulators already have an incentive to become nodes. We describe in the following section an incentive mechanism for aggregators and vendors to participate.

Since DCL statements (ledger entries) flow into various nodes in the system, there needs to be a way to organize the serialization and resolution of blocks into a blockchain. Our approach is to combine a leader-based consensus mechanism (such as RAFT [OO14]), with incentives to spread the leadership capability and prevent any one node from dominating as leader. In particular:

- Each node in the system receives ledger entries from any node and can generate ledger entries.
- A node broadcasts its ledger entries to its neighbors (or any subset of nodes) in the system.
- A leader is elected as in the RAFT algorithm [OO14].
- Only a leader has the authority to resolve the next block in the blockchain.
- A leader resolves the next block through a sequence of computations to: (1) determine the potential ledger entries in the current block; (2) compute pairing if needed; (3) hash and sign the block; (4) broadcast the current block.
- The resolution of the next block triggers a new leader election (to prevent domination). Also, the current leader is ineligible to become the next leader in the next N rounds of leader elections. Here, N is a parameter of the system.
- Leader elections are also triggered by timeouts so that no leader can deny service by merely blocking further action.

When a leader resolves a block, it is awarded a participation point. These participation points accumulate as an incentive. We propose two alternatives. One is for the central bank to charge a tiny fee for each actual transaction and to let that fee accumulate into a fund used to reward aggregators or vendors who have accumulated sufficient points. Another approach is to allow the points generated to be incorporated into the ratings of aggregators and vendors so that they compete for visibility and recognition.

3.5 Algorithmic pairing

The purpose of algorithmic pairing is to have each donation be paired with a suitable project or recipient according to the desired criteria. A secondary goal is to ensure some degree of spread amongst multiple competing projects that satisfy the criteria. Finally, we also wish to design a system that is robust to gaming or attacks.

Our approach is to use a variation of the classic Gale-Shapley algorithm for the stable marriage problem [IM08]. In the most basic version of the problem, there are M type-A items to be paired with N type-B items (most often $M = N$), where each item ranks every member of the other type. Under certain conditions, a simple algorithm provides a stable "marriage." Much research has gone into variations as well as attempts to make the resulting allocation fair to both side [IM08], some of which are now a fundamental component in computing systems [MS15]. In our case, we can choose the donations as type A, and potential recipients or projects as type B. However, we need a variation that must account for the following:

- In our case, preferences are often binary: either a project satisfies the criteria or not. Thus, to decide amongst equivalently satisfying projects, the projects can be randomized before selecting the next one. For transparency, the method of randomization and the particular choice made would need be made public and written back into the ledger. A simple way to achieve this is to use a known algorithm (such as a Lehmer generator) where the seed uses a combination of the timestamp of the last ledger entry in the most recently resolved block and that block's hash. This way, the sequence is reasonably unpredictable.
- It will almost never be the case that M=N and so some items will not receive a pairing and will have to "wait." Also, it is equally probable that the current set of donations are not sufficient to meet the goal of a particular project, and thus, the difference will remain to be fulfilled. Thus, the pairing algorithm will need to go as far back as needed in the blockchain to find incompletely-satisfied projects.
- It should be possible to prioritize projects that are either very early or have received very little so that no project remains unlucky for too long or is totally unfunded.
- Finally, donations can also specify that rankings should be used, in which case, there is a natural preference order. However, to prevent a few aggregators from dominating, some randomness can be inserted to enable spreading the wealth.

3.6 Reporting and donor interfaces

One of the computational tasks of a leader is to aggregate expenditure data for donors. Here, the algorithm is simple: examine posted receipts, find the originating transactions and post reports to the ledger, one for each donor whose funds were used towards a receipt. Thus, at any given time, a donor might have a list of such report entries awaiting them.

How does an actual report entry finally get back to a donor? We do not expect end users to be browsing ledger entries. Instead, just as end-users do not actually write or read SQL, in our system we expect donor websites to incorporate this functionality and then present end users with a friendly web-interface. Thus, an individual donor signs into a donor aggregator that provides a convenient interface. The aggregator then hands off the transaction to the central bank (in much the same way an ecommerce site involves a digital payment system as part of the transaction), after which the aggregator then issues a ledger entries. The same aggregator browses the resolved blocks to see if any reports need to be displayed to the end user.

3.7 Validation and challenge

A third computational task for any leader, beyond pairing and reporting, is to cross-check the entries written by other leaders. This feature is at the heart of public verifiability, so that any leader who games the system or even makes a mistake is identified, followed by appropriate corrective action.

There are many ways to organize cross-checking. One simple way is to have the current leader cross-check the previously resolved block. Another way is to have the current leader cross-check a randomly selected previous block (but where the random generation is fully determined based on hashes and timestamps). These can be combined so that there are multiple checks.

What happens when an inconsistency is discovered? It is tempting to try and design an automated way to "undo" a resolved block and re-resolve the block following an inconsistency. However, this may lead to catastrophic consequences (actual expenditures) that are difficult to roll back. Instead, we propose that an inconsistency be escalated to human intervention, through a steering committee or governing board. If the penalty is severe enough (denial of participation for a year, for example), there is little incentive for legitimate entities to cheat.

4 Security

We present our threat model and specific details that allow for building a system that is robust to the given threat.

1. An adversary might remove or change a donation
2. A donor might go back on a pledge
3. A bank might withhold funds
4. A vendor might use the same receipt to claim money from multiple donations (double claim).
5. Denial of service by the system participants i.e., suppressing a DCL query or activity.

In order to thwart the above threats, we define the system's transaction design and the process of maintaining consistency. We will reflect back on the threats to explain how the design removes the risks posed by the above threats.

4.1 Transaction structure

We propose the following structure for a transaction in our system. First each transaction has a header that consists of the header length field, version number, transaction id, originator id, originator's bank id, time of creation, status code, link address, retry flag, a retry count and an optional expiration date. Following the header, the transaction body consist of the contents of the DCL query. Finally, a signed hash of the transaction header and body are attached to the transaction. We represent a transaction with the following notation: $T_a(status, tid_i)$ where a is the originator id and tid_i is the i^{th} transaction id. The digitally signed hash of the transaction is represented as $E(K_a, H(T_a(status, tid_i)))$. Having this structure helps us maintain the data that is needed to thwart each one of the threats mentioned above.

4.2 Types of Transactions

Table 1: Table: Donation status codes

Name	Status Code	Value
Donation Pending	'P'	001
Donation Expired	'E'	010
Donation Contract	'C'	011
Donation Claimed	'D'	100

We propose four types of transactions, Donation pending, Donation expired, Donation under contract and finally Donation claimed (see Figure 1). When a donor makes a pledge, the Donation pending transaction is created which appears as a pending payment against the credit card or bank. These funds remain in a pending state until the donation expired or Donation claimed transactions are created.

The link address field in the Donation Pending transaction is set to the bank's transaction authorization code. This creates a link between the digital currency and an actual bank authorization binding both the donor and the bank into honoring the pledge. The header field 'link address' in the Donation Contract transaction is set to the Donation pending transaction id and that of the Donation Claimed contains Donation Contract's transaction id. The Donation Expired transaction carries the Donation Pending transaction id in its link address field. This information ties each transaction directly and immutably with the original donation.

4.3 Transaction chaining and Consensus

We propose a chained merkel root based ledger similar to that of Bitcoin. Thus lets say a transaction $T_a(status, tid_i)$ is generated by an originator a. Then $T_a(status, tid_i)||E(K_a, H(T_a(status, tid_i)))$ becomes a blockchain transaction. The signature can either be of a donor or the donor's bank. A participant q in the blockchain consensus process, constructs a block as a merkle tree composed of all transactions that were received during a block generation period δ. In addition, the merkle tree of block B_i consists of the merkle root of B_{i-1} signed by the creator r of block $B(i-1)$. Thus $B(i-1)||E(K_r, H(B(i-1)))$ becomes the leftmost leaf in the block $B(i)$. We do not believe that a proof-of-work like blockchain is sustainable in this scenario. Therefore, as explained earlier, we use a leader-based distributed consensus algorithm (such as RAFT), with sufficient additions to ensure proper functioning in the presence of malicious participants in the system. Participants in the consensus system might be charities, aggregators and vendors who have a vested interest in maintaining an honest and verifiable chain.

4.4 Non-repudiation of transactions

Each transaction of type P and E is signed by a private key of the donor or the donor's bank. The aggregator/charity signs transactions of type C and the vendor sign's the donation claimed transaction. These signatures help us achieve non-repudiation for each stakeholder. The charity and donor may still keep paper trail or electronic evidence of their physical transactions for internal and specific audit.

The transactions are structured as components of a linked data structure due to the inclusion of the address of the previous transaction. Thus a claim transaction can be linked to only one donation claimed transaction. If a claim cannot be fulfilled by a single donation, it will need to be broken down into multiple transactions to be posted to claim money from different donation pending transactions (similar to real life partial payments when multiple methods of payment are used to pay for an expensive purchase).

Due to the linked structure, an independent validator can easily create an accounting balance sheet of any donation and all claims against it to verify how a donation has been spent.

4.5 Defending against threats

The threat of tampering with any transaction is removed due to inherent properties of the blockchain. Each block consists of the signed merkle root of the previous block, which makes the chain self-certifiable. The block creator signs the merkle root of each block it creates. Thus any inconsistencies can be traced back to the participant who constructed a bad block provided the participants in the consensus system remain honest.

The threat of donor's going back on a pledge is solved through standard banking processes as donors authorize the funds they are committing through their bank/credit accounts. In addition, source non-repudiation is achieved due to the donor's or their bank's digital signature accompanying a donation pending transaction.

The donation pending transaction includes the id of the bank that authorizes the payment as well as the transaction id that the bank creates. Thus a pending donation can be traced back to the original bank that authorized the transaction and issued a transaction id. Therefore, the bank is bound by federal regulations and must honor the commitment to release funds when the transaction enters the claimed status.

The vendor signs a donation claim transaction and includes the address of the original donation pending transaction in the link field of the transaction header. The body of the claimed transaction itself contains the receipt of the delivery of the goods signed by the charity. The recipient bank generates the transaction id for the donation claimed transaction. A vendor would not be able to post fraudulent claims unless they collude with a charity. Through our openly verifiable system, such collusion can be easily detected by someone who has knowledge of the ground truth in these transactions, which makes it easy to raise alarm and alert the appropriate law enforcement authority.

4.6 Denial of service by the consensus leader

We have earlier proposed that the Raft Consensus protocol can be used as an alternative to the expensive proof based consensus used in crypto-currencies. Denial of service is one threat that is inherent to a leader based consensus mechanism. In computer networking protocols, a standard and perhaps the only way to provide reliable two way communication is to retry a failed message. We will also follow a protocol based approach to this threat which we illustrate in the following steps:

1. A donor or stakeholder S sends a DCL query to the current leader L_i.
2. The leader L_i must acknowledge the query with a return message that contains sufficient information and cryptographic protection so that it can be uniquely tied to the original DCL query.
3. The sender logs the acknowledgment as proof that the query was received by the leader, waits for a block generation period and then queries the blockchain to ensure that the transaction has been entered in the ledger.
4. If the sender does not receive an acknowledgement it increments the retry count field in the transaction header, sets the retry flag to true and sends the transaction again.
5. The above step may be repeated until a maximum number of retries have failed or a successful acknowledgement is received.
6. If the sender fails to receive the acknowledgement after the maximum number of attempts, it waits for the next leader and sends the query with retry flag set to true and retry count reset to 0.
7. If after receiving the acknowledgement, the sender does not find the transaction in the next block that was created, the sender would follow up by sending the acknowledgement and the transaction to the next leader.

Unfortunately, there is no simple method to recover from failures such as denial of service by a malicious or faulty leader. The above protocol steps enable senders by generating sufficient proofs that their transactions were acknowledged. They can be extended to allow the consensus participants to build a blacklist of malicious/faulty leaders.

5 Related Work

In earlier sections, we have cited related work where appropriate to provide context, including related work in scholarship on charitable donations, and smart-contract languages. This section focuses on the most closely related work that provides additional context withing the areas of cryptocurrencies, consensus and Byzantine consensus.

Crypto-currencies have given a new life to research in distributed, asynchronous consensus. The most prominent consensus system used today is the bitcoin miner networks which is based on proof-of-work [JJ99]. Proof-of-work, based on Adam Back's hashcash [Bac03] requires the user to solve a cryptographic puzzle. The user computes a string that when hashed gives a value with a certain number of leading zeros. The problem of computing the string is computationally extensive but verification is cheap. In Bitcoin blockchain, the proof-of-work is used to implement a distributed timestamp server that maintains a public ledger of the time order of transactions that used the crypto-currencies. Each transaction is sent as a broadcast to the entire mining network. Miners compute the next block using the transactions they receive. The first miner to compute a block, broadcasts to the others. If the block can be verified, other miners convey their acceptance by working on the next block using the hash of the previous block. This timestamp ledger acts as a safeguard against multiple spending of the same coin. Proof-of-work requires heavy compute resources and therefore reversing a block requires re-doing the work. If the block chain has advanced, any alteration or tampering may need computation of several blocks. Therefore, the integrity of the blockchain is guaranteed because of the computational difficulty of changing a block after it has been committed into the chain. The system requires a minimum of 51% to turn rogue for compromising the security of the system. Therefore, the security is largely dependent on the availability of large number of miners which reduces the possibility that a cartel could take over by acquiring 51% of the mining power of the network. In order to maintain the difficulty level of computing blocks, the number of leading zeroes needed in the hash value is periodically revised. In addition, the transaction fee for computing a block needs to be adjusted according to the minting rate of crypto currency. When the minting rate decreases, the transaction fee needs to be increased accordingly in order to maintain the incentives that attract a large number of miners needed to ensure security. If the transaction fee falls close to zero, the system may suffer from the tragedy of the commons with honest miners leaving the mining network, making it vulnerable to the 51% attack.

An alternative distributed consensus is called proof-of-stake [But13] where the system consists of several currency holders who lock their currency for a certain amount of time, i.e., place a stake. In order to extend the consensus, the stakeholder signs the next block/extension. The proof-of-stake algorithm may randomly selects some currency holders to sign an extension. The system is secure as long as the stakeholder is locked in for a sufficient amount of time that prevents a stakeholder from first cashing out his stake and then create a new fork starting at the point in history where the stakeholder had control. The security of the system still depends on having a critical mass of miners to reduce the possibility of collusion. A hybrid method called Ppcoin was proposed [KN12] in 2011 where the system consisted of two blocks. The first type of block, called kernel, is generated using the proof-of-work method. The miner who generates the kernel then creates a special transaction called coinstake in which the miner pays a stake to gain the privilege of generating the next block. The generation of the kernel using proof-to-work introduces a stochastic process that ensures random selection of stakeholder. Since then several systems based on proof of stake have been proposed and launched [But14, Vas14].

Byzantine fault tolerance (BFT) has been studied in the context of distributed processing and distributed databases [CL$^+$99]. A BFT based system functions correctly as long as the number of faulty entities is less than one-third of the total. When the conditions of fault tolerance is achieved, the system is said to have quorum. In the context of public ledgers such as in crypto-currencies, a system of Federated Byzantine agreement system (FBAS) has been proposed [SYB14, Maz15] under the names Ripple and Stellar. They attempt to reduce the latency involved in achieving quorum whe there are a large number of miners using BFT to achieve the consensus. Nodes participating in FBAS select their own trust group to form quorum slices. A single node may be a part of one or more such slices. In order to have protection against collusion attacks, the network must have quorum intersection i.e., there may not be any pair of quorum slices q_i, q_j with membership sets v_i, v_j such that $v_i \cap v_j = \emptyset$. This requirement ensures that no two quorum slices accept contradictory views of a block. In addition, when byzantine nodes are present, a robust FBAS system with a set of nodes V will consist of a dispensable set $D \subset V$ such that the quorum slices formed from the set V
D has quorum intersection as well as quorum. D may contain the set of byzantine nodes B as well as nodes that have been blocked by nodes in set B from acting correctly.

A hybrid proof-of-stake and BA system was proposed to enable consensus without mining (Tendermint) [Kwo14, Buc16]. Thus, instead of sending transactions as broadcasts to the network, they can be sent to a single node in the network. Transactions are validated by users who have coins at stake, deposited as a bond deposit using a bond transaction. Thus validators have the incentive to only sign valid transaction because if found guilty of signing a fraudulent transaction, they might be forced to forfeit their bond. A block consists of valid transactions and signatures. Only blocks with a 2/3 majority of validator signatures is considered committed to the ledger. The signatures are obtained through a partially synchronous BA algorithm that assumes an upper bound on message latency. The consensus system ensures that if a fork is created, both branches may not consist of all valid transactions i.e., one of the branch must contain some duplicitous validator signatures. Any block holder can detect the guilty validator. As long as the guilty is found within the unbonding period i.e,, before the validator cashes his bond out, the system can recover from the fraudulent transaction. In order to thwart long range double spend attacks, users must re-sync their blockchain continually within the unbonding period.

6 Discussion and Conclusions

This paper attempts to explain how blockchain technology can be adapted for social causes. In the absence of an incentive model as in digital currency, we use a well known leader based consensus scheme from distributed computing. We propose algorithmic pairing to give donors more control over how their money is used and to ease the burden of searching for best matches to their intentions. It also provides a fair distribution of money by removing human biases, gives better access to funds to smaller organizations and NGOs, reducing the outsize power currently held by foundations and big charitable organizations.

Most importantly, the features of the system promote social good through incentives for transparency, accountability and participation.

Acknowledgements: Some of the ideas in this paper first appeared in a whitepaper written by the lead author [Sim16].
Some of this work is based on the work supported by the National Science Foundation under Grant No. 1742919 and PSC-CUNY Grant No. 60814-00 48

References

[A.B15] A.Butcher. What donors want when it comes to communication. *Nonprofit Quarterly*, September 2015.

[AM16] Inés Alegre and Melina Moleskis. Crowdfunding: A review and research agenda. 2016.

[Bac03] Adam Back. The hashcash proof-of-work function. *Draft-Hashcash-back-00, Internet-Draft Created,(Jun. 2003)*, 2003.

[BBD$^+$06] Rafael H Bordini, Lars Braubach, Mehdi Dastani, A El F Seghrouchni, Jorge J Gomez-Sanz, Joao Leite, Gregory O'Hare, Alexander Pokahr, and Alessandro Ricci. A survey of programming languages and platforms for multi-agent systems. *Informatica*, 30(1), 2006.

[Bri]

[Buc16] Ethan Buchman. *Tendermint: Byzantine Fault Tolerance in the Age of Blockchains*. PhD thesis, 2016.

[But13] Vitalik Buterin. What proof of stake is and why it matters. *Bitcoin Magazine, August*, 26, 2013.

[But14] Vitalik Buterin. Slasher: A punitive proof-of-stake algorithm. *Ethereum Blog URL: https://blog. ethereum. org/2014/01/15/slasher-a-punitive-proof-of-stake-algorithm*, 2014.

[Cau13] Root Cause. Informed giving: Information donors want and how non-profits can provide it. *Root Cause, Boston, MA. http://www. rootcause. org/docs/Blog/Informed_Giving_Full_Report. pdf*, 2013.

[CL$^+$99] Miguel Castro, Barbara Liskov, et al. Practical byzantine fault tolerance. In *OSDI*, volume 99, pages 173–186, 1999.

[DAK$^+$16] Kevin Delmolino, Mitchell Arnett, Ahmed Kosba, Andrew Miller, and Elaine Shi. Step by step towards creating a safe smart contract: Lessons and insights from a cryptocurrency lab. In *International Conference on Financial Cryptography and Data Security*, pages 79–94. Springer, 2016.

[DuP17] Quinn DuPont. Experiments in algorithmic governance: A history and ethnography of the dao, a failed decentralized autonomous organization. *Bitcoin and Beyond: Cryptocurrencies, Blockchains and Global Governance. Routledge*, 2017.

[FM13] Reid F. and Harrigan M. An analysis of anonymity in the bitcoin system. In *In: Altshuler Y., Elovici Y., Cremers A., Aharony N., Pentland A. (eds) Security and Privacy in Social Networks. Springer, New York, NY*, 2013.

[Hep08] Martin Hepp. Goodrelations: An ontology for describing products and services offers on the web. In *Proceedings of the 16th International Conference on Knowledge Engineering and Knowledge Management (EKAW2008), Acitrezza, Italy*, 2008.

[IM08] Kazuo Iwama and Shuichi Miyazaki. A survey of the stable marriage problem and its variants. In *Informatics Education and Research for Knowledge-Circulating Society, 2008. ICKS 2008. International Conference on*, pages 131–136. IEEE, 2008.

[JJ99] Markus Jakobsson and Ari Juels. Proofs of work and bread pudding protocols. In *Secure Information Networks*, pages 258–272. Springer, 1999.

[KB96] Michael Kantor and James H Burrows. Electronic data interchange (edi). *National Institute of Standards and Technology*, 1996.

[KN12] Sunny King and Scott Nadal. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. *self-published paper, August*, 19, 2012.

[Kwo14] Jae Kwon. Tendermint: Consensus without mining. *URL http://tendermint. com/docs/tendermint {_} v04. pdf*, 2014.

[Maz15] David Mazieres. The stellar consensus protocol: A federated model for internet-level consensus. *Draft, Stellar Development Foundation, 15th May, available at: https://www. stellar. org/papers/stellarconsensus-protocol. pdf (accessed 23rd May, 2015)*, 2015.

[McG97] Simon McGrath. Giving donors good reason to give again. *International Journal of Nonprofit and Voluntary Sector Marketing*, 2(2):125–135, 1997.

[MS15] Bruce M Maggs and Ramesh K Sitaraman. Algorithmic nuggets in content delivery. *ACM SIGCOMM Computer Communication Review*, 45(3):52–66, 2015.

[OO14] Diego Ongaro and John K Ousterhout. In search of an understandable consensus algorithm. In *USENIX Annual Technical Conference*, pages 305–319, 2014.

[RB16] R.Dietz and B.Keller. Donor loyalty study: A deep dive into donor behaviors and attitudes. *Nonprofit Times*, 2016.

[Sim16] Rahul Simha. Directed cash: A distributed ledger for charitable donations. *Whitepaper, Department of Computer Science, George Washington University*, 2016.

[STM16] Pablo Lamela Seijas, Simon J Thompson, and Darryl McAdams. Scripting smart contracts for distributed ledger technology. *IACR Cryptology ePrint Archive*, 2016:1156, 2016.

[SYB14] David Schwartz, Noah Youngs, and Arthur Britto. The ripple protocol consensus algorithm. *Ripple Labs Inc White Paper*, page 5, 2014.

[Vas14] Pavel Vasin. Blackcoins proof-of-stake protocol v2, 2014.

[Ves06] Lise Vesterlund. Why do people give. *The nonprofit sector: A research handbook*, 2:168–190, 2006.

[WPH06] T.D. Wang, B. Parsia, and J. Hendler. A survey of the web ontology landscape. *The Semantic Web - ISWC 2006. Lecture Notes in Computer Science.*, 2006.