Gift Notes: A Fundraising, DID, Stablecoin, and DMM Platform

Adam Z Winter
Seattle, WA, USA
adam@giftnotes.org
2019/01/02

---------- Rough Draft ----------

Abstract

The following describes a transaction-oriented, decentralized, system of raising money for charitable non-profit organizations by creating a stablecoin with the capacity for high transaction volumes, and giving the transaction fees directly to the charities. Participants will be able to choose which charity they want to support by routing their transactions through the node that is controlled by that charity. Additionally, a DID (Decentralized Identification) framework is provided, which dovetails with a built-in Decentralized Money Market, to create Stake-in-Identity as a means of verifying the authenticity of a DID. A request-only Proof-of-Consensus algorithm is used that provides the possibility of 100% consensus in minimal round trips, while also providing maximum fault tolerance. Furthermore, participants can create superimposed ledgers for issuing their own tokens, for a fee, but with no coding necessary. In one type of these ledgers, participants will chose to accept permission to write IOU's to a creditor, thereby preserving a space for a creditor to act as an arbiter and escrow agent between consumers and retailers, while keeping both lender and borrower accountable to the terms of the financing. Existing gift card technology, and the legal ability of businesses to issue gift cards (even in digital form), will be used as a means of backing and decentralization. It is in this respect that the resulting stablecoin is named **Gift Notes**. This base stability provides the opportunity for the lending platform which further promotes the value of Gift Notes through supply and demand, inasmuch as there is demand for money that makes money. Meanwhile, that market is opened up to wide participation and makes the benefits of it available to everyone. Lastly, an ongoing 50/50 raffle, using **Proof-of-Decentralization**, will be built into the system to further raise money for the charities that provide the nodes for the network. This system aspires to reach the yet-unattained goal of wide adoption as everyday money, while striving to decentralize all the things about it.

Transaction Fees

The node to which a transaction is initially broadcast is a fundamental matter when considering the structure of a distributed ledger. Surprisingly, it seems the matter is rarely discussed, for its own sake. How many wallets specifically ask the user what node they want to broadcast to? How many users would even have an opinion about it? Nevertheless, transaction fees are often required which end up being included as part of a block reward. This takes the matter of the transaction fee, and the matter of coins being created out of thin air, and tangles them up with each other. In a PoW scheme, it ignores the value of the decentralization that those nodes provide in favor of the work that miners are doing. Meanwhile, and ubiquitously, that power is centralized into pools.

In looking at the relationship between the participant and the node they route their transactions to, we find an opportunity for citizens to exercise democratic influence on the individual nodes that operate in a representative democracy. One specific way we can accomplish this is by giving the transaction fees that participants pay directly to the node that they route their transactions through. In this way, the people can support, or refuse to support, the charities of their choice every time they send a transaction.

An acceptable section 501(c)(3) charitable non-profit organization will have a board of trustees with many members who are legally required to act according to guidelines that are meant to keep charities acting charitable. The law provides for significant consequences to anyone who uses their organization's charitable status as a personal checkbook or means of tax evasion. Be that as it may, rather than putting stock in negative consequences to establish trust, we should see the people's belief in an organization's will to do good, for goodness sake, manifest in the support for that organization. More to the point, democracy will exist at multiple levels within the system: by participants supporting the charity of their choice, within the consensus algorithm between the nodes, and over the decision making with regards to each individual node, by a board of trustees.

Proof-of-Consensus

Simply put, the Proof-of-Consensus algorithm seeks to communicate to all nodes what every node thinks should be the next block. If the required majority of nodes agree, then it's settled and they move on to the next block. If no such majority is established, then they compare notes and try again. How to do that part, specifically, will be discussed later.

There are two probable structures that can be used to build this consensus algorithm. The first conforms to the idealism of the public chain where anyone and everyone can run a full node. For the purposes of what we're trying to accomplish here, this would be the structure where any and all charitable non-profits can run a full node. Naturally, this raises the issue of how to keep people who don't meet the description of "charitable non-profit" from running a node, or malicious actors from creating nodes that interfere with our consensus algorithm. In our idealized public blockchain, we might weight very-heavily the votes of a node that procured transactions from the most unique participants. Thereby, a node (held by a charity or not) that isn't as popular won't hold as much weight in establishing consensus. Else we have Sybil attacks run rampant, this requires that we have a way to establish that a given participant is a unique individual. With Stake-in-Identity we might have just that. However, we have reasons to want to limit the total number of nodes participating to a manageable number; "manageable" in the sense that communicating to every node what every other nodes thinks should be the next block gets more difficult with more nodes. This brings us to the other structure that we will continue with.

By communicating to every node what every other node thinks should be the next block, we don't need to broadcast the winning block from a single point of failure, or wait for the time it takes for a new block to propagate through the network (creating the likelihood of a fork, and orphans). In the process of communicating all proposed blocks to all nodes, we can also communicate all the transactions for the next block at the same time. So, if we are to limit the number of nodes to as many as is manageable, then we can maintain a high level of decentralization while removing those single points of failure. The process for applying this limit can be done in a democratic and decentralized way that separates itself from being called "private" or "permissioned."

Rather than having a unique node list (UNL) that *allows* (or not) other nodes to post information to their memory and trigger the execution of code, communication between nodes will happen *only* by request. That is, a node will get information from another node only when it asks for it, and a node will get information from every node by asking every node for that information. Furthermore, for the sake of making the entire system open-source, public, verifiable, and auditable, any actor may request the same information from any of the nodes in the same manner. Each node will simply decide to get its information from the specific sources of its choice, as opposed to actively rejecting any possible node outside of that list. Any node could choose to get information from a node outside of that list, but the consensus process would inevitably find that incompatible. Therefore, no participant would choose to transmit transactions to an outsider node, and the very notion of "permission" becomes irrelevant. If a charity wishes to usher a new node into the fold, this can be accomplished simply by sending a specific token to the new node, and requiring the signature of a majority of other nodes on that transaction for it to be validated. At that point, a dissenting minority might slow down the consensus of each block by a trivial and negligible amount, as a matter of protest, or choose to go with the flow for the sake efficiency and the incentive to see the system work well. The entire supply of said tokens could be created and distributed evenly to the initial active nodes to be further distributed from there at later times. This would set an initial limit on how many nodes there can be while decentralizing the process of distributing them.

Let's get into the particulars of the consensus algorithm. Consider BitTorrent and the act of downloading parts of a file from a thousand different sources at the same time. Given the proper bandwidth, a server will have no trouble doing this, and would clear something the size of a transaction block off its plate very quickly. In this way, we will simultaneously download from every node all of the newest transactions that were sent to each of those nodes, and what each of those nodes thinks should be the next confirmed block. Each block will contain one **batch** of transactions from each node. A **batch** will be a signed group of transactions that were all posted within the same time frame. For example, if we are using 30-second batches, then all of the new transactions a node receives between 2018-12-20-10:29:00 and 2018-12-20-10:29:29 will be bundled together, marked as the 2018-12-20-10:29:00 batch and that entire data set signed by the node to finalize the batch. Once a node has collected all the 10:29:00 batches, that node will bundle all the batches into a block, mark it as the 10:29:00 block, and sign that entire data set. Note that, in order for the blocks to agree, the order of the batches must be the same for everyone. We can simply arrange them alphabetically by domain name, or by another identifier. By request, a node will receive from another node the latest batch(s), and the node's current take on the state of all nodes (the proposed block). Once a node has collected "all" (some nodes might not respond) of the proposed blocks, if the consensus meets the **difficulty level**, then each node moves on to the next block. Otherwise, all nodes proceed to the next difficulty level. **Difficulty level** refers to a variable threshold that decreases with each iteration of the consensus process as applied to a single block, where the highest difficulty level cannot have a threshold below 51%. Higher difficulty levels indicate that the network is having more difficulty finding a consensus.

Naturally, when it comes to retrieving the blocks from each node, since the details of each batch will already be in-hand, only the block hash, signature, and representation of each batch will be needed. Of course, that hash will reference the previous block that we are working from (blockchain), and be the identifier for the block, and signatures will use padded versions of those hashes.

Let's take this step-by-step; we'll assume 1,000 nodes for sufficient decentralization, and use some possible thresholds for example:

Level	Actions

L_0	• Sign your batch.
	  ○ If you have no new transaction, sign empty batch with random data (for security)
	• Collect 1,000 signed batches
	• Sort, label, and sign the 1,000 batches as your proposed block
	  ○ A representation for each node must be made. If a node is non-responsive, your block will include your representation of that fact in place of a batch.

L_{1a}	• Collect 1,000 signed (proposed) blocks
	  ○ In time elapsed since L_0 new batches may also be available to get concurrently
	• If 95% of nodes agree, block accepted, continue to next block (L_{1b})
	  ○ If this block is different from your own, get missing information from one of 95% of other nodes
	  ○ If this block does not include your batch, push those transactions to future batch
	• Else, get missing information from nodes that do not agree with yours
	  ○ If another node has the batch from a node that was previously-non-responsive, get the batch from either.
	• Label your new proposed block as L_2 and sign
	  ○ Mark other batches as non-responsive/error, or faulty as necessary.
	  ○ Node is faulty if node has signed contradictory batches. Keep evidence of this for other nodes to retrieve.

L_{2a}	• Collect L_2 signed blocks • If 75% of nodes agree, block accepted, continue to next block (L_{1b}) • Else, repeat steps from L_1 and continue to L_3 • If another node asks for your L_2 block, but L_1 succeeded, they will receive the L_1 block proof of consensus.
L_{3a}	• 51% threshold • Levels can continue beyond this point but all errors should be able to be identified and agreed on in L_2
L_{1b}	• At this point, depending on how long the previous block took, you may have multiple batches waiting in queue. • For each batch, create a block that includes that batch and all of the outstanding batches that proceeded it. For example, if you have three batches ready, create three blocks: block1 = [batch1], block2 = [batch1, batch2], block3 = [batch1, batch2, batch3] • Collect same blocks from all other nodes, and repeat steps from L_{1a} for each block • No. of batches in successful block will be the most that meets target first ○ i.e. as many as we can agree on ASAP

Let's look at the fault tolerance of this algorithm by considering the possible actions of a malicious actor in control of a single node. The actions are limited to what the actor can leave sitting there for another node to pick up; 1) bad data, 2) connection timeouts, 3) gigantic batches/blocks, 4) contradictory batches/blocks. Of course, a fundamental function of blockchain technology is rejecting bad data. So, that is trivial. Connection timeouts are a simple reality that we have to accept, and we'll have to choose how long we want to set those. Similarly, we can set a batch-size limit to keep a malicious actor from lagging the system by creating batch files that take forever to download, or we can split the batches into limited-size chunks to prevent the same problem. Finally, is the matter of contradictory batches and blocks. This could take place in the form of a node "flickering" in and out of responsiveness, and/or giving different data sets to different nodes, in an attempt to confuse the consensus process. However, a flickering node can have its good data made available to any node by the ones who are able to retrieve it. This will have all those transactions distributed by the second level. In the case of contradictory blocks/batches other nodes will be reaching out to each other to compare notes. By the second level, most nodes should establish that a node has created, and signed, contradictory data. Since, malicious nodes are not actors in the communication between other nodes, even if 49% of nodes are compromised, malicious, and coordinating efforts, the other 51% should still be able to come to a consensus, though it may take a number of levels that is 49% of the total number of nodes. This would only have to be done once, though, as we would maintain the state of those nodes as faulty through subsequent blocks.

Block Rewards

First, transaction fees can go directly to the node that a transaction is routed through. Second, rewards do not have to be written into the block that establishes it. We can make it so that a person/entity entitled to a block reward has to write that reward to themselves as a separate transaction at a later time. Naturally, that transaction would reference the correct block, for verification purposes, and the reward will likely have an expiration date to claim by. In fact, using **Proof-of-Decentralization** will have us do it outside of the block. The core aspects of Proof-of-Decentralization are as follows:

- Block rewards are given based on the comparison of individual transactions to an unknown number found at a later time by hashing the inputs from multiple nodes. Such is the nature of hashing functions that only two nodes need be not-conspiring in order for sufficient randomness in the target hash, and the last entrant cannot have knowledge of all the other entries before submitting.

Metaphorically speaking, contestants write their own raffle ticket numbers and put them all in a hat. The winner has the number closest to the hash of all the inputs (arranged numerically, for example). Within our Proof-of-Consensus scheme, the hash of the agreed-upon block is the unknown number. That is, short of every single included node conspiring to give a reward to a particular person/node that they all agree on, none of them can possibly know what the hash of the block will be, given that it contains inputs from all of them. If one, or few, node(s) withholds it's batch in an attempt to create a winning transaction, consensus will likely be reached without it, and without its batch included. If many nodes withhold their batches, they render each other's efforts useless.

Third, there can, and will, be multiple types of block rewards to award. The new reward particular to this system is the 50/50 raffle. Participants/contestants will write a transaction that puts a quantized amount of money into the pool and the transaction itself represents one entry into the raffle. Per Proof-of-Decentralization, the hash of that transaction in comparison to the hash of the block that transaction is included in will decide if it is a winner. Specifically, we will use the mathematical difference between the block hash and the transaction hash. We could choose to pick the very best of these out of so-many blocks. However, we can set the likelihood of a winner within T transactions by setting a target threshold (difficulty) for that mathematical difference. Briefly, here's some code-intended math for that:

If we say that the probability of the difference between the transaction hash and the block hash, $prob(\Delta)$, is 1 out of T, and the probability that a random number Q (l digits in length) is less than, or equal to, a scalar s is $prob(Q \leq s)$, then:

$$prob(\Delta) = \frac{1}{T} = prob(Q \leq s)$$

where

$$s = \frac{10^l}{T}$$

and

$$l = \text{strlen(T)} = \text{the number of digits in } T\text{, in base10}$$

So, since, in our case, T is a constant, making s a constant, we can just grab the first l characters of Δ, map each to a random digit from 0 to 9, and see if they are less than or equal to s. Here's a quick visual check on that math:

T	$1/T$	l	10^l	s	$s/10^l$
2	1/2	1	10	5	5/10
10	1/10	2	100	10	10/100
50	1/50	2	100	2	2/100
100	1/100	3	1000	10	10/1000
200	1/200	3	1000	5	5/1000

This method could also be extended to base2, or any base, and allows for higher precision than counting leading zeros.

Naturally, half of the raffle pool will go to the originator of the transaction that met the threshold, and the other half will go to the node that the transaction was sent to. The description of this 50/50 raffle is just a prototype for a wide range of similar raffles that could be happening simultaneously. A threshold with higher difficulty would have a higher payout. So, multiple repeating raffles could be happening simultaneously, each with its own odds, and the participants could choose which they wanted to go for. Moreover, a quantized portion of the transaction fee from every transaction could constitute an entry in that raffle. We could, in fact, replace the transaction fees (as previously described) with the entire transaction fee going to this raffle, thereby creating an incentive for participants to simply use ♪Notes.

Another type of block reward has to do with initial fundraising for the entire project. If we give everything to charity, then there's no room for investors. Also, if we're backing a stablecoin at 1:1 backing, then we can't just make coins out of thin air to award as a block reward. Instead we'll create three other coins that will form a kind of DC bridge between the rate that value can be added for rewards, and the rate that block rewards will be earned. The only thing any layperson needs to know about this part is that this block reward will come in the form of **Beka,** and Beka will be worth whatever its spot price is on your standard crypto exchange. If that's good enough for you, skip the next two paragraphs. Otherwise, recognize that the complexity of this system does not need to be understood by the lay-person in order for them to benefit from it.

The entire supply of **Anticoin** will be created at inception, and distributed to all the donors who supply funds to make this project happen. For legal reasons, it may be important that we emphasize that this is not a "sale" of tokens, or an ICO. Furthermore, no specific amount of Anticoin should be promised for any donation amount. Let it suffice to say that the amount of Anticoin a donor receives will be only an expression of gratitude. Anticoin will simply be a thank-you gift from a non-profit charity making no guarantees or representations about any future value of Anticoin whatsoever.

That said, when Anticoin and Beka are deposited into a wallet, in equal parts, they will annihilate each other and produce **Gamma.** Gamma will be the currency by which a participant can bid on blocks of newly-minted Notes. Thereby, the value of Notes will be de-coupled from the rate at which block rewards are awarded. Also, the total supply of Anticoin will be decreasing as it is annihilated, causing its value to increase. It's important to note that only a portion of newly-minted notes will be put into circulation this way. The initial bankroll used to cause backing of Notes should grow and not be depleted by this process. The continuous dividends earned by that bankroll, or some portion thereof, will perpetually provide the value backing the particular Notes that will be distributed through this auction. Also, the auction will be a function built into the whole system, thus decentralized and not administered by any particular entity. What's more, this bankroll is not solely responsible for creating the backing of all Notes. How could it if we aspire to the volume necessary for money? Also, the entire system would be sitting on one centralized foundational pillar. The entity controlling the bankroll must be just one of many entities capable of adding value into the system.

Namespace

One reason Zooko's Triangle has been an issue for early blockchains is the ability to send to addresses that did not previously exist on the blockchain. That is, you can print a paper wallet from a quarantined computer and then send Bitcoin to that address. In contrast, our system requires that an entity be established on the blockchain before it can receive a payment. This gives us the opportunity to attach a handle. It's important to note that our system for establishing an entity does not require hot private keys and still allows private keys to be quarantined, from creation, in the same way just described. On a side note, care has been taken here not to use certain terminology, as the collection of non-profits is not a "bank," but separate contractors employed by We the People to server *our* system of accounting.

Transaction-Oriented

A transaction-oriented system is akin to the notion of object-oriented programming. Here, different types of transactions are different classes of objects. All functions of the blockchain happen in

response to the verification of a transaction as it is included in a block and validated. What matters here is the mechanism by which changes happen, and the matter of what belongs on the chain and what should be an accessory function/service that is left to the private sector to provide. For example, the issue of double spending could be tackled by the system attempting to send an alert to a participant that it found another contradictory transaction. However, this is beyond the scope of what can be accomplished in the verification of a single, isolated, transaction, according to the current state of the blockchain. We'll leave it to the private sector to collect batches and send alerts to their customers. In fact, the system will not reject double spends across batches of the same time grouping. Certainly, a node will be responsible for making sure that double spending does not occur within its own batch, but it cannot change its batch after signing it and finding another node's batch with the double-spend. So, the consensus process will ignore this and choose which transaction to apply to the balance sheet after the block is agreed upon.

With a transaction-oriented system, all actions happen in response to a transaction. Take the 50/50 raffle for example, rather than have some kind of daemon running that is looking for a winner, the transaction will define itself as a winner. Furthermore, rather than having the system issue the reward (somehow), the reward is claimed by the winner through a transaction with a single identifier at a separate time. It is with this mindset that we approach the structuring of all functions in a transaction-oriented system.

Superimposed Ledgers

Superimposed ledgers occur when transactions that map to different balance sheets are all mixed together in one blockchain. Conceptually, at least, this is different from a "sidechain" in that there is not a separate chain that anchors to a main chain. There is not a separate chain at all. Furthermore, nothing can move from one balance sheet to another. This will further our notion of **transaction-oriented** systems. With superimposed ledgers, a transaction, as an object, *must* have an attribute set that maps it to the correct balance sheet. Also, the advent of a token is equivalent to the creation of a new balance sheet. Though different balance sheets are mutually exclusive, they can be thought of as isolated circuits capable of induction through a mechanism that acts as a transformer. The backing of Notes can happen in such a way.

Backing

Gift Notes (♪Notes) will be backed by the overcollateralization of gift card value that is issued, by a business, over a ledger that is native to the system and created by that business. As mentioned earlier, there will be a standard form for creating a superimposed ledger on the blockchain, for a fee. A business will only need a commercial wallet client and enough Notes to pay the fee. Creating such a ledger can be as simple as choosing a name for the ledger and how many tokens to issue. In other words, businesses will be able to easily use the infrastructure as the backbone for their own gift-card systems. Additionally, the mechanism for over-collateralizing the value of those gift cards into the backing of Notes will be built into the system. The ability to do so will be democratically controlled by the nodes in a transaction-oriented manner.

For example, a business with an existing ledger will create and broadcast a transaction. The transaction will include a self-signed certificate that includes a URL that resolves to a self-published document outlining the details of the offer that the business is making and binding the business to the

terms, if the transaction is approved. The details will include the collateralization rate, total collateral, and portion of the newly-minted Notes that will be given to the auction. With sufficient signatures by all nodes, the transaction will be approved. In the case where the bankroll is being used to purchase the value of the gift cards, that up-front money shall be used to leverage additional value, and a good portion of that additional value shall be distributed through the auction system. This is the mechanism by which that bankroll will grow over time and continually increase the value of Anticoin, Beka, and Gamma. Ultimately, decentralizing the backing of Notes with the basket of goods that is the aggregate of all these businesses may be the only way to actually source the sheer magnitude of backing required for the currency of an entire nation.

Obviously, there's the matter of what to do if a business goes bankrupt. The backing deficit, in Notes, will have to be taken out of circulation. This can be accomplished by toggling on, by transaction, an additional transaction fee that simply burns the Notes. Not to settle on proportions right now, but this could reduce or replace the amount of transaction fees going to the nodes, and/or increase the transaction fees and only burn some portion thereof. At scale, that deficit probably wouldn't have any impact on the value of Notes, but we can limit our reliance on that fact to bridging the time span till we have burned the deficit.

It's important to note that the ability of a business to back Notes, in a way that increases their own cash-flow and market share, also creates the proper landscape for the same businesses to sell these instruments as any other product is sold within their stores. Certainly, the ability for customers to buy gift cards that have more likelihood of being circulated than redeemed is favorable to a business as they receive pre-payment for goods that they might never have to procure. Also, a mechanism is created by which tax-deductible donations to charity can be made at-cost.

Decentralized Identification (DID's)

First, let's establish the paradigm we'll be working from. This paradigm is rooted in a pragmatic recognition of how identity, money, anonymity, the law, lending, and decentralization are connected to each other. Specifically, governments trying to fight crime and advocates of anonymity are at odds with each other. The point is not to choose sides here, but, simply, to recognize the incompatibility between the two, and to recognize that the result of this is the KYC and AML regulations that have dampened innovation and decentralization within the industry. Whether we like it or not, it's a simple reality that governments will continue to fight crime by fighting anonymity. Unfortunately, the baby is being thrown out with the bathwater.

The Baby	The Bathwater
Sovereignty Democracy Decentralization Security Innovation Inclusivity Accessibility Privacy	Anonymity

You'll notice that privacy and anonymity are clearly separated from each other here. This is central to the paradigm, and how we might thread the needle of maintaining privacy without allowing a person to anonymously commit a crime.

The current state of regulations are off-target and counterproductive. AML and KYC regulations are practically useless for what they are *supposed* to be for when only the main gateways of the crypto world are regulated, backdoors abound, and, once within that sphere, money can change form and move completely untraceable. However, they do a fantastic job of undermining the ideals we are trying to proliferate, and concentrating power and profitability to the few who already have that.

Another glaring, and obvious, incompatibility of anonymity is with lending. In order for a coin to become money, lending must be available in that denomination. Financing is not given to the anonymous. Therefore, any coin that holds its value in anonymity has no chance of becoming money. Furthermore, for any coin to have a chance of becoming money, it must have a reputable, reliable, and efficient system for lenders to identify borrowers. Furthermore, debt collection needs to be supported by the law. That is, there needs to be a way of documenting that a particular public/private key is that of a particular person and that particular person signed the promissory note for the financing. That documentation needs to be available in a form that can be given to a judge or collection agency etc...

Fortunately, the connection between identity and financing is a two-way street that we can leverage for establishing identity on the blockchain. Simply put, if a well-known lender has extended credit/financing to an entity on the blockchain, then we can assume (with a small amount of confidence) that the identity of said entity is known by that lender. However, we can assume with much greater confidence that the identity of said entity is known, if *many* well-known lenders have all offered financing *and* signed a DID certificate. This is even more true with a verifiable history that said entity has made timely payments on the borrowed money. Likewise, if you, as a participant, have peers who have extended credit/financing to you (as a sort of personal micro-loan), then others can be (somewhat) confident that you are the person that they have identified you to be. Some combination of all of the above should make for a high level of confidence in ID specificity. This is the basis for **Stake-in-Identity.** The authenticity of your identity on the blockchain will be established by the fact that others have a stake in the accuracy thereof.

KYC requirements make buyers and sellers identifiable by authorities, but those identities aren't published, or even known by authorities unless the need be. That is, customers aren't anonymous, but they have some kind of privacy. One problem with this scenario is that it centralizes guardianship of these identities, and you, as a participant, have very few choices for who will keep your identity private. Instead, let's decentralize the guardianship of identity to a plurality of known entities. As a balancing measure, we'll also require that all public keys are connected to a known entity by just one degree of separation. To get biblical, for example, your brother could have a well-established digital identity on the blockchain that qualifies him as a CA. This would allow him to sign your certificate, which would establish your existence on the blockchain. The result is that only your brother knows that it's you behind that public key, but, also, if you commit a crime, your brother will be contacted by the authorities. Of course, this is all a lot of hoops to jump through for the minority of people who don't just pay lip-service to privacy, and actually do not use debit/credit cards, Facebook, Instagram, phone apps.... (i.e. the myriad of ways we've surrendered privacy in the digital world). For the rest of us, our privacy will be as

good as the **privacy policies** of the various lenders and CA's we use, and everyone that we reveal our identity to.

It's clear that this creates a spectrum of confidence levels in the identity of an entity. As a working title, let's call it an **ID score**. It's important to note that it's not the place of the developer to decide what threshold is good enough for what purpose here. The task is strictly to develop a platform that makes these tools available and easy for anyone to make those decisions for themselves. Ideally, local governments, that currently regulate buying and selling as money-transfer businesses, would define an acceptable threshold for creating an entity on the blockchain, such that Notes would earn an exception to the regulations and be legitimized at the same time. Arguably, a system that accomplishes what a law attempts to accomplish, better than the law does, should not be significantly undermined by that law. Furthermore, given an opportunity to do a better job, local government should step up to fill their role, which also includes acting as CA's. In the meantime, we will have to make an exception and establish some basic, provisional, thresholds in order to maintain the fundamental integrity of the system. For example, the same charities controlling the nodes can act as a decentralized network of CA's who's attestation is required to be qualified to create another entity on the blockchain. The businesses using the system for gift cards could also be utilized in this way.

Decentralized Money Market

Certain simple contracts will be built-in and readily available for use. In a single transaction, any entity can become a lender by posting a transaction that specifies the financing terms (interest rate, term, pay period, expiry, etc...) and the entity to whom they are offering that financing. You can imagine a window within your wallet app that lists all of the financing offers that have been extended to you, and the ability to sort them according to whatever fields are important to you. To accept that financing, a borrower posts a transaction that transfers the money to themselves, and references the financing offer as validation. With this tool, a couple of things have been accomplished. First, your friend, brother, mother, associate etc.. can bolster your ID score by staking some small amount of money in your identity. With the understanding between the two parties that this is the intent, that offer would remain unclaimed long-term, and that duration would further bolster the weight of that small stake. Secondly, the basic mechanism for a larger money market is created.

Many entities would make financing offers to one (or more) of many more-centralized entities that act as middlepersons who aggregate the money made available to them and use that money to make higher-interest loans. Let's call the entities from where the money originates "micro-lenders" and those middlepersons "merchants." With this structure, micro-lenders only need to vet and verify the merchants they offer money to, and the risk of lending to individuals is pooled among the micro-lenders. The merchants will then be responsible for verifying the identity of borrowers and handling collections. This also creates a landscape of fewer well-known CA's for Stake-in-Identity than a scattered, fragmented, and disconnected one, but also keeps the market open and competitive. This also puts a lot of power back in the hands of the people, as the merchant can also be a DAO of the micro-lenders.

The current interest rates available on a money market account, through a traditional bank, are a long distance from the interest rates that the same banks charge on credit-card debt and loans. This has, largely, to do with the fact that banks also have the option to borrow money from the Federal Reserve at relatively low rates, and that money can be made available by creating it out of thin air. Furthermore, it's

a long distance between the origin of that money and the borrower. Closing that gap, removing some middlepersons, and creating competition between micro-lenders should benefit borrowers and create more demand for Notes.

IOU's and Credit Card Companies

We can create a system that also includes the equivalent of credit-cards. That is, we want to have financing available by-purchase and the balance of debt according to only what was spent. Moreover, credit card companies act as arbiter and escrow agent between customer and retailer. When you make a purchase with your credit card, you essentially give an IOU to the creditor and the creditor pays the retailer at a later time; provided that the purchase was not fraudulent and you do not dispute the purchase. Despite these unfavorable conditions for the retailer, they still pay a fee on each transaction, to the creditor/network, in order to make this all available to you (the customer). Likewise, one type of our superimposed ledgers will be an IOU system where the borrower writes an IOU to the money merchant naming the retailer as the intended recipient of funds. This allows the purchase to happen on-demand, and the money merchant can source the funds from the micro-lenders afterwards. Naturally, there has to be an established relationship between the money merchant and the retailer. The standard format being provided would make it easy for retailers to establish these relationships with many creditors, but standards and coalitions between money merchants would likely be formed to reduce the number of individual relationships the retailers must have. Either way, infrastructure is opened up to more than just the major service companies (who currently monopolize that industry).

There's a loose end to tie up here, that is when a person needs to make a payment on debt but has lost/exposed their private key. For this, another kind of transaction will be available that allows any entity to make a payment on behalf of any other entity. There will simply be an "on_behalf_of" field, and such payment will keep the debt in good standing. On that note, all wallet clients should have the functionality to shard and distribute private keys.

DID's in Practice

Crypto payments at a busy, brick-and-mortar, big-box, retailer are the holy grail. This should be easier to realize when those same businesses are invested in Notes and benefit from their use. However, there are some practical considerations that must be addressed before it will be possible. First is the matter of fraudulent payments. That is, the retailer needs to be very confident that the checker can, and will, verify the identity of the customer paying with credit. Secondly, the system needs to be accessible to those who do not have their own data connection. Equivalently, customers should be able to pay with a portable cold-storage hardware wallet, in the case they want to separate that functionality from their phone. Let's go through all the steps taking place at the check-stand to look at the probable way of dealing with all this.

Posted on the check-stand is a digital display which, after the clerk is done scanning your items, displays a QR code containing the balance owed, the name-address to send to, and the preferred CA's that the retailer accepts. You scan the QR code and approve the transaction. At this point, your device does not broadcast the transaction. Instead, a transaction will be like a digital check. You will transmit the signed transaction back to the retailer, along with a unique URL resolving to a certificate signed by a CA that the retailer accepts. For credit purposes, the CA would be the creditor and the cert would reside

on the creditor's server. Your signature of this URL also resides on the blockchain, attached to your name-address. With this, it is possible for the certificate to, actually, be just a high-definition photograph of yourself. So, upon receiving the signed transaction and certificate info, the clerk's computer looks up your name-address on the blockchain, verifies your authorization to write an IOU to that creditor, retrieves your signature of the URL as attached to your name-address, verifies that this signature is properly aged, verifies that this is the signature of the URL you provided, downloads the photograph from that location, and verifies the CA's signature of that file. At this point, a second display, facing the clerk, displays the high-definition photograph of you, and the clerk verifies that you are the person in the photograph. With that, the clerk approves the transaction, you go on your way, the transaction is posted by the retailer to the node it chooses (or the node you specify), and the clerk never has to see your name or DOB.

It would be a matter of respect that the retailer transmits to the node that you have specified. This would only be enforced by customer loyalty. Also, the screen displaying your photo could be in front of a security professional in another room or location, and there could be a camera pointed at you, instead of the clerk doing that verification. This also provides the opportunity for this verification to be done algorithmically by facial recognition. This system also provides the opportunity for retailers to accept digital payments when the internet is down, as they may store a copy of the blockchain, privately, for verifying transactions and then post those transactions to a node once they were back online. It would be as risky as accepting a check, but they would have the option. Also, privately storing a copy of the blockchain would expedite their ability to retrieve data from it. As for your DID, attached to your name-address will be an array of signatures representing a multitude of certificates for you to choose from. These certificates will vary in the amount of personal data that they contain; anywhere from having just a photo to everything including your SSN. Clearly, this is the way that you will be able to *selectively* reveal parts of your ID. Checking that a signature is properly aged is simply a matter of making sure that someone didn't fraudulently just insert a new signature that the proper owner has not had a chance to deal with yet.

Scaling

It was stated earlier that superimposed ledgers exist on a single blockchain. The point was to define superimposed ledgers, not to say that we cannot *also* have multiple chains interacting with each other for the sake of scaling. Since many ledgers will map to mutually-exclusive balance sheets there's no reason we can't put them on separate chains (that's already the fragmented state of the industry). In fact, putting different ledgers on different chains will allow us to distribute the load to additional sets of nodes. Note that this is slightly different than sharding, as there does not need to be communication between the two. However, with our transaction-oriented approach, for interoperability between chains, a node needs only the ability to read the state of the other chain in order to validate a transaction. For example, we can have Notes existing on one chain by itself, while Anticoin, Beka, and Gamma exist on another chain. Given a participant who has won a block reward with a transaction in ♪Notes, they will claim that reward by posting a transaction to the Beka ledger. In order for that transaction to be validated, the node running the Beka chain only needs to be able to read the Notes ledger.

We will use a sort of buddy system to coordinate between chains. Here, the difference between one organization running one node that handles the entire load, one organization running two nodes (each

with a portion of the entire load), and two organizations working together (each running one of the nodes), is negligible. That is, two different nodes, each running a different chain, who implicitly trust each other, can be treated as the equivalent of a single entity. As stated earlier, the means of distributing signatory rights of a node will be done in a decentralized manner, by the nodes themselves. In fact, they will choose their partner organizations to run nodes on separate chains and form this kind of trusted partnership. As new ledgers are created by participants, for all kinds of uses and reasons, the scaling of the system can happen laterally in this way. So, ultimately, there will be as many chains as needed to handle all of the load.

Established Accounting Practices

If you Google "blockchain balance sheet," the first several pages of results are almost exclusively about how financially-successful various blockchain companies have been. This is telling of the state of the industry. Early distributed ledgers are decentralized accounting systems, but they seem to ignore the benefit of time-tested accounting practices. There are obvious reasons for this, but there are also good reasons to incorporate what we know from other disciplines. That said, a **balance sheet** affords us the ability to trim our **ledger** down to a manageable size, have a **closing** date/time/block, streamline our consensus method by finding and correcting errors that may occur while mapping transactions to the balance sheet, and generally makes our decentralized accounting system compatible with the existing industry. Keep in mind that our method for claiming block rewards requires that our transaction history goes back far enough to verify a claim.

The ledger system finds consensus on the individual transactions, not the resulting state of the balance sheet. Any error that might occur in applying new transactions might go unchecked and slow down the consensus algorithm. With a second consensus layer, we can "close the books" on last year, last quarter, or the last hour. This will be a new kind of block chain. In this block chain each block will strictly be the consensus of the balance sheet of the respective ledger blockchain. Since we can choose to close the books at long intervals of time, and the data passed around will only be a hash of the balance sheet, verifying consensus of the balance sheet should not add a significant load, and can be done by the same nodes handling the ledger. Optionally, we could employ redundant balance-sheet-only nodes to vastly increase the level of decentralization in our consensus method without slowing down how we process individual transactions. For now, we'll assume that's not necessary.

Theoretically, all nodes should map the same ledger to the same balance sheet. So, consensus on the balance sheet should be quick and easy. Given that the node receives proof of consensus on the balance sheet, as of a particular closing block, certain transaction data from before that block can be dropped from memory and no longer needs to be stored. So, the storage of the entire transaction history will be left to the private sector, and individuals will be responsible for storing their own transaction history while it's available, if they don't want to have to buy it from the private sector. That is, if they are concerned with having it at all.

On a side note, the perspective from which we are applying our accounting practices is the collective "our." One should hesitate to call it the accounting of the DAO, as that infers that participants are customers of a more centralized organization, rather than the fact that those non-profit entities are individually contracted by We The People to provide a particular piece of support to *our* collective

accounting system. We must take care that our **debits** and **credits** (left and right) align with this perspective.

Finally, with regard to accounting practices, wallet clients should include the functionality to tag every transaction with an account label/type, store all transactions with these tags, and export this data according to a standard that will make it easy to import into accounting software (or generate financial reports itself). The advent of decentralized accounting systems should help participants keep track of and make informed decisions about their finances.

Conclusion

Let's take the mass and momentum of the financial sector, and their laws, and redirect that energy into a system that makes the world a better place.