Cicada: A Distributed Direct Democracy and Decentralized Application Platform

@laugh1ng.m0nk3y - @jade.rabb1t.23 - @ox.head.826

cicada [AT] iamcicada.com

Version 2016.28.11.GA.003

Table of Contents

Abstract

This paper outlines a revolutionary, open source, decentralized application platform (DAPP) along with its first major application, a distributed direct democracy (DDD). To create something scalable enough to run an entire nation with no representatives, we created two cutting edge technologies to serve as the foundation of the platform:

1] A decentralized, privacy guaranteeing Human Unique Identifier (HUID) unique to every person on the planet that returns control of personally identifying information (PII) back to individuals, allowing for revocable role based access control to that data.

2] A new Distributed Proof of Work (DPoW) backed blockchain that is immune to centralization because of a unified client/miner with only one miner allowed per person, linked anonymously to a HUID. Miners are randomly drafted into built-in pools, meaning everyone contributes yet nobody dominates. The system provides a workable UNIVERSAL BASIC INCOME (UBI), i.e., it pays users to participate, since everyone is drafted to secure the network. It is incredibly energy efficient because it does not mine all the time and it is also storage efficient because it uses a Distributed Hash Table (DHT) of historical blockchain transactions, allowing it to fit and run on a cell phone.

Motivation

A true Direct Democracy (DD) requires representativeless government. The concept is so new that the word representativeless does not even exist. We had to invent it. Think of a digital DD as true algorithmic government. It holds the potential to unleash the true will of the people, while being a natural evolution from current governmental systems. While Direct Democracies have been tried in

the past, in limited scope, such as in Athens and pre-republic Rome, there was simply no practical way to make such a system work with pure analog technology. A Direct Democracy absolutely requires mass communication and digitization and it almost certainly requires Artificial Intelligence.

The inspiration for the idea was two-fold. The first was author Daniel Jeffries' work on *The Jasmine Wars*, an epic sci-fi and military saga in which China becomes the world's first direct democracy, run by a highly advanced, artificially intelligent version of a DAPP platform called Cicada. This led Jeffries to wonder what exactly it would take to create such a platform, and whether the technology to do so was even feasible. When he conceived of the idea twenty years ago, the answer was decidedly "No," but society is now at the stage where it can at least begin to create this technology, even if some of the necessary pieces are still evolving.

The second inspiration came from the Arab Spring. Beginning in Tunisia in Dec 2010, a revolutionary wave swept the Arab world as the people rose up and finally managed to throw off generations of brutal and oppressive regimes. After the heady and hopeful early days of these revolutions, the results were dismally predictable: Tunisia remains unstable and, at the time of writing, unable to adopt a constitution. The Egyptians have replaced one military dictatorship with another. Syria has devolved into the brutal and horrifying civil war that gave rise to the vicious and fanatical ISIS, while creating a refugee crisis that threatens to rip the civilized world apart.

In other words, people are worse off than when they started.

This result is sadly the norm throughout history. Even when people manage to peacefully overthrow decades of dictatorship, with very, very few exceptions, they wind up with yet another one. Communist China is another perfect example. The Chinese people rose up and replaced five thousand years of dynastic rule with a dynasty by yet another name: the People's Republic of China, a communist "people's dictatorship" that is essentially just a minor evolution of their dynastic system.

This problem of replacing one bad government with another stems from various organizational and systemic issues. After the Egyptian revolution, the only people sufficiently prepared and organized enough to gather votes were the Muslim Brotherhood, who had been organizing underground for years. More liberal and secular minded parties were left scrambling to make their voice heard. This led to the Muslim Brotherhood getting elected with a majority despite their relative unpopularity. Once in power, they sought to create a constitution bent to the will of religion. This was no surprise, since their primary worldview is that religion should be supreme in governing people's lives. People's worldviews directly shape and limit their ability to create solutions. Unfortunately, since the Brotherhood's view was not shared by everyone in the system, and they failed to account for and give voice to alternative viewpoints, which is the very foundation of a democracy, their swift collapse was inevitable. Within a year they were overthrown by the only other organized group: the military. So in just three years Egypt wound up right back where it started, trading one military junta with 40 years of "emergency executive powers" for another military junta masquerading as a democracy.

A properly developed Distributed Direct Democracy that runs from everyone's phones and/or the personal IoT electronics of tomorrow has the potential to break this cycle forever.

In its idealized state, a Direct Democracy is self organizing, self-booting, self-replicating system that provides, instantiates and automates all functions of government. It is able to sustain itself indefinitely, provided sufficient communications infrastructure still exists within the country or sufficient client devices still exist to power a mesh network.

Creating such a platform may take ten years. It will require cutting edge programming, critical thinking, and a willingness to solve a string of currently unsolved problems. But the potential rewards are vast and manifold, easily making it worth the huge engineering effort required to make it a reality.

If there is one goal for this project, above all others, it is this:

The next time there is an Arab Spring, the people will be able to replace their leaders with code.

Defining Distributed Direct Democracy

Before we dive into the technical details, let's discuss what a Distributed Direct Democracy is and why it is even needed?

To start with, a DD is not another digital democracy initiative that focuses on augmenting or streamlining current representative systems of government. A DD is a representativeless government, where every idea is voted on by every citizen.

This, of course, creates a number of new challenges, such as driving voter participation, avoiding voter fatigue, and dealing with the average voter's imperfect understanding of essential issues. There is also the problem of preventing the system from being overwhelmed by irrelevant, impractical, or disingenuous proposals, an issue which plagues existing frameworks such as California's ballot primary system or the UK government's petition website.

A later section addresses many of these issues via automated voting, gamification of voting, and the ability to call expert private groups to create fast tracked proposals for complex topics such as economics, technology, and the sciences, rather than relying on politicians with no experience or meaningful understanding these disciplines. We will also provide a filtration system that dramatically limits the amount of proposals presented to the entire nation for voting, ensuring only the cream rises to the top.

In short, we propose creating:

- Algorithmic government

- Representativeless government

We are also focused on building something that is:

- Entirely decentralized and serverless, with no central choke points

Existing projects and initiatives, such as DemocracyOS, Liquid Feedback, and BitCongress, while interesting, innovative, and admittedly further along than this project in some respects, are primarily about providing an easy way to talk to elected representatives.

The problem is that those representatives are under no obligation whatsoever to care about that feedback.

And we already have a robust platform for debate and learning about issues. It's called the Internet.

Representatives are not interested in constant feedback from those they represent. They are interested in their own views and agendas, which is why these platforms have seen very limited adoption and likely will remain that way.

A representative system is flawed because representatives don't really have to care what people think. They are elected as proxies, not as a direct reflection of the people's ideas and opinions. They are there to govern for us. If their views begin to widely diverge from popular opinion, there is no patch for that flaw in the system. The people are out of luck, doomed to watch as their "representatives" begin to implement policies that work against them, with no way to slow or stop the slide. While the United States still remains the standard bearer for democratic success in world history, the policy positions of elective representatives significantly diverge from those of the average "man on the street." The now famous Princeton Study, which looked at 40 years of lawmaking, found that if a policy had a 100% approval rating from the people it had only a 30% chance of passing. Conversely if a policy had 0% support it STILL had a 30% chance of passing. These divergences are intrinsic to a representative system. Thus any system that looks to provide feedback will not change much at all.

By contrast, a Direct Democracy creates a powerful framework for continuous voting that reflects the changing will of the people, exactly as they intend. While there are certainly challenges and weaknesses to solve for in this type of system, we believe the system can be crafted in such a way that it will be strongly resistant to distortions while providing powerful checks and balances at an algorithmic level, making it less prone to subversion when elected representatives decide to erode or ignore constitutional and legal checks and balances on their power for personal gain.

Technology Introduction

Making such an audacious idea a reality requires a fundamental rethinking of DAPP platforms. While it is possible that Bitcoin and Ethereum, the two major contenders to the DAPP throne, will one day be ready to scale the heights needed to securely run an entire nation, that day is not now. Bitcoin is almost exclusively focused on the limited use case of creating a cryptocurrency and has major scalablity issues that the debate over 2MB or 8MB blocks barely addresses. If the Lighting Network project creators' calculations are correct, then 8MB doesn't even come close to solving the problem. To grow to a planet scale, Bitcoin will require a radical re-architecture or the successful implementation of an off-blockchain payment system, where the blockchain merely functions as a dispute mitigator. To put the problem in perspective: if 7 billion people perform 2 transactions a day, that alone gives us 24 GB blocks, generating a blockchain of 3.5 TB a day, or 1.27 PB per year.

Ethereum has different challenges. It is a truly ambitious project, run by a brilliant and visionary founder in Vitalik Buterin, and it may yet prove successful and revolutionary in the long run. But today it is still in its infancy. Recently Ethereum suffered a major breach because of a security vulnerability in its smart contracts, allowing a hacker to steal $60 million in Ether from a Decentralized Autonomous Organization (DAO), forcing a hard fork in the blockchain to restore the stolen currency. This precipitated a kind of civil war inside the platform, with Ethereum Classic maintaining the original blockchain and the official project working off a forked blockchain, a civil war that is still raging to this day.

In truth, Ethereum is really an extension and evolutionary iteration of the ideas of Bitcoin. It was built to enable some technology that Bitcoin refused to support, such as a Turing-complete scripting language. However, it maintains many of the limitations of Bitcoin, such as its proof of work (POW), which is prone to heavy centralization, and its reliance on a wildly fluctuating currency.

We need a new way of thinking about DAPPs to create the platform that will run the next generation of decentralized, autonomous applications and give us the Internet we deserved in the first place, instead of the increasingly brittle, insecure, and centrally controlled system we have today.

There are several crucial technologies necessary to build this new system, each of which will be explored in turn.

1] A HUID generated through the intersection of revocable biometrics and cryptography. Revocable biometrics improves biometrics by allowing systems to revoke a biometric token without revoking the underlying biometric. This is further improved with the addition of cryptography. By using biometric markers (initially utilizing both irises as inputs) to create public/private keypairs, we can generate a unique ID for each person on the planet. This ID is not centrally controlled, which will virtually eliminate all Sybil attacks, along with the vast majority of problems in peer to peer networks. The

HUID also allows for blind signature/zero-knowledge proof sub-IDs that allow role-based access control to personally identifiable information (PII).

2] A peer-to-peer network based on a modified version of the Kademlia protocol. By using a special, single-use sub-ID of the HUID to generate the Node IDs in the system, we can prevent Sybil attacks and allow for a new PoW system. This special purpose sub-ID has no access to even the basic ID numbers of the HUID, and thus provides unlinkablilty and unobservability. This in turn provides the kind of capabilities we are already used to in our democracy, such as anonymous voting and the principle of one person, one vote.

3] By linking the HUID to the Node ID, we can then move on to create a powerful new Distributed Proof of Work (DPoW) for securing our blockchain. The key is to eliminate the split between client and miner that currently exists in all cryptocoin systems. Instead, we unify both client and miner, meaning every client is a miner and vice versa. Each person is allowed exactly one miner. To secure the system, miners are randomly drafted into built-in mining pools to compete against each other. This has the happy benefit of providing a workable and practical Universal Basic Income (UBI), since everyone is drafted to secure the network. However, since only a fraction of the network is required at any time to secure the network, it remains energy efficient enough to run on a cell phone, since expensive and battery-draining operations are not running constantly. In addition, this system eradicates the current plague of centralized mining in Bitcoin and other cryptocurrencies, where private companies build ASICs secretly and never release them to the public, trading one form of centralization (government-sanctioned banks) for another, more pernicious one (private, unaccountable, and ultra-secret corporations/organizations).

Lastly, we make the platform storage efficient by storing only the most recent transactions in the program, likely only the past few months' worth, and then shard up the historical blockchain via a Distributed Hash Table (DHT), which is then redundantly distributed to all miner/client participants so that it can fit on any tiny device.

We will also outline other technologies specific to the functioning of the DDD and the underlying DAPP platform. But before diving into each of these fundamental building blocks in detail, let's take a quick look at the design philosophy behind those ideas to understand why they matter and why they are necessary.

Philosophy of the Project

Our philosophy is simple:

There is always a solution.

Just because a problem is challenging or people have not previously discovered a solution, does not mean no such solution exists. It means the problem is not being looked at correctly.

In order to create solutions for as-yet-unsolved problems, we do the following:

- First, we identify ideal characteristics for the system.
- Next, we develop a solution that meets ***all*** of these criteria. Solutions with known flaws are not acceptable unless those flaws are mitigated completely or extensively minimized.
- We then create these solutions by taking an interdisciplinary approach, uniting ideas from multiple areas of human knowledge, taking inspiration from seemingly disconnected fields. But no field is truly disconnected. Separateness is an illusion. All things in the known universe share similar patterns, and when studied together, offer insight into difficult or seemingly impossible problems.

Many people and teams are prone to saying "there is no solution to that problem" and then defaulting to known, flawed solutions. This is nothing but lazy thinking and unacceptable in the Cicada project. An example is using "trusted" central authorities, a system with extensive flaws and known weaknesses. Reliance on central trust doomed attempts to create digital currencies, such as e-gold, before Bitcoin solved the problem of decentralized trust and consensus.

Where possible, **we favor the following characteristics for our system**, with any deviations needing to offer significant benefits:

- No single organization, business, or person can control or alter the system without extensive consensus
- Completely decentralized and distributed
    - Reliant on no centralized trusted entity for its core functions
        - However, it is worth noting that pluggable modules, outside of the basic infrastructure plumbing of the system, may be open to relying on the services of centralized entities for highly specific functions, such as current web APIs
- Privacy and the right of a person to be in control of their personally identifying information
- Openness
    - In methodology, source code, protocols and security measures, upgradeability
- Security of the system based on emerging "trusted computing" practices
    - Though the term "trusted computing" is controversial because of its use by corporations to limit what users can do with their personally owned systems, in this case the power of the concept will be inverted and used to return the power to the individual. Its ideas will be leveraged to protect the user and the system from malware and centralized attempts to subvert the platform.
- Strongly resistant to coercion and subversion

- Usability/Simplicity
    - Easily adoptable by a broad group of people with no technical skill to advanced technical skills.
- As much of the technology we propose relies on encryption, we stand absolutely against key escrow or weakening of cryptography in any form. Ultimately, this is the only true threat to the system
- End-to-end verifiable and auditable
- Resistant to hostile actors

If a proposed solution fails to solve for any of these principles, it is inadequate and to be discarded swiftly.

The philosophy behind a project directly corresponds to the code that is created. Starting with an incorrect perspective leads to broken results that do not serve all the people in a system. The philosophy of this project is to build an "agnostic" or "universal" system. To serve as many people as possible it must be agnostic and immune to attempts to take control of it and force a particular worldview or agenda on it.

The history of the world is a battle of ideologies. Each of those ideologies believes that if they could just gain complete control, everyone would be happy. Nothing could be further from the truth. There is literally no way for a single worldview, be it capitalism, socialism, communism, or any other "ism," to satisfy all the people all the time. If the system is not agnostic, in that it serves the needs of everyone, it serves no one, and it will not be widely adopted. To be agnostic it must be open and unlimited in its use, fostering competition among ideas, with a system of underlying checks and balances that keep it all organized and operating fairly.

Many of these core principles will create unique challenges. Some of the principles seem at odds with each other, making it hard to solve for all of them at once. When people design a system with such challenges, they run across hard or seemingly intractable problems. This often leads them to compromise. For instance, in voting systems, strong identity management and anonymity seem to be starkly opposed. As such, the architects often decide one of these principles must fall by the wayside. This leads to flawed and broken platforms that do not meet the needs of all the actors in the system. For example, Democracy OS "simply does away with secret ballots." For this project, that stance is unacceptable. It indicates that the designers did not iterate enough through the challenges to come up with a truly revolutionary concept, instead settling for a partial solution.

Partial solutions are no solutions at all.

Many problems seem totally unsolvable. This happens when systems are required to have traits that are in seeming opposition to one another. For example, before Satoshi Nakamoto created

Bitcoin, all systems were believed to be subject to Zooko's triangle, which states that systems cannot have the following three characteristics simultaneously:

- Human meaningful
- Decentralized
- Secure

Before Bitcoin, people were able to solve for two of those properties but not all three.

Before Roger Bannister broke the four-minute mile in 1954, it was thought to be physically impossible. After he broke it, multiple people were able to break it in short order, simply because they now realized it was possible. They now had the correct frame of reference to actually achieve their goal. When we assume that something is impossible, we start with an incorrect framework for solving hard problems. We are starting at a deficit. It often takes one person to prove something is possible to enlarge the problem domain.

Unfortunately, solving a previously unsolvable problem tends to lead to a new problem that we call the "Satoshi box." We often jump out of one box, right into a slightly bigger box. People quickly adopt the solution, but are unable to think outside the framework it provides. Instead of finding additional solutions to the challenge, they iterate on the same solution over and over, even when it is not appropriate. We have seen this with almost every other cryptocoin to come after Bitcoin. With a few notable exceptions, all of them are but tiny variations on the same idea, with little to no divergence or original thinking.

This project will require both building on the past and completely new and undiscovered solutions. It requires big-picture, ambitious thinking.

Tiny iterations are not enough.

Finally, it is also the position of this project that **this technology is inevitable.** Either we create it ourselves, or someone/something will do it for us and we will not like the results. Likely any centralized creation along these lines will end up being a highly partisan, privacy destroying, repressive, insecure and broken machine that enslaves us rather than sets us free. It's as simple as that. We can't let that happen. They way to stop it is to get ahead of it now.

Already banks and foreign governments are creating huge, weakly secure, non-revocable biometric identification systems in centralized databases that we have no control over. Companies like GenKey are already working with the Indian government to create universal IDs for billions of people, using a proprietary, copyrighted algorithm that people have no insight into whatsoever.

These systems should be replaced with universally vetted, secured, and openly designed systems.

We simply can't allow this to be a private, closed system locked away from public scrutiny.

It must be open source, and it must run on a blockchain, to ensure no single entity controls it.

Why? Because, human history has demonstrated again and again that centralized trust is an oxymoron.

If we allow private companies or current governments to create this technology, you won't know where or how it is stored, what info is stored with it, whether the proprietary algorithms used to create it are any good, or even if it's really secure at all. You won't know most of this until it is inevitably hacked and all of your personal data spills out onto the nets, including your iris or fingerprint template or your DNA, which is now useless as an ID because you can't take it back or change it.

Architecture

In this section we discuss the high level architecture of the system.

First, let's list the primary components of a DDD:

Key Parts of the System

- Human Unique Identifier (HUID)
    - Controllable by the user
    - Not stored in a centralized database of any kind
    - Deeply resistant to coercion and corruption
    - Nearly impossible to duplicate or steal
- Organizational IDs
- Organizational and personal RBAC
- Decentralized DNS such as the system designed by Blockstack
- End-to-End Verifiable Voting framework
- Groups
- Routing protocol
    - Based on next-gen Kademlia distributed hash table design for P2P networks
        - Relies on the HUID to prevent Sybil attacks and uses a genetic learned trust algorithm among peers for routing and reliability ratings of nodes and peers
- P2P fully encrypted communication system
- Law framework encompassing:
    - Parsable XML based law
    - Proposal and initiative system
- Smart contracts

- Participant and Group Entity Reputation system
- Decentralized Message Bus
- Distributed code repository
- Distributed Proof of Work (DPoW)
- AI/Machine Learning/Deep Learning system (Potentially as the tech develops to work on less powerful chips)
    - An AI system would be used for
        - Auto-vote casting
        - Arbitration
        - Simplifying voting descriptions
        - Improving voting gamification
        - Budget distribution and analysis

Cryptography Technologies

- Zero-Knowledge Proofs
- Multi-signature transactions
- Blind Signatures
- Mixnets
- Public/private keys
- One way, non-invertible hash functions

Abstracts/Primitives of the System

Before discussing each of these components in turn, let's look at some metapatterns of the platform. The protocols designed for such a system would have broad applicability to a huge number of other types of systems, not just voting/government. Therefore, it is best to outline these protocols in generic, reusable terms so they can function as a complete DAPP platform.

We are creating a method to define social interactions at the planetary scale. Social interactions are links between various groups or individuals. Think of these groups as a series of spheres connected by strings. A society is nothing but a huge sphere encompassing smaller spheres, like states, counties, cities, companies, local clubs, volunteer groups, reading groups, movie lovers meetups, and sewing circles. Humans voluntarily form and drop out of spheres all the time. The interactions between those groups are nothing but a series of social protocols. A society is nothing more than a series of implicit protocols for how we organize, connect, agree, and disagree. Agreement and disagreement, connecting and severing ties are transactions in the system of human experience.

Think of this system as a human interaction modeling platform. These are what we refer to as "grouping primitives."

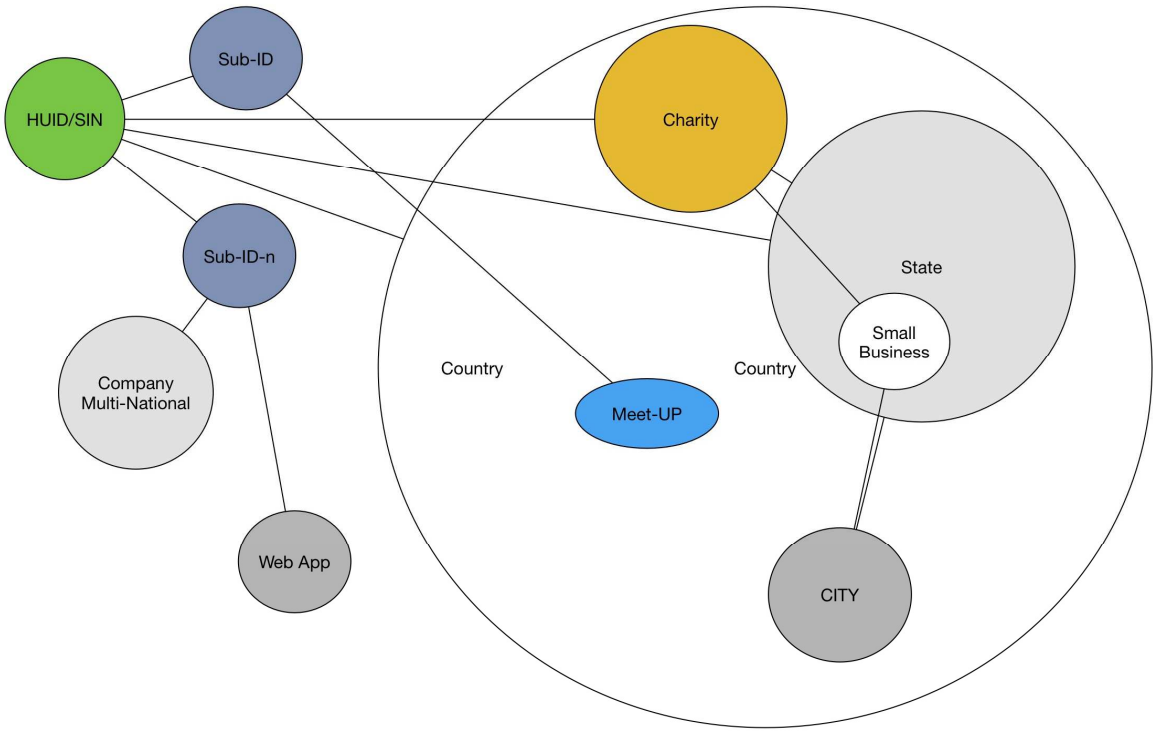
While the project starts off as a direct democracy engine, a democracy is merely a subset use case of these abstracts/primitives. It should be noted that the direct democracy engine is a particularly unique use case, and as such will have some abstracts designed specifically for its use (e.g., action tokens and unlinkability to prevent vote tracking).

The system is also completely scalable. A local charity or social club could limit their sphere of votes to members of that group. This would allow the platform to work for everything from a gamers' hub to a nation state.

The system rests on the back of an open, totally decentralized universal ID system that uniquely identifies every human on the planet, is highly resistant to compromise and coercion, and is not controlled by any single corporate, group, or government entity. And yet those corporate and government and group entities can consume the ID system to attach an individual to their sphere or organization. However, the ID might be a unique sub-ID that simply grants limited access to only a subset of the what the very human owner of that ID wants them to see.

For example, Nielsen may want to make you a member of their rating team and they might want access to a subset of data that you have stored in a PII database or Info Wallet, which is described in detail later. You might generate a sub-ID that is cryptographically linked to your primary ID, but that only gives them access to a small subset of those interests.

We are looking to model every aspect of how humans interact and form bonds. How do they exchange value, do business, put forth ideas, create laws, enforce laws, join groups, leave groups, agree, disagree?



Individual Component Outlines

Here we outline the various critical foundations of the system. They form the framework for constructing the specific use case of a distributed voting system, but they can be used to govern many, many types of human interactions, such as joining or leaving a company, creating a decentralized autonomous organization (DAO), ensuring trusted computing, and decentralizing privacy, to name but a few.

Human Unique Identifier - HUID/SID/SIN

We have already broadly discussed the HUID, but this section will dive into it in detail.

The "Human Unique Identifier" (HUID) or "Single Identifier" or "Secure Identity Number" is an ID unique to each human on the planet. This allows us to prevent Sybil attacks and ensure everyone has a voice in the system. **The HUID must be unique, incontrovertible, incorruptible, and not centrally stored or administered.**

The system's integrity rests on this ID, so it must be the most well-designed part of the system. No previous ID effort is an acceptable stand-in here. As such, consider this only the most basic framework for such a system.

The HUID uses biometric markers to create public/private key pairs, guarded by passwords, which in turn are used to create IDs and linked sub-IDs. To be very clear, what we are talking about here is **biocryptics NOT biometrics**. **Biocryptics are the intersection of biometrics and cryptography**. Biocryptics holds the promise of solving for all of the problems that biometric systems create.

Traditional, non-revocable biometrics has a long history of insecurity and compromise. In the general public they often incite fear and uncertainty, a fear and uncertainty that is justified. However, in the past five to ten years, cutting edge researchers have spent countless hours considering revocable biometrics, biometric security, and using biometrics in combination with cryptography to create strong ID systems that mitigate the flaws of earlier systems.

Biocryptics will be the focus of this system, as there is currently no better methodology to create a universal identifier than to start with the markers that make us all unique: our biology. The primary difference between biometrics and biocryptics is that biomarkers, such as fingerprints, irises, retinas, DNA, and/or spectrometer readings, are used as inputs to cryptographic key functions.

There may still be hidden flaws in the implementation of this idea that will come out as other researchers consider it. Nevertheless, even if the specific parameters outlined here are found to be flawed or incorrect in some way, the underlying idea of creating a unique ID for each person, linked to their biometric markers, remains crucial and foundational to the system, and designers should look to iterate until they have found a perfect one with no known weaknesses.

To start with, the Cicada project will focus on using double-eye iris scans as the foundation of its identity system. Iris scans appear to be the best current candidate for the HUID, because they are fast, reliable, and have a very low false positive rate. With the release of the Samsung Galaxy 7, despite its issues of blowing up, which we expect to be resolved, the technology is about to become widely available, and may eventually become a standard feature of end user devices.

Unlike retina scans, iris scans don't require you to be too close to the scanner. Lastly, because the scan can be done from a distance, it makes it easy to identify both eyes at once. This is an essential feature, as we want to keep people from making multiple IDs for themselves.

However, just because we have outlined the prototype system with iris scans does not mean this is the only possible solution or that better ones will not come along in the future. The system should be pluggable and allow for upgrades/replacements/sunsetting as the technology develops along these lines. DNA holds the strongest possibility as a perfect ID system, considering that the problem

of telling twins apart is now largely considered solved. Unfortunately, DNA testing at the time of writing still involves taking blood or saliva and sending it off to an external lab, which makes it vulnerable to numerous layers of compromise. In the future, we foresee systems, such as a watch or bracelet, that can extract imperceptible amounts of blood and synthesize that DNA without the data ever leaving your wrist. At that time the technology will be reconsidered.

The HUID is created by generating a public/private key pair from an iris scan and storing that ID in an identity blockchain, known as the HUID chain.

The HUID has the special advantage of being largely immune or at least incredibly resistant to Sybil attacks, because it is unique to each person and cannot be reused in the system. This is an essential trait for a system where one person, one vote applies.

To create the **public/private key pair,** we scan for biometric markers known as "**minutiae points**." We identify these critical minutiae points to create a "**template**" of the points, convert them into a hash, and use them as seeds for a pseudo-random number generator to create a **public/private key pair**, along with salt to create additional randomness.

This public private/key pair will then be used to create an "info wallet." This is similar to a bitcoin wallet, but used for storing personally identifiable information (PII). The public key is run through a function to create a unique numerical ID for a person.

The public/private key pair will include a built-in requirement to create a strong password, utilizing a rainbow table of prohibited weak passwords, enforced by the blockchain, to ensure good security practices. A password is necessary to prevent coercion, as a person could be forced to use their biometric markers to do something they don't want to do, such as unlock a phone. Without a password, that phone simply unlocks.

It is also possible for a user to create a "coercion password" that can be used if they are threatened or under duress. This password would appear to perform all the actions requested by the attacker but allow them to be rolled back at a later time. This creates some problems, as it could be used to scam legitimate transactions in a system, so this will not be considered in the rest of this paper. However, it should be addressed at a later design stage.

The system will also include a blockchain-driven challenge/response system for resetting passwords, re-enrolling keys, or changing keys, which marks old keys as blacklisted/unusable. People forget passwords. With no friendly web company to reset it for you, there must be a secure, powerful, decentralized method built into the protocol to make it easy for people to do this without compromising overall security.

To secure the template, yet prevent repeated use of the biometric signature, we delete random sections of the data and store it in a different biohash/fingerprint blockchain that is not linked to the

HUID chain in any way. To do this we use a "Blockchain of Blockchain" approach, as outlined later in the paper. By deleting random chunks of the template, we can use that data to check if a person is already enrolled, but not simply take that template and attempt to create a new key.

It's critical to note that a number of template security protocols for revocable biometrics have been defined in recent years, and this method may not be ideal. For the purposes of this paper it is considered at best a working design. It is also possible to consider homomorphic encryption for the template, a method where data is encrypted and never decrypted as part of the verification process. Homomorphic encryption is undergoing extensive R&D currently and may prove the best method in the future for securing biohash templates.

The design of the system also allows for the creation of various special purpose (used for one purpose only once) and non-special purpose (used for anything) sub-IDs linked to the primary ID, which will be outlined in the next section.

The protocol must (eventually, though not in the initial POC design) allow for alternative biometric markers as a backup, should a person have only one eye or none. This creates unique and difficult challenges in that it allows for various attacks by using Hollywood style makeup, which could allow a user to attempt to create two IDs. These will not be discussed in great detail, however. There are also other potential attacks on the system such as constructing fake eyes, but these may be mitigated by "liveness tests and/or anti-spoofing tests" and will require further considerations, perhaps requiring a user to show their face and wave their hands, etc., as well as various fuzzy logic and reputation-based algorithmic defenses. Since we are also building a legal framework, we may choose to empower a government organization with the power to repudiate or test biometric markers in court.

We will sidestep these limited scope attacks for now and assume there are sufficient defenses against them, since researchers have been addressing these type of techniques for years. Our concentration is on making a robust ID system that utilizes both eyes.

A private, bonded, proof of stake agency may be the key to allowing alternative biomarkers as an option in the future, which can verify that a person is missing the required biometric markers and issue a one-time code to create an alternative ID based on different biometric markers. However, other biometric markers have a number of problematic characteristics. For example, a person's fingers can be swollen, cut, damaged, sweaty, etc, all of which creates variance, which in turn creates different inputs and hence different keys. In addition, all of a person's fingerprints would need to be entered to prevent multiple IDs being created, which is cumbersome and time consuming.

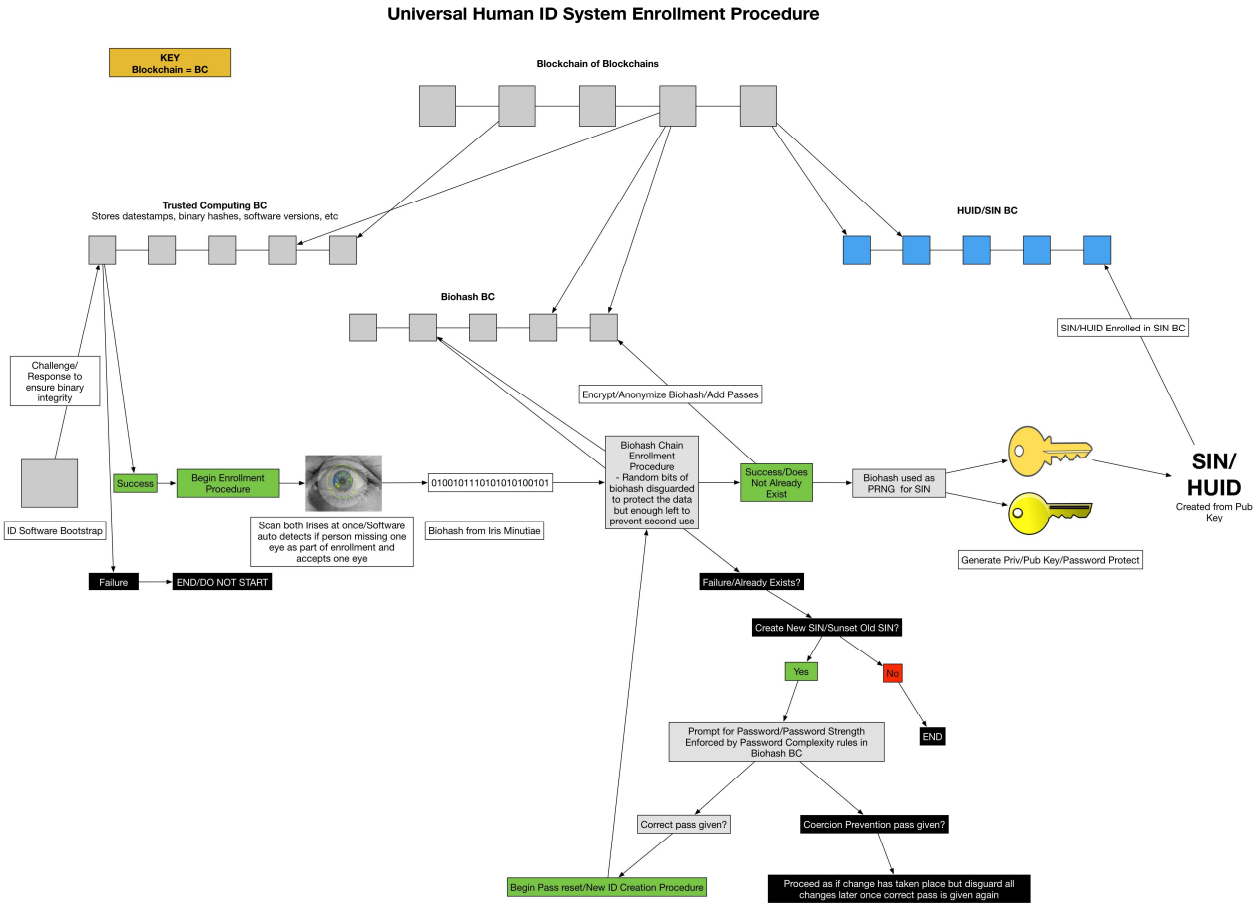
For the time being, we will simply say that the system may choose to allow for additional and/or replacement biometric markers to provide weight and "reputation" to the original ID or to replace a

non-recoverable or stolen ID by flagging a lost ID as permanently blacklisted. Candidates for secondary biomarkers include, but are not limited to:

- Voiceprint
- Retinas
- DNA
- Palm prints
- Fingerprints

Again, all of these present special problems and will not be considered further here, although they may be addressed by the project at a later time as good solutions become available through research and development.

Diagram of Proposed Iris Enrollment/Creation Procedure



Outline of Proposed Steps for Enrollment in the System

1. Enrollment software boots an encrypted microkernel that checks for live network connectivity through a series of tests, using randomly generated secret information, challenge/response, pings, time checks against the blockchain, and TLS-connected NTP servers
2. After ensuring network connectivity, the microkernel checks the software blockchain to ensure its binary hash matches, as well as the correct, current protocol version, and any other checks deemed necessary to ensure the system is running the latest, unmodified code

3. If a pair of Irises is already enrolled, the system should check the user's password with challenge/response and refuse to boot if the password is incorrect after a series of *n* failures
4. Should any of these checks fail, the system should refuse to boot
5. System boots and is ready to scan the irises
6. Irises are lined up and scanned
7. Additional anti-spoofing tests are conducted
8. The protocol enhances the raw image through filters
9. Critical unique "minutiae" points are extracted
10. The center or core point of the image and its corresponding radii are identified
11. The type, angle and position for each minutiae point with respect to this coordinate system are calculated
12. The entire iris print configuration is mapped into a string of bits known as a biohash
13. The biohash is tested against the Biohash blockchain to check whether it already exists, to prevent double enrollment
    a. If the biohash does not exist:
        i. Random bits of the biohash are deleted. Homomorphic double encryption techniques may be used to create a secure biohash iris (SBIOH-I)
        ii. The SBIOH-I is projected into the biohash blockchain
        iii. HUID creation begins
            i. The user is prompted to create a password, using a rainbow table blacklist to prevent easily guessable passes and enforce password complexity
            ii. The user is also (potentially) prompted for a secondary "coercion resistance" password, which can be given in the event of violence against the individual and which gives the appearance of having reset the ID, but which silently discards the new data
            iii. A random set of challenge questions for resetting the ID and password is also generated
        iv. Answers and a one-way, non-invertible hash of the password are stored with the biohash in the biohash blockchain
        v. The biohash is used to seed a PRNG, which in turn is used to generate a main public/private key pair (PUBID,PRIVID)
        vi. An "info wallet" is created for local storage of keys
        vii. The public ID is created from a secondary hash function against the public key, just as with Bitcoin. The PUBID is NOT the HUID.
        viii. The public key and HUID are projected into the HUID blockchain, separate from the biohash blockchain.
    b. If the biohash already exist in the biohash blockchain

. The user is asked whether they are attempting to reset or recover their biohash
    . IF NO
        i. End

i. If YES:
    . Start reset process
    i. Prompt for password
        . IF COERCION PASSWORD RECEIVED:
            1. Silently discard updates but provide identical messages and confirmations to the users to ensure their safety in the face of an attack

ii. IF RESET PASSWORD RECEIVED
    . Allow new biohash data to be projected into the biohash blockchain and start the process of generating and replacing a new public/private key
        . Do not delete old biohash data, simply add the new data to a new spot in the chain and set to flag any bit that labels previous biometric data as obsolete/defunct
        i. RETURN TO NEW ENROLLMENT PROCEDURE

Use of the HUID in the System

The HUID is immutable to the individual and cannot be changed. It is unique, even in twins. It can be augmented or potentially replaced with additional biometric markers such as voiceprints, spectrum readings, DNA hashing, iris scans, and the like, which would add additional "weight" or "reputation" to the initial ID as the ID is used over time, as a backup for proving ownership. As defined by this system it can even be resilient to changing characteristics such as age or scarring.

The HUID underpins the system and allows arbitrary capabilities to be tied to the HUID, through linking to additional tokens or capabilities, without being reliant on a central authority to secure an unsecurable number such as an SSN, which must be given to those authorities. This creates a weakness whereby any compromise of that authority is a compromise of the number itself.

The HUID allows for a 1-to-1 voting token to be assigned to a person and rights and capabilities to be assigned to an HUID, such as the rights and protections of citizenship in a particular country or the rights of an individual in a company or collective/co-operative.

In the Direct Democracy system, a voting token is taken from an anonymous pool of voting tokens and assigned temporarily each time the citizen votes. This token is then routed through a decentralized mix-net and released, preserving voter anonymity. The token is assigned on a one-to-one basis to the anonymous voting sub-ID linked to the HUID. The voting right represented by this token can be taken by a centralized law body, or automated law entity, but the ID itself remains with the individual indefinitely and cannot be taken or corrupted. The voting token is separate and associated on a 1:1 basis with the HUID.

Some research into using biometrics for cryptography allow for the deletion of the private key, which prevents it from needing to be stored by the end user and keeps it from being compromised. This method does create some challenges though in that it likely removes the essential password component and requires the person to continually scan their biometrics to use it. While this method holds some promise, unless an answer to the password problem is discovered it is likely unworkable. However, if it is feasible, and an attacker does manage to spoof the system, then they will not be able to do it for long, as only the person with the actual biomarker that made the HUID can prove their right to that HUID in court. This will virtually eliminate fraud.

Sub/IDs

A crucial component to the privacy of the system is that it allows the creation of sub-IDs. **These sub-IDs (SIDs), linked to a HUID/SID, enable people to join or leave a group.** They could be an ID that marks you as a citizen or an employee or as one of the Nielson raters, or as a game player in a MMORP. **These sub-IDs also provide anonymization or unlinkability through blind signatures and/or zero-knowledge proofs. The signature is a single piece of secret data that can prove that someone's primary ID is linked to the sub-ID without revealing the primary ID. The sub-IDs should also allow for RBAC (role-based access control) to personal data**. For example, a sub-ID may grant access to only a phone number, name, and email address but not an HUID or SSN. The sub-IDs can also be generated with zero privileges, allowing access to none of the user's information, but still allowing the user's identity to be verified via an embedded secret that can be verified via challenge/response. So an application could demand that a user prove who they say they are and the client could then reveal that they know the secret information via a prompt, proving their identity but not revealing their PII. The system may also choose to use non-interactive zero-knowledge proofs (NIZK), which prove ownership/identity without even having to prompt the user depending on the needed security of the use case involved.

The system will also require single-use sub-IDs (SUSIDs). For example, to link an HUID to a NodeID (which IDs a client to the network) while simultaneously protecting the identity of the user and preventing that user from creating multiple nodes for themselves, we create a

protocol for single use sub-IDs that, once generated, cannot be generated again without revoking the original HUID and banning the original NodeID on the network, rendering the client unusable. A flag is set in the HUID blockchain that this single-purpose sub-ID slot is filled with a Boolean yes/no. If the flag is marked "no," the program will not allow the creation of a new sub-ID. In the event of compromise, the sub-ID can be deleted and generated again. This sub-ID presents some serious design challenges and the authors may not have worked out all major attacks against it, but the general concept should allow for solving the unique ID + anonymity problem if all attacks are considered carefully and appropriate countermeasures are created from the beginning.

RBAC sub-IDs have the potential to revolutionize our current, overly centralized, and brittle system of keeping everyone's PII on web servers and private databases for verification purposes. Our current system of information security is a house of cards. In order to prove that you are who you say you are, an entity such as a bank or a website has to keep lots of data on you. They have to keep that data secure and you have to trust that they can keep it secure. And your information is just a speck in the data they have to store, so they become a target-rich environment for attackers, be they individual hackers or a nation state. Why rob one man on the street when you can rob a bank? This is why we see major data breaches in the news virtually every day.

Anyone who has worked in the field of IT for any length of time knows a dirty secret: it's virtual impossible to keep an IT environment entirely secure, even when you have massive amounts of capital and human resources and a culture that respects security. Very few entities are Google or VISA, with the ability to effectively manage all of their systems. The chances of most big companies securing everyone's data are essentially zero. Modern systems are too complex, with too much code and too many variables, be they people or systems. **The solution to this is not better security for central stores of data. The real solution is simple: *Don't store that data centrally in the first place.***

Organizational Identities

The system also uses identities for "spheres" or groups within the system, borrowing the concept of "compound identities" from the paper Decentralizing Privacy: Using Blockchain to Protect Personal Data. It assigns a single general purpose ID to an entity, such as country, state, company, local sewing circle, etc.

Each organization would have its own public and private keys, its Organizational ID made from a hash of the public key, as well as its own signatures for message transmission, as will be described in the Networking section. An organizational structure might require Multisignature Keys for specific types of transactions, such as access to the corporation's various cryptocurrency wallets. Additional

actions may require the fulfillment of smart contract rules in order to complete a transaction, such as transferring shares of ownership between members in a corporation.

Those entities can then create provable but unlinkable RBAC sub-IDs, just as individuals can, for individual parts of an entity, such as a department or individual set of employees or for transacting privately with other businesses or other organizations in a private fashion.

Administrators are linked to organizational spheres. They control who can join or access certain spheres and their functions and privileges. Joining a sphere may be manual or automatic. For instance, an automatic system might require you to pass a test or provide proof of some particular bit of ownership, knowledge of a passphrase, or maybe even simple geolocation data over time to prove you exist in a specific area. This is discussed later in the creation of citizenship tests that can automatically grant people rights within a nation state.

For smaller spheres, such as a club or corporate entity, administrators can be designated during and after the entity's creation. Sub-IDs or HUIDs are granted access over the various functions of that sphere, which allows administrators to control who can and cannot join.

For example, a private comic book collecting group might have a simple structure whereby the creator of the sphere designates himself and a second ID as the administrators with "yes" or "no" access when a HUID or sub-ID requests to join. They may also have additional capabilities, such as the power to grant voting rights or take them away.

A company might have a more complex governing RBAC structure. The initial creation of the structure might designate board members or founding members as admins, with varying tiers of power, and give them the rights to grant administrator rights to other employees, such as HR staff. The admins would bring on or terminate employees through their administration console. Admins can also use that power to grant rights to various employee HUIDs, such as stock allocations, percentage of voting rights in the organization, pay tiers, access to various systems/documents/data, access to VPNS, and other day-to-day rights and privileges of company employees.

Backup keys or universal keys to ownership of an organizational entity can be held in a smart contract that has rules to follow in the event that no more administrators are available, critical employees are terminated, or ownership is transferred. Careful analysis of the design of such a framework will need to be carried out during the project's design session so there is a simple, reusable methodology that can be adopted by other organizations.

The organizational templates should be stackable, or able to draw from various baseline templates in a layered fashion, making the drag-and-drop creation of new entities easy. The rulesets governing such a template could be hierarchical, with custom rules at the top overriding generic language at a lower level.

Even more complicated organizations like a nation state might have designated trust groups that can grant or remove rights from its citizens, such as for violations of the law. A nation-state organizational entity might have the power to rescind voting rights under specific circumstances or to reinstate those rights, which would set flags in the blockchain that allow or disallow certain actions.

Info Wallets

Info wallets are encrypted private information stores that are held offline. **This information wallet might store personal details which can be selectively shared with an organization or not, based on their sub-IDs, which can grant role-based access control to personally identifiable information (PII)** OR **the individual can share proof that they own a particular piece of data WITHOUT sharing that data through a blind signature and/or zero-knowledge proof**. This allows for strong security in that centralized entities no longer have to keep vast troves of personally identifiable information. They can be backed up and additionally secured as necessary. A great deal of clever thinking on Bitcoin wallet security will benefit this effort as well. It seems that every day people are coming up with easy to manage Bitcoin wallets from hardware wallets to brain wallets that harken back to the cyberpunk masters of yore.

Distributing the PII back to the individual forces attackers to attack millions of small areas instead of a large, target-rich environment. Web companies and other centralized entities will no longer have information on site to steal.

Blockchain/Distributed Ledger

A central ledger is a blockchain, such as Hyperledger, Bitcoin's blockchain, or Ethereum's blockchain, which is similar to hashchain technology but has several unique properties. Whereas **a hashchain is a method of consensus among parties that already know and trust each other, a blockchain is a distributed cryptographic chain of record in a system that is used by untrusted entities to form agreement**. A hashchain requires that everyone generally trust each other, such as a local charity group that wishes to vote on where to have their next party. **A blockchain is a trust machine, to be used as a way to drive consensus between parties that may not always see eye-to-eye. Parties in the transactions may be friends, rivals, enemies, but the chain acts as a way to govern the protocol of their interaction. In essence it acts as a trust mitigator in the event of a dispute**. Nodes in a blockchain decide on consensus for canonical transactions to the system. Transactions may be conducted off-chain such as outlined by the Lightning Networks, a cascading, time-locked smart contract system, with the chain acting as an

arbiter of disputes. The system will utilize multiple blockchains, with a primary "blockchain of blockchains" with pointers to smaller chains that can be archived for retrieval later.

Blockchain of Blockchains

This project uses a central, established blockchain as a source of truth to store pointers to other blockchains. This could be a custom blockchain crafted for this project alone. It could be built upon the rapidly expanding codebase of an existing project such as Hyperledger or Ethereum. Or it might simply take advantage of extra fields for general purpose data stores, such as in Namecoin.

It is likely that the best bet for the project however is a custom chain as it will not be dependent on the critical weakness of existing chains which the authors call the "chicken or egg problem" which is that a person needs to have cryptcoin to use these blockchains. To use the existing ledgers, you have to acquire currency in that system. That is the major limiting factor. Perhaps you simply want to build a voting application on Ethereum?

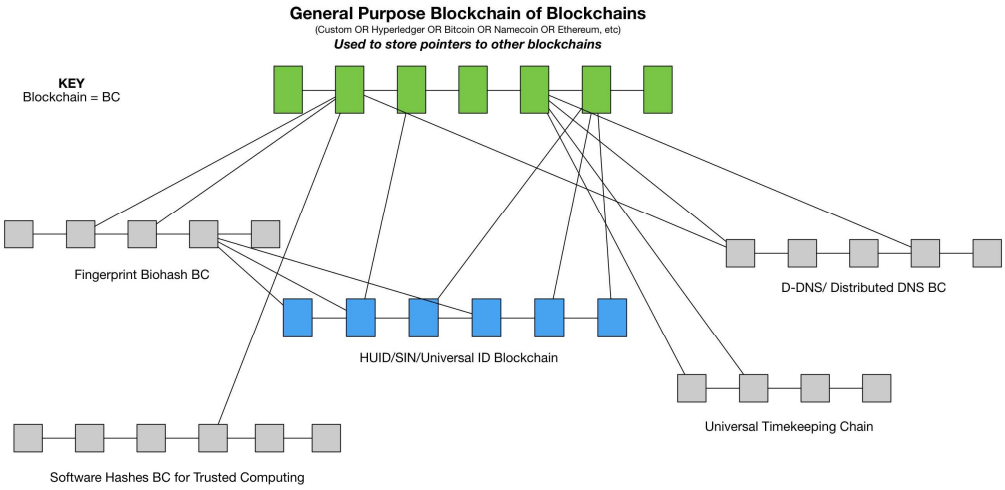
Unfortunately that requires you to acquire Ether, and everyone needs to pay to vote, even if it is some nominal fee. Since these currencies fluctuate so often, you may pay a tiny amount one day and then a lot the next. Even worse, you may run out of ether and not be able to vote. For the purposes of a nation running itself that makes zero sense. Some things must inherently be free.

This project differentiates monetary actions and simple actions, which spend non-currency based action tokens, as detailed later in the paper. This allows for actions on the system that should never cost money, such as voting, while still allowing commerce-based transactions as well.

The use of multiple chains allows for the atomization of types of data, such as ID data in one chain, distributed DNS in another chain, software date stamps, and hashes in yet another chain to facilitate trusted computing.

The outline of the blockchain of blockchains (BoB) is below. Since the BoB is storing only pointer data to other blockchains, it can be incredibly compact and space efficient. Chains with specific purposes can be pulled into memory as needed and discarded after a time of no use.

Most importantly, the BoB can be made secure with a new proof of work, detailed in the next section, that may prove to be the most import creation of this project.



Distributed Proof of Work

The creation of the HUID allows us to build a new type of Proof of Work, called a Distributed Proof of Work, as outlined earlier and elaborated on here.

Consensus is a method for determining agreement among the various nodes about which transactions are stored in the blockchain. The current consensus methods in use today are the Proof of Work (PoW) and Proof of Stake (PoS). While some other semi-novel concepts about how to secure the blockchain and create consensus have emerged, none of them have been able to dislodge these two primary methodologies.

Current Proof of Work systems have one serious problem: they almost inevitably end up as centralized systems. As mining becomes more and more competitive, it creates a moat around entrenched players, making it ever harder to raise the capital necessary to compete against them. This is similar to corporations today. Once you become Coca Cola, it is very difficult for new companies to rise up and create a competing soft drink to dislodge it. **This leaves us right back where we were with current monetary and social systems, with central players controlling and manipulating the flow of money and ideas indefinitely, leading to eventual collusion and 50% problems.**

This project's answer is a novel Proof of Work proposal called the "Distributed Proof of Work" (DPoW). It takes advantage of the HUID/SIN identifier to create a new kind of 1-to-1 client and node. **Every node is both a client and a mining server. In most instances this will be a person's desktop or phone with current technology levels and additional unspecified**

computing devices of the future such as AR glasses or smart clothing. Nobody is allowed to have more than one mining server. It is limited by their primary HUID which prevents Sybil attacks on the network and creates a level playing field for everyone. Additional Sub-IDs are not allowed to create new client/servers.

This 1-to-1 ratio makes the protocol essentially evenly distributed forever, with no one pool of servers or people controlling the power of the platform. This essentially eliminates the 50% problem. **It also has the added benefit of creating a built-in Universal Basic Income (UBI) as a reward for participation in the system.**

How does it work? It builds on the idea of the distributed hash table of the Kademlia P2P network, popularized by 3rd gen P2P systems like BitTorrent. Each client/server is assigned a NodeID stored in the distributed hash table of all systems on the network. Now lets take something like the Bitcoin system, which updates its blockchain every 10 minutes with a new block, which we will call the "block epoch." During those 10 minutes, transactions are broadcast to the network and miners gather them up into Merkle Trees and then compete against each to to solve a Proof of Work problem to win and release newly created coins and collect all the transaction fees from the last block epoch.

In the DPoW, miner nodes are elected randomly at the start of a block epoch for later blocks from the node IDs of the Kademlia style hash table. They are sent a randomized code that they will need to respond to correctly to acknowledge they exist within a set period of time. In order to avoid a scramble at the beginning of each block epoch, nodes are elected in advance for an as-yet-undecided amount of blocks, perhaps six. **If nodes are offline and do not respond, additional nodes are selected until they reach X percentage of the network. A pool of backup nodes is designated as nodes are expected to go offline during the process and diminish their pools. This will allow new entities to be swapped in as needed.**

The nodes are then grouped into random mining pools, similar to a built-in version of the distributed mining pools popular with Bitcoin private miners today. The PoW is then distributed to the various head nodes, also randomly selected, to distribute to their mining pools. The head nodes that hand out the random shards of the PoW and collect them to win coin are randomly clustered, and given a checksum to validate that another head node is not cheating. They all check each others' work so that no one node can simply go rogue and alter the transactions or attempt to cheat the others in the pool. The different mining pools elected compete against each other to win coins and transaction fees in whatever monetary system is built into the platform, be that Ether, Bitcoin, or some 3.0 currency created specifically for this platform.

As miners are called to the network, they are checked against the standard verifiable computing practices, such as that the binaries match hashes in the Software Blockchain, that the time is correct, etc., etc., as outlined in the Trusted Computing section later in the

document. This prevents and/or reduces malware affected nodes and other malicious attacks on the network.

The reason for the random pools instead of having all nodes running DPoW all the time is to save on electricity and battery life of devices such as cell phones, until dedicated ASICs are popular and massively distributed. Mining is an inherently hardware and electricity intensive proposition, so distributing the mining duties randomly on the network makes it economically and energetically feasible, while preserving end user experience.

The winning mining pool collects the coins and transaction fees and then the mining servers become dormant again, leaving them as simple clients until called on again to participate. This allows each person who participates in the network to have a built in Universal Basic Income simply for participating, since every node will eventually be called upon to secure the system, making the UBI real and plausible right now, something that is virtually impossible with current systems due to our unequal income tax and earning levels, not to mention most current governing bodies' inability to agree on practically anything of significance.

As more clients and servers join network, the consensus becomes even more secure and simultaneously more evenly distributed/egalitarian, unlike current PoW systems that leave us with powerful centralized miners as the system develops.

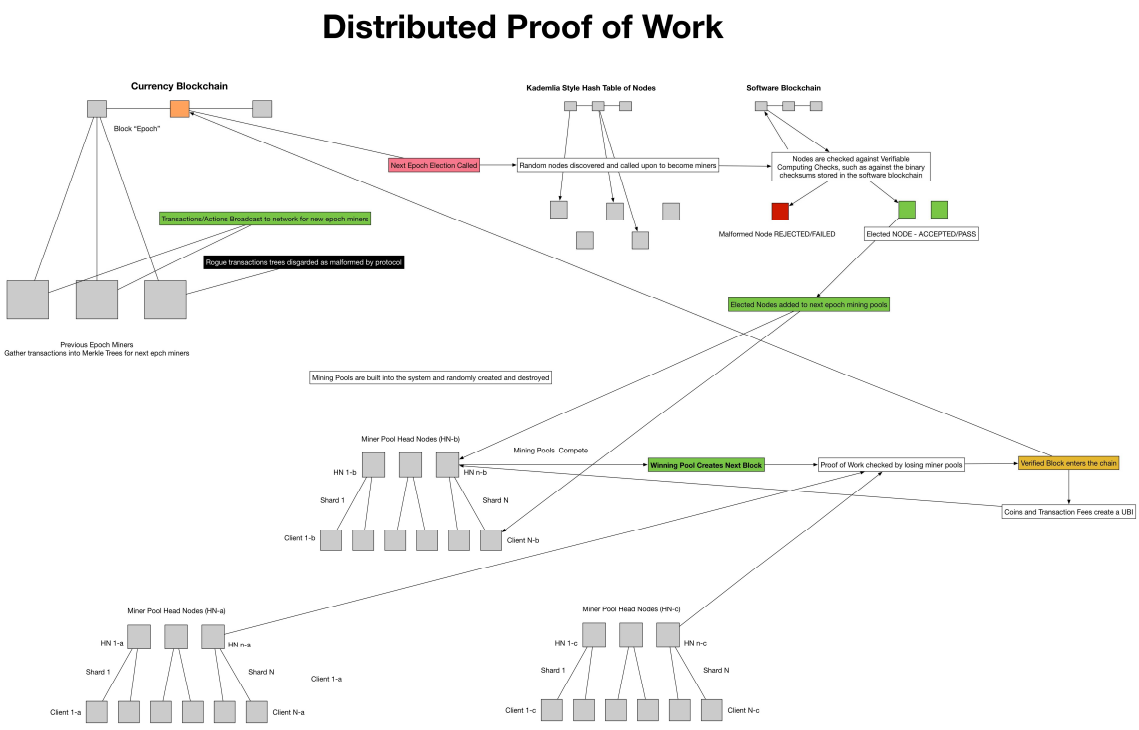
It is also worth considering a reputation system for miners that allows reliable miners to be called on more regularly. However, any such system must be carefully weighted so as not to allow the same miners to consistently drive all traffic. Trust would be defined algorithmically, for example based on whether the node responded to all requests throughout time, whether it ever had malformed packets or out of date software, whether its binaries have ever been altered from standard hashes, the node's speed, dropped packets, etc. This idea is outlined in more detail in the Networking section. Nodes could be less likely to receive requests if they have issues or a low reputation rating or even blacklisted if they have repeated issues such as compromised software, altered binaries, etc.

Lastly, blockchains can grow very large, so how would a small client like a cell phone store the whole blockchain? The answer is none of them would need to at all. Instead, clients need to keep only a current subset of the most recent transactions and the rest of the blockchain is sharded up and replicated in pieces to all clients, so that each client is responsible for storing a small portion of the historical blockchain. If nodes drop out or go offline a shard may be replicating temporarily from an existing node to maintain a proper percentage of the chain to avoid losing any piece of it. Because other nodes contain redundant instances of that particular shard, all of the instances would have to be destroyed to destroy that part of the blockchain, which makes the protocol space efficient and able to run and be stored on a mobile device. In addition, the blockchains could be backed up by anyone or by a backup company to keep the system safe in the case of total disaster and to allow for restoration later, while providing evidence of their

completeness. Lookups would use a DHT to find which nodes currently hold the necessary part of the blockchain for a lookup, should an old transaction need to be verified, and multiple clients would be checked to ensure accuracy.

That means that the Distributed Proof of Work delivers the following in a single platform:

- **An egalitarian system which is highly resistant, if not completely immune, to centralization**
- **A feasible UBI system as a byproduct**
- **High security**
- **Dynamic consensus which encourages participation from everyone in the system**
- **Energy efficiency**
- **Eliminated need to develop ASICs and the resultant ASICs arms race in current cryptocurrency platforms**
- **Minimal storage requirements**



Verifiable Computing

We must assume that every node is untrusted and/or compromised and design the software in such a way that it is resistant to subversion. Take smartphone voting as an example. While eventually Cicada might simply have her own hardware designated for running this platform online, such as a simple card computer with no ability to run other software, running with a stripped down micro-OS that is highly resistant to damage and malware, in the early days of adoption we will need to create it for running on cell phones or AR glasses.

A software blockchain could contain hashed stamps of binaries, software versions, protocol versions, etc., and a system booting on a cell phone or any future end user device that would check against the chain with a challenge to ensure that their local code has not been compromised. Should it fail to reach the network, fail a binary or date stamp check, or otherwise not pass any part of a comprehensive sanity check protocol (such as the sanity check protocol that miners go through for each transaction broadcast to the Bitcoin network to ensure that the transaction is not malformed), it would refuse to boot/run or otherwise perform only a small subset of features.

Once the software boots, it would pull down the code from a distributed blockchain backed storage code repository to perform a binary update, perhaps a containerized layer style update with built-in layer security checks, including signed layers, as further evidence of trust.

Ironically, this idea was inspired by malware writers who used such techniques to prevent analysis and reverse engineering of viruses by anti-virus companies. The malware would boot a small kernel and check that it could communicate to the network and it would also look for the presence of virtual machine drivers, as anti-virus companies often used virtual machines to isolate the virus. The systems would also only accept commands from pre-trusted centralized command and control units that signed messages with their private keys. Eventually anti-virus companies were forced to attack the command and control servers themselves to attempt to disrupt these networks, which further proves the weakness of any centralized system. BitTorrent was often attacked by similar means, going after the centralized websites that listed torrents or the DNS servers that pointed to those links.

Upgradability

The system must store version, protocol version, software versions, date and time stamps, as well as other metadata in the system to allow for enforcing upgrades. For example, should a client version contain bugs, the system might require an upgrade before it can conduct business. Likewise, an encryption protocol may become outdated due to the invention of new computer hardware, and as such it would need to be forcibly retired and replaced with a new one. Updates should happen

through a container-like model where layers are added on top of existing binaries or binaries are flushed and re-imaged with new copies that match the correct agreed upon hash in the software blockchain.

Networking Outline

One of the most crucial pieces of a decentralized application is the robustness of its networking protocol. Fortunately, P2P networks have evolved significantly in the last fifteen years. Of particular note is the Kademlia design, a distributed hash table (DHT) with a novel XOR system, which provided the framework for a number of successful and popular 3rd generation P2P apps, notably Gnutella and BitTorrent, which have successfully scaled to 100s of millions of users. Additional research work has focused on improving the efficiency of the routing protocol to minimize hostile actors and inefficiencies that stem from nodes going offline while still having entries in the DHT due to the DHT's eventual consistency model, which will include nodes that are no longer online for a short period of time. Broadly, those research efforts have fallen into two categories:

- Improving the reliability of nodes with a reputation system
- Improving routing efficiency through the use of fuzzy logic or biologically inspired algorithms

This project advocates for both of these ideas and adds a novel concept of the HUID that will help mitigate or eliminate the most widely used attacks against Kademlia style P2P nets: Sybil attacks.

As stated in the Design Specification document linked above:

"A Kademlia network consists of a number of cooperating nodes that communicate with one another and store information for one another. Each node has a nodeID, a quasi-unique binary number that identifies it in the network.

Within the network, a block of data, a value, can also be associated with a binary number of the same fixed length B, the value's key.

A node needing a value searches for it at the nodes it considers closest to the key. A node needing to save a value stores it at the nodes it considers closest to the key associated with the value.

NodeID

NodeIDs are binary numbers of length B = 160 bits. In basic Kademlia, each node chooses its own ID by some unspecified quasi-random procedure. It is important that nodeIDs be uniformly distributed; the network design relies upon this.

While the protocol does not mandate this, there are possible advantages to the node's using the same nodeID whenever it joins the network, rather than generating a new, session-specific nodeID."

In essence what the above says is that we have a 160 bit NodeID, which should be more than large enough for everyone on the planet for thousands of years to come, even if humans ever manage to push out into space and dramatically expand their numbers.

This project takes the stance the there are definitive advantages to the node using the same nodeID every time it joins the network in that it prevents, mitigates, or impedes a number of known attacks on the system. In addition, this project bans freely generated IDs on the client side. Generating the ID with the help of a blockchain and from the HUID non-interactive zero knowledge proof, single use sub-ID, and using that sub-ID as a seed to creating the NodeID works to prevent users from creating multiple nodes on the network.

The health of the network, as outlined in the section on the DPoW spec, depends upon the one HUID/one client design of the system. As such we utilize a novel approach to NodeID creation, in which the NodeID is generated from a single-use, non-interactive zero-knowledge proof token/sub-ID of the HUID. In particular, we spin up a sub-ID with zero RBAC privileges to access a client's information, to prevent linking a HUID to a NodeID (i.e., unlinkability), which in turn prevents spying on that node to gather intel on voting or participation in the platform. The sub-ID includes a secret piece of information that can be verified through challenge response which is owned by the user, but that does not leak additional information about the user.

We will discuss attacks and other countermeasures against P2P networks in this section as well as the sections "Attacks on Kademlia and Trust Networks" and "Defeating Attacks on Kademlia and Trust Networks."

The design specification site also specifies the following, which outlines routing and routing tables in the system:

A Kademlia node organizes its contacts, other nodes known to it, in buckets which hold a maximum of k contacts. These are known as k-buckets.

The buckets are organized by the distance between the node and the contacts in the bucket. Specifically, for bucket j, where 0 <= j < k, we are guaranteed that

2^j <= distance(node, contact) < 2^(j+1)

Given the very large address space, this means that bucket zero has only one possible member, the key which differs from the nodeID only in the high order bit, and for all practical purposes is never populated, except perhaps in testing. On the other hand, if nodeIDs are evenly distributed, it is very likely that half of all nodes will lie in the range of bucket B-1 = 159.
Bucket Size

Contacts

A contact is at least a triple:

the bigendian nodeID for the other node

its IP address
its UDP port address

The above basically says that each node maintains contacts which it can use to route to other nodes in the network. Those contacts generally consist of a NodeID, an IP, and a UDP address. In the case of Cicada, NodeIDs are difficult to forge as they are generated from a hash of the public key of the single-use, zero-knowledge proof sub-ID and nodes use their own public/private key pairs and self-signed certificates to sign and secure messages. But before digging into those details, let's look at the various attacks on P2P networks.

Lastly, for the purposes of secure end-to-end messaging communication, each node in the system can act as a router and VPN, forwarding packets onion router style, made famous by Tor. However, unlike Tor, which requires dedicated endpoints to be set up by tech-savvy users or which allows malicious users looking to compromise the network to set up nodes to capture traffic, a pure peer-to-peer system would make it much more difficult to assault. Just as with the DPoW, nodes would be randomly selected to act as temporary VPN/onion router node endpoints in the same way that miners are elected. They are drafted for a period of time and their NodeID is stored on a VPN blockchain for forwarding specific types of messages. This will eliminate attacks on VPN endpoints as a means to suppress dissent and alternative viewpoints.

Attacks on Kademlia Networks

Now that we have a general understanding of the essential Kademlia concepts, which you can dig further into at the design site, we can shift to how to enhance and improve the basically sound framework provided by the authors.

Much of our thinking on how to improve the networking of the DD platform comes from the excellent research paper called "S/Kademlia: A Practical Approach Towards Secure Key Based Routing." The authors go a long way to creating a secure version of Kademlia, outlining a number of attacks (briefly summarized below), as well as excellent though incomplete solutions to these attacks. However, this project deviates from the thinking of the paper in one key aspect:

- Defeating/Impeding Sybil attacks

The authors did not know of or conceive of a universally unique human identifier, and as such had to use a number of techniques to attempt to mitigate Sybil attacks that are largely solved simply by having a HUID.

Sybil Attacks

The S/Kademlia paper makes the assertion that "Douceur [8] proved that [Sybil attacks] cannot be prevented, only impeded."

We make the case that they are more dramatically impedable than previously thought via a proper, successful implementation of a HUID/SIN, as described earlier. Since one client/one node is the default design of the system and users are not allowed to spin up additional nodes, many of the typical Sybil attacks which involve generating multiple IDs and spinning up a number of clients to gain partial control over the network are now infeasible. The only caveat is that compromised nodes, i.e., malware-infected nodes, still provide a feasible path to a Sybil attack as well as a variety of other attacks.

As such, a strongly verifiable computing framework, as outlined earlier, is necessary to help ensure the reliability and trustworthiness of this platform, not just for Sybil attacks but for a number of other types of attacks as well. Software blockchains, forced checking or binary signatures, time stamps, and other sanity checks should severely limit the number of malware attacks on the network to an imperceptible amount. This is akin to reducing the number of flu particles in a host system, such that the host is still infected but not to a degree that the overall system is still threatened.

Attacks on the underlying network

We assume that attacks on the underlying network, be it IPv4, IPv6, or a wireless peer-to-peer mesh network are possible and unpreventable. Since we are talking about building an overlay network to piggyback on existing insecure protocols, we must simply treat the underlying network as unreliable and potentially hostile at all times. We also assume that nodes can spoof IP addresses, be compromised, and as such allow denial of service attacks on the overlay network.

The focus of the project is to make the overlay network as secure and reliable as possible while assuming that the underlying network can be assaulted. This is the philosophy of protocols like Bitcoin, which has remained remarkably immune to serious attacks, other than denial of service attacks, despite moving billions of dollars around in its ecosystem. This is certainly not for lack of tying. Hackers who manage to successfully crack the system would stand to yield a significant payday. As such it is safe to assume that there are many attackers working on it at any given time and that they have generally failed due to the reliability of the design, the robustness of the codebase, and the design team behind it.

In addition, the Cicada platform will include a mesh network design for operating when no under network is available, such as when networking and communication infrastructure is destroyed in a war, or when the platform is operating in openly hostile territory.

Node Impersonation

Node impersonation is where a NodeID attempts to impersonate the ID of another node on the network.

Eclipse Attacks

This attack attempts to put malicious nodes on the network to force traffic flow through them and cut off or hide good nodes from the network**.**

Churn Attacks

An attacker who has compromised a number of nodes may force them offline and online again and again, creating instability in the network.

Reputation Attacks

One category of attacks that are not outlined in the S/Kademlia paper are reputation attacks, because the authors were not proposing to expand the system using a reputation-based system. Reputation attacks tend to focus on either ruining the reputation of another client, such as knocking them offline repeatedly to make them seem unreliable, or on systematically building a perfect reputation only to subvert that goodwill later.

The primary attacks on reputation systems, as outlined in the paper A Survey of Attacks on Reputation Systems, are the following or a combination of all three:

Self-promotion

This is where a node attempts to create a positive reputation, only to exploit the system at a later date when its trust is highest.

Whitewashing

An attacker attempts to repair their bad reputation by exploiting vulnerabilities in the platform. In other words, this is the equivalent of breaking into the teacher's office to change your grades.

Slandering

Attempting to destroy another node's reputation unfairly in order to get the node blacklisted or banned.

Defeating Attacks on Kademlia and Trust Networks

Now that we have outlined various attacks on the system, let's look at ways to prevent these attacks. A well-designed system includes countermeasures for all known attacks as part of its ground-up design.

First, Sybil attacks and node impersonation can be defeated by not allowing the nodes to freely create their own IDs. Since the HUID's single-purpose sub-ID is linked to its NodeID, it should be very hard to spoof a NodeID on the network. In addition, we propose that the NodeID be created from a hash of the public key of the sub-ID, allowing the public key of the sub-ID to be projected into a NodeID blockchain. This would allow other nodes on the network to determine whether a NodeID is valid, simply by calculating its hash with the sub-ID's available public key.

Second, the best way to prevent most of the other attacks listed is to have each node create its own public/private keypair. The public key of the node is projected into the node blockchain. **This idea builds on the work of Michael Kohnen's comprehensive TrustedKAD PHD candidate dissertation. In his paper, Kohnen outlines a method to prevent slandering and other attacks on reputation by generating a self-signed certificate for each node that includes its IP address and port, as well as a date and time stamp.** TrustedKAD differs from the Cicada platform in that it does not include the node's public key or the concept of a blockchain. The Cicada platform will use the node's ID in the certificate system, as it is not attempting to prevent Sybil attacks at this layer, as it has already worked to prevent them during the HUID and node creation process. The node signs all messages with its private key and includes the certificate in the message header. If its IP address changes, the node generates a new certificate. The receiving node can check the hash against the stored public key, as well as check that the packets originated from the correct IP and port. If the signature or underlying network information do not match, the packets are discarded.

The system also used a trust-based system to rate nodes. Nodes are rated for correctness of protocol, whether they have up-to-date software, whether they have ever sent malformed packets, etc. In addition, the system may use genetic or neural pathway inspired checks such as how long a node has been online, how close it is, whether it has ever connected to the sending node in the past, how many other nodes it connects to and knows about, how many hops away it is, whether its time is correct, etc. An exhaustive list of application and network checks will not be examined here, but will be further outlined in the project's formal design phase.

Ratings are binary. 1 is for good and 0 is for bad. With too many negative scores, a node may be blacklisted from the DHT routing table for a period of time that increases with each violation, followed by an eventual total blacklist. The total blacklist would prevent the node from participating in DPoW and hence deprive it of UBI and it would prevent it from forwarding messages, so nodes would have a strong incentive not to lose their UBI. However, it might still allow a user to vote, as votes would be taken away or granted by governing bodies in a nation state or via automatic violations. In addition, for permanently blacklisted nodes, a bonded/trusted authority may be created to restore access, though any such entity and any system that allows for total blacklisting must be perfectly designed to prevent ensnaring users that were unknowingly compromised, which will likely be the vast majority of

end users. As such, total blacklists should be considered impractical in the early phases of network design.

Lastly, because new nodes cannot be very efficiently evaluated beyond software version, IP address reputation, etc, the Cicada platform includes a grace period for new nodes.

The ratings for nodes must be continual. As Kohnen notes in the TrustedKAD paper, it is necessary to continually "rate the behavior of a node and to avoid interaction with it if it does not behave correctly,"

The problem of where to store the rating information could be solved by storing it in the DHT and including a method to ensure that no node is able to store its own reputation information, even if that information is a copy, to prevent manipulation of that information. A second option is to again use a fast moving blockchain (one with less then ten minute blocks) as a trust mechanism for storing the information. Since the Cicada platform uses sharded up chains to store them in the small space of local clients, a flag would again be needed to ensure a node does not store its own reputation information.

The methodology of storing the reputation information is one that requires further research, and the above suggestions should only be considered starting points. However, considering the number of research papers that include some type of reputation system, it is likely that a viable solution already exists. However, this project takes a firm stance against the S/Kademlia "supernode" concept, which includes a large number of supernodes that act as trusted arbiters of the network. In the opinion of the authors, those supernodes would create the very weaknesses we are trying to avoid. However, it may be possible to create a modified form of the supernode. Recall earlier that we recommended drafting nodes into temporary onion router/VPN endpoint nodes. Perhaps a temporary cluster of high bandwidth nodes, verified through repeated tests, could be elected for brief periods of time to facilitate better traffic throughput and management while not allowing supernodes to become dedicated attack points.

Mesh Network Fallback

In addition to building a scalable overlay network that secures communications over the top of compromised and untrusted networks such as the Internet, the project should work to create a fallback to a mesh network. Mesh networks are crucial to overcoming obstacles when a society's centralized communication infrastructure is destroyed or so compromised through collusion with an authoritarian power that it cannot be trusted under any circumstances.

Mesh networks are not a new idea. Many militaries rely almost exclusively on mesh networks, because field forces simply cannot count on existing infrastructures in hostile territory. Utility

companies often use mesh networks to gather meter stats. A number of darknets exist in San Francisco that are not internet connected and "invite only," in that you must know a member to be asked to join the network. In war-ravaged countries a mesh network for running Cicada could become essential, and so further exploration is needed on this topic.

A number of open source and commercial mesh networks are already coming to the marketplace, such as FireChat from OpenGarden, so we will not explore this concept in-depth in this paper. Instead, we would look to consume a well-written, transparent, and scalable mesh network from other project first, before creating our own from whole cloth.

MixNet

Nodes will also be randomly drafted as tumblers for forwarding messages. MixNets are useful for further obscuring information about the origin of a packet, vote, or message, by scrambling up those messages with other messages such that the input and output are difficult to trace. MixNets are particularly useful for voting systems.

Nodes are drafted randomly by the system, during the same round as the draft for the DPoW election to secure the next block. For safety reasons, nodes for the MixNet are not the same as the DPoW miners. The protocol would work to choose different nodes based on flags in the DHT. Malware infected nodes or nodes with bad reputations are blacklisted by the reputation system and not eligible for selection. Votes cast are tumbled with other votes to obscure who the original vote caster was, in order to further protect privacy.

P2P Communication System

A fully end-to-end encrypted communication system for establishing peer-to-peer or multipeer communications is essential for any society to be able to communicate freely and securely. Existing systems expose everyone to mass surveillance, assault by malicious actors, and compromise of their PII. In many countries, laws also mandate keeping backdoors in commercially provided encryption systems, and this problem will only increase as ignorant legislators weaken our communication systems under the guise of safety and security. An open system is the only answer. It is also necessary to implement off-chain transactions, such as those proposed by The Lightning Network, which uses a series of hashed, time-locked smart contracts to facilitate most transactions off the Bitcoin blockchain, thereby reducing the eventual size of the blockchain blocks while dramatically scaling the number of transactions per second to scales much bigger than even a company like VISA is currently capable of. If there is no secure system to find people to transact with, the Lightning Network does not work.

We can use David Chaum's PrivaTegrity as a template here. He proposes a scalable mixnet called cMix to provide unlinkability and secure communication. In the Cicada network, nodes are randomly drafted into MixNets, which differs from Chaum's design which relies on centrally trusted servers.

Chaum also provides a novel but controversial universal decryption scheme. Universal decryption keys are generated and sharded up among nine different server farms in different geographies. If all nine servers agree, the sharded key is reconnected and we are able to decrypt communications on the network. This is to prevent something "totally evil" from being hidden. Once the keys are used, they are destroyed automatically, and new ones are created and sharded up again, so that the nine servers must again universally agree to decryption.

Chaum's proposal sparked fierce debate, because there are some obvious problems with this methodology. For example, the nine systems can collude, or if they are actively hostile and cannot agree then nothing can be done, despite the vast majority of users agreeing to the proposal.

A direct democracy and smart contracts provide a simple solution that is powered by the people. Universal keys are created and encumbered with a script and stored in the keys blockchain. Upon a 2/3rds majority vote, the keys can be released for a specific purpose, written in the Legal XML language, as a part of the voting contract. Once the keys are used once and the rights expressed in the contract are fulfilled, as enforced by the contract parameters themselves, the keys are destroyed and new ones are created and re-encumbered, requiring an additional vote to unlock.

In a traditional system, people must make a choice between full encryption or compromising encryption for everyone. In our system, transactions are fully encrypted by default, but the people can vote to decrypt certain transactions or audit transactions from a group.

The system may also allow for the creation of sub-master keys. These sub-master keys can be given to an organization, such as a police department or central investigation unit, but can also be revoked by the people in the event of abuse by that trusted organization. In the current real world system, trust must be given, and if that trust is broken there is no method to revoke that trust. One clear example is the mass surveillance system currently in place all over the world, with no opportunity to hold those entities responsible for direct violations of the Constitution or law of the land.

Additional Components of the System

Now that we have outlined the major components of the system, it's time to address some of the additional components necessary to make it work. These pieces will be covered only briefly, and this list may not prove exhaustive once the project gets further along.

Stores of Value

Property deeds/intellectual property ownership rights/benefits/shares/contracts are all stores of value that can be created, distributed, reclaimed, revoked, and dispersed to anyone in the organization.

Money

Currency, a store of value which can be exchanged for goods and services, is a part of the system. Distribution of that currency is covered by the system. While it can be argued that money should be lumped in as a "store of value," in actuality it is a special case in that it can be exchanged for things perceived as valuable, be that value real or imagined.

Action Tokens

Action Tokens are distributed to IDs or sub-IDs to allow or disallow actions to take place. While they are like currency, they have no actual store of value. We may choose to do this by handing out the absolute smallest value of a particular currency, so that eight decimal places of Bitcoin aka a Satoshi is one "action token" for voting, changing a ruleset, changing a law, enacting a law.

A much better choice is to simply create and distribute a completely separate set of blank tokens, taking into account overall usage in the system to replenish the supply. Just as the Bitcoin blockchain calculates the power of the network every few weeks, we can choose to track the overall usage of action tokens and variably produce new ones to meet the demand. When a miner wins a specific block round, the protocol would check the current action token blockchain tracker to see the current number of new tokens to create. These action tokens would be stored in individual system wallets, owned by nobody, controlled by smart contract, and used by the entire platform. An end user node would request a new supply of tokens as they run down and their amount would be enforced by the blockchain and released by these system wallets.

Action tokens with no monetary value are an important innovation in future decentralized apps, because they allow for certain functions that simply should not cost money. A currency can always fluctuate, raising the price of performing simple tasks, which is not desirable in a complex system. Some actions, such as participating in the system through voting, should always be free. We do not want participation limited to how much currency you can acquire. Current decentralized app platforms have a chicken or egg problem. I need currency to use the system, but I have to acquire that currency outside the system to use it. This is far from ideal. In addition, app designers may wish to provide free services to people and paid services later. Action tokens make it easy to adopt the platform, which should help to scale it rapidly and bring in less tech-savvy users.

Blind Tokens

These are tokens that utilize blind signatures and mixnets to keep transactions anonymous, such as in voting systems.

Transactions

Transactions are any action taken by the collective or individuals within the collective, such as, but not limited to:

- Distribution of value
- Transfer of ownership
- Votes
- Law creation/repeal
- Communications
- Smart contracts
- Off-chain payment chains

Proposal System

This is a system whereby people can put forth proposals to vote on.

The proposal system is two-fold. Expert groups can be called to put forth fast track proposals that are given national votes. For example, a group of renowned economists might design the laws for governing commerce, while a group of engineers might design safe operating limits for vehicles. This has the strong advantage versus our current system of allowing people with actual knowledge of the subject to put that knowledge to work, as opposed to what we have now, which is agenda-driven non-experts (aka politicians) who make laws for other people without any understanding of what they are doing.

The citizen proposal system is different. Anyone can put forward proposals. However, the system routes any proposal through a random series of voters, starting small and going wide. A proposal might be sent to 100 people, then 1,000 if it passes, then 10,000, and so on through a series of defined thresholds designed to filter out useless/baseless/disingenuous proposals. An algorithmic filter can analyze the proposals through something like Bayesian statistical inference to immediately eliminate duplicates.

If a proposal does not meet a certain number of votes, it does not move forward. If people simply don't vote on it, it does not have enough interest and it does not move ahead. If it is voted down, it does not move ahead.

This graduated and filtered approach prevents spam to the system, as well as marginal and fringe proposals from ever gaining steam.

This system in particular will require careful consideration and design. Additional methods of filtration may need to be created to keep people from repeatedly submitting garbage proposals, such as limiting the number of proposals per person, per month or year. If proposals are passed by an individual, a reputation scoring system might allow them to put forth additional ideas, provided their reputations stays at a certain threshold.

Deep learning systems may come into play here as well as they continue to develop. A system that can learn common proposals, streamline them, and present them to the whole might prove very valuable. Deep learning might also detect novel attacks on the system.

Rulesets

In a nation state, a ruleset could be a series of laws that govern citizenship or business relationships or the rules for borrowing money. In a small organization it might be bylaws. In a corporation it could be a mission statement or a series of rules governing behavior. These rulesets would be applied as "connections" or "weights" to the HUID to identify people as a member of a group.

Rules/Law Framework

XML for law/rules, which exist in a given ruleset. Define a series of law allowed/disallowed statements that all contracts/laws/systems must be defined as. These consist of variables, conditionals, etc. Law must be expressed as a series of logical statements so that the system can parse these laws.

Voting Framework

This encompasses voting integrity, voter intent, collecting and tabulating votes, determining result of those votes. In short, this is an E2E framework or end-to-end voter verifiable framework.

The system must provide end-to-end integrity for all votes cast, while preserving anonymity with blind signatures, mixnets and cryptographically verifiable voting records. Much research exists on E2E, so this paper will not spend too much time discussing all of the ins-and-outs of this.

Vote Suggestion System

A system allows for suggestions on how to vote. Based on a question survey, such as demonstrated by the "I Side With" website, a set of basic rules can be created about how a person would particularly vote. The system would then examine the law XML and come to a conclusion, marking

the document/proposal/bill/rule change with green/yellow/red based on how likely you are to agree with the majority of the provisions. This could eventually lead to an auto-voting system.

Auto-Voting system

A deep neural net can be trained over time on how you vote, learn your predilections, and predict your votes. This would allow you to set up an auto-voting system for yourself that maintains engagement with the system without manually having to vote on everything that happens in the network, which can lead to voter fatigue. The system could have three possible defaults:

1. Full auto voting, where the system votes on everything for you and sends you an auditable receipt, with the option to change your vote within X number of hours or days.
2. Semi-auto voting, where you retain manual votes on crucial issues to you as an individual, such as guns or abortion, and the system votes on everything else.
3. Full manual voting, where you vote on every issue presented to you.

Rewards

Leaderboards, rewards for participation, and reputation systems.

The system can "gamify" voting by rewarding and incentivizing certain behaviors while de-incentivizing bad behaviors within the system.

Rewards should never be allowed for voting a specific way, only for continual, thoughtful participation in the system.

Data Storage

This could be files, information, or anything else that a person or organization would like to stored/hosted/distributed in the system. It could be a hybrid on-chain/off-chain system, such as the one proposed by a group of researchers at MIT, called the Enigma system. Since there is plenty of ongoing work in the field of distributed data storage, this will not be a focus for the system. A simple keypair-based distributed storage system, with sharded, encrypted data and audit logs stored in a blockchain, will be enough for basic transactions and state control. We will likely be able to consume other projects for this purpose.

Sub-Organizations

Sub-organizations are dependent organizations (such as a cabinet department in a nation-state or an engineering department in a corporation) that can exist in within a larger organization.

Information Post

This could be an URL or BB service or forum that can people can share information on and which they can collaborate.

Marketplace

This can be a model of a marketplace, such as a swap meet or a decentralized currency trading exchange.

Modules/Extensions

An extension might be something like an auto-voting system, which is a can of worms. Modules can be marked as experimental/alpha/beta/rc/prod.

Procedural Patterns of the Platform

In order to get the system up and running smoothly over the long haul, a number of methodologies, rules, and protocols will be necessary to keep it error-free, secure, and functioning well.

System Bootstrap/System Instantiation

The system must have a protocol for bringing an organization online. What are the desired steps/patterns that are necessary to bring the system online? This will not be closely addressed in this paper, though it will require very careful consideration. For example, what are the steps necessary to bootstrap the system and what are the first steps any person and/or group must take to participate?

Founding a company requires very different actions from the birth of a nation. For example, a company may acquire capital, create a mission statement and business plan, create a legal framework, invite leaders, hire employees, etc. A series of basic steps would need to be outlined for various types of uses for the system.

These use cases should be definable in YAML or another type of consumable text file and usable/modifiable by anyone in the system.

An example of a nation state bootstrap might look like this:

- Form a provisional "working group" organization to help bootstrap the system
- Download the client/node
- Code an XML law framework or select from a provisional constitutional framework
- Call and organize a vote on a full constitution
- Create departments
- Choose heads of a department

Procedures of the DDD

Defining Citizenship by Linking HUID to Citizenship IDs

What is a Citizen?

A citizen is a "person who is entitled to enjoy all the legal rights and privileges granted by a state to the people comprising its constituency, and is obligated to obey its laws and to fulfill his or duties as called upon. Also called national. See also domicile and resident," as defined by the business directory.

Defining a citizen or non-citizen and associating those rights to a person's HUID is a multifaceted issue, but one that is solvable. Unlike other voting or democracy systems, Cicada allows votes to remain anonymous by have personally identifying information used once to ID a citizen before being discarded, retaining only the public key of that information.

Citizenship is a human defined construct. Unlike DNA or fingerprints, which are immutable, unique to the individual, and designed into our biology, citizenship is conditional, mutable, and revocable. As such, there is no simple way to define it without artificially created documents (such as a birth certificate), as opposed to naturally gifted attributes (DNA/Fingerprints). Countries are defined by humans and as such are changeable in terms of borders and composition. They exist for a set period of time before those boundaries change, or disappear altogether in the event of a country's collapse. In short, they are ephemeral.

Citizenship is enshrined in the **citizen blockchain. Additional sidechains can be created for different types of information, since each chain is merely a different kind of data. All chains are pointed to by the blockchain of blockchains. For example:**

- HUID chain: HUID {human unique identifier}
    - Backed by voiceprint, DNA and/or fingerprint chains
- Citizenship chain
- Crime chain: convictions {must be totally anonymous}
    - Add a module for revoking or restoring based on criminal convictions, with a blockchain for crime convictions that is anonymous/hashed and a check against that.

Vote tokens can be issued to most citizens by a DGO (Decentralized Governmental Organization), as they are able to fulfill the following citizenship requirements. This can be one of the first DGOs that is established by the system after it is bootstrapped and instantiated.

There are three types of people in a country:

1) **Current Citizens**: We must be able to define current citizens, if the government is "rebooted" with this system. Pre-existing citizens must be verified by looking at past evidence of citizenship. There are two scenarios where the system would be implemented.

1a) The government is replaced through displacement/revolution.

1b) The existing government adopts the system and adapts to it or adapts it to its current system.

In both cases you would look at pre-existing documents, even if those documents do not have current legal status {as in the case of revolution}. Those documents would not be stored in unencrypted form. Instead, they would be hashed and encrypted, either through scanning (in public terminals for those without regular digital access to scanners) or by putting in their unique number (in the case of SSN or equivalent) and sending it to an appropriate storage chain for verification and then stored in a local info-wallet. These are then attached via pointers to an anonymizing HUID sub-ID to prevent attackers from using the information if they can successfully attack any of the sidechains. Multiple documents attach weight to the ownership.

We must prevent against Identity theft. The simplest way is with a legal system that allows a person to show up at a court and scan their iris and to verify their ownership of a HUID in the case of a dispute. If not, the ownership is transferred back to the correct owner through a flag on any alternative records made through the reputation system.

Stealing an online ID would become incredibly difficult as a birth certificate, once hashed, encrypted and linked, cannot be applied to another HUID, so it is not something that is easily gamed.

Examples of pre-existing documents that can be hashed/attached to SIN.

- Birth certificate.
- Driver's license.
- Passport.
- SSN or equivalent.
- Property ownership in country. (Not necessarily a citizenship requirement, but may add further weight to the system.)
- Hashed geolocation time stamps over a period of time.

2) **New Citizens:** The second problem is how new citizens are added, i.e., how are immigrants, refugees, etc., allowed into the country?

This would have to be through a DGO (Decentralized Governmental Organization).

A series of rules would define a citizen. This would be a pluggable system. As long as all of the following criteria are met, a person would be granted automatic citizenship in a nation-state. For example, they may need to:

- Pass a citizenship test. They could take this online, and the passing score would count as one of the variables needed.
- Commit to investing a certain amount of money. A check against bank accounts or crypt-coin wallet would ensure correct funds exist and are under the ownership of the individual.
- Live in the country for XYZ years. Property deeds (which could be traditional or blockchain based) could be hashed along with a timestamp and a ticker. Once the number of years has passed, the criteria are met.
- Not come from a country that citizens have prohibited immigration from or where immigration is restricted and certain people are listed as prohibited. A blockchain of prohibited SINs {P-SINs} would need to be checked against banned individuals.
    - The system could request a human check for verification that a person is not from a prohibited country or on a banned list in the event that the country does not use the same system.

3) **Temporary Visitors and Worker Visas**: Lastly, how are temporary non-citizens allowed into the system?

These could be workers validated by their own tracking blockchain and gifted a temporary VID (Visitor ID) that is attached to their SIN, which is then timestamped in the visitor's blockchain and issued a countdown that indicates how long the person is allowed to stay. Once that timestamp is passed, an officer of the law would be able to check their cryptographic timestamp against the current date and decide whether they are legally allowed to continue their stay or if they must leave or be deported.

Conclusion

Both decentralized application platforms and distributed direct democracies present numerous complex and multifaceted design problems. They require novel solutions to existing problems and solutions to previously unsolved problems. This white paper is by no means an exhaustive exploration of the concept. It is merely a starting point for considering how people would begin to build such a technology platform.

When author Daniel Jeffries conceived of the idea of a digital, distributed direct democracy twenty years ago on his friend's balcony in New York, *none* of the technology to make it a reality existed. Open Source as we know it today was in its infancy. None of the tools for massive, distributed development, such as Git, existed. The Internet was a shadow of what it is today, with only about 36 million users, or 0.9% of the world's population. Smartphones were a glimmer in Steve Jobs' eye.

Even as late as 2008, much of the technology to make this theoretically possible had not been developed. This changed in 2009, when the (possibly still unknown) creator of Bitcoin provided the foundation of blockchain technology, which in turn created the possibility of designing DAPPS. However, even with the breakthrough of blockchains, which are essentially trust machines, many concepts needed to make Cicada a reality are still in the early stages of their development. Even the ones that are furthest along have not fully crystallized around best practices.

Even worse, DAPPS lack many of the capabilities we've come to expect from client/server or centralized cloud-based applications (which are really nothing but a client/server model extended to hyperscale).

Other technologies, such as revocable biometrics and biocryptics, exist in research paper form only and will need to be implemented through trial and error. Many of these ideas may require significant adaptation or prove entirely unworkable once they are committed to code and subject to the friction and realities of everyday life.

Some technologies such as AI are just starting to evolve. Who knows what breakthroughs AI research will bring? The field is currently going through a Cambrian explosion of development, receiving massive amounts of investment that will only accelerate this trajectory.

Will CPU- and memory-intensive AI calculations ever be able to run on a distributed platform? Probably, but who can say when? Certainly the project will benefit from the rapid developments in this sphere in the coming years.

Lastly, it's also likely that some of the technology necessary to create this solution simply does not currently exist in any form, and will require dependent technology to be developed first. For example, nobody could possibly have conceived of the internet until computers were invented.

Nevertheless, after considering all of the research and working through the thought experiment of what it would take to create a technology like this, the authors have concluded that the time to begin the Cicada project is now. It offers the possibility of a better ID system, a more secure and scalable application platform, a true universal basic income, as well as providing countries with little to no successful governance a viable way to live and grow.

And that can change the world.