A Trustless Decentralized Bandwidth Incentive

Mark Nadal, Dr. Amber Cazzell
mark@gunDB.io, nadal@stanford.edu

Abstract. This paper proposes a protocol that allows for data to be sent through an untrusted server. Because trustless systems remove traditional revenue generating mechanisms, disinterested servers must be incentivized to relay data, especially if that data is encrypted. We propose how to reward those servers for transmitting data, yet simultaneously discourage bad actors from exploiting the decentralized system.

1. Introduction

As more people use end-to-end encryption (E2EE) to keep their communications private, it becomes increasingly difficult for servers to mine and sell people's data in exchange for providing free bandwidth. Our key insight is to incentivize traditional centralized servers to act as if they are decentralized peers. This aligns the interest of all parties, but makes it difficult for users to know which servers they can trust. Servers would not be needed at all if a direct peer-to-peer (P2P) connection could be established, but because that is not always possible, like with a phone moving through a mobile network, servers still provide some utility. The goal then is to design a protocol that removes the need for trust, such that a direct connection can be emulated through those servers, but still compensates servers for their efforts.

Contents

2. Organization

Incentive based routing has a rich history[1][2][3][4][5][6][7][8][9], which an entire paper could be dedicated to summarizing. We leave it to the reader to peruse the academic literature on the subject, and will instead focus this paper on explaining the details of our proposed bandwidth incentive protocol.

2.1. Sections 3-5.

We discuss the **motive** behind our research and the **problem** it solves given certain protocol **definitions**, **prerequisites**, and **implementation**, along with a concrete **example** to help conceptualize the network.

2.2. Sections 6-8.

We address the incentive mechanism for driving **adoption**, the end **goal** it attempts to achieve, and the **economic properties** it should satisfy. This is distinct from why these **methods** were chosen, which is discussed in its own paper[10] that outlines the psychological underpinnings of behavioral economics.

2.3 Sections 9-10.

We explore the **foundation** of the network, why it has the **transactional properties** that it does, and how it solves for the most basic **counterfeit attack**. This will set a baseline for understanding more complex problems and attacks that the network mitigates.

2.4 Sections 11-13.

We explain the **waste** that is generated in the network as a result of securing its desired properties, and how we can make the network **scalable through mining** that waste. However, this poses its own side effects, such as **collusion and Sybil attacks**.

2.5 Sections 14-16.

Finally, we show how the design of the incentive system **mitigates those thwarts** based on the self-balancing economic properties. The known remaining **limitations** are discussed, and the **conclusion** is presented.

3. Motive

Historically, users have had no choice but to rely upon centralized servers to act on their behalf. However, recent development in decentralized logic[11] has changed this, making E2EE more viable. We wanted a way to bridge this transition such that all parties could benefit.

3.1. Data Mining.

Previously, there was no way for a user to know whether their data on a server would be kept private or not, other than trust. As a result, it became more competitive to mine and sell users' data in exchange for providing free services. While a user may not want their data sold, it at least removes the problem of any uncertainty.

3.2. End-to-End Encryption.

Users have turned to using E2EE to guarantee that their data has been kept private, regardless of whether they are paying for a server or not. This however, makes it difficult if not impossible for a server to profit, potentially decreasing the competitive availability of servers. We foresee this as a problem, users will either not have access to servers, or will have to be skilled and wealthy enough[10] to run their own.

3.3. Token Mining.

If we can incentivize servers to mine tokens from bandwidth instead of mining users' data, then their would be a competitive demand for servers again. One that aligns the revenue generating interest of a server with the privacy interests of the user. Our key contribution is in designing the algorithms that will power the economic incentive, as described in the next sections.

4. Proof of Propagation

Let us define a server as simply another person's computer that facilitates the communication between two or more other computers. These are all peers in a decentralized network, such that there is a peer who sends data ("send peer"), some peers that propagate ("server peers") the data, and a receiving peer ("end peer") whom the data was intended for. Our system has the following cryptographic prerequisites:

4.1. Asymmetric Cryptography.

Let p be a private key, $p`$ its public key, where $\sigma$ encrypts and $\sigma`$ decrypts. Such as with RSA, ECDSA, EdDSA, or potentially BLS, Schnorr, and others.

4.2. Checksum Hash.

Let $\Phi`$ be a hashing function, thus $\sigma(\Phi`(d), p) = s$ signs d data, $\sigma`(s, p`) = \Phi`(d)$ verifies. SHA-256 or similar can be used for the checksum.

4.3. Smart Contracts.

The sending peer signs a b bid of trivial tokens for bandwidth with $s_n = \sigma(\Phi`(b_n), p)$. If a server wants to enter into a contract, they add their rate to the bid, sign it, and send it back.

4.4. Merkle Trees.

The contracts are a Merkle tree of $\mu(p) = \sigma(\Phi`(\Phi`(b_1 + s_1) + \ldots + \Phi`(b_n + s_n)), p)$, which establishes the fact that the sending peer has signed the tokens over to the server.

4.5. Blockchain Ledger.

Contracts can be committed to a blockchain by either parties, such that anyone can verify $\sigma`(\mu(p_a), p_a`) = \sigma`(\mu(p_b), p_b`)$ as being equivalent.

4.6. Merkle Hashes.

Propagating peers check if b is owned by p, and can mine Merkle hashes of the contract between them to create new token coins.

4.7. Verified Messages.

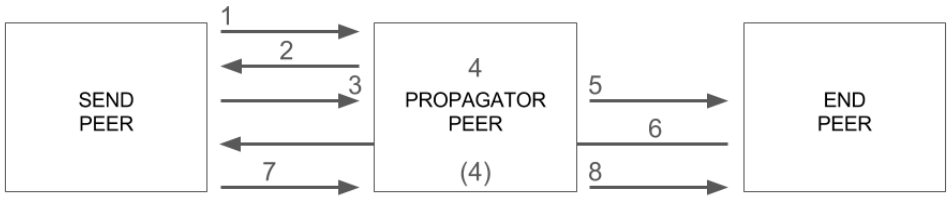
The end peer sends a signed acknowledgement, $a = \sigma(\Phi`(m), p)$, of an m message or messages back to the sender.

This is a *Proof of Propagation* (PoP) to the sending peer that data was transmitted by a server. These cryptographic proofs are necessary to establish the authenticity of bandwidth served in a decentralized system, which allows our system to be trustless.

5. Picture of Propagation

Let us visualize a concrete example of how the system works, by going through each step that happens in the network.

5.1. Figure.



5.2. Steps.

1. A laptop, as a sending peer, makes a small token bid to some servers.
2. A server adds their rate to the bid, then signs and sends it back.
3. The laptop chooses one or many servers, based on competitive rates.
4. At any point, either parties can commit this to a blockchain.
5. The server verifies that bidded tokens are in a blockchain during (6).
6. Data is sent to the end peer, like a phone, that signs an ack back

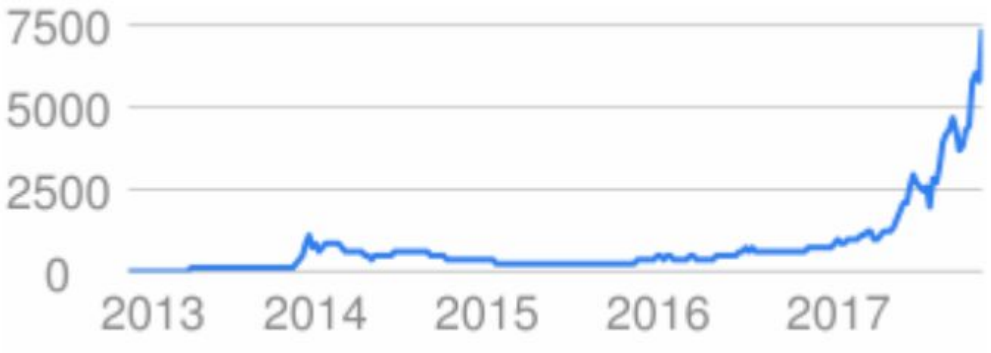
7. If (6) is proved, laptop pays for more data. If (6) fails, abandon bid.
8. If (5) is proved, server signs upped bids, and relays more data.
9. Repeat steps (4) and above.

6. Adoption

A useful incentive for driving adoption would be to reward earlier adopters with coins that increase in worth more than latecomers' coins, as they join in and use the system. We explore some real world analogs below that describe the intuition behind the mechanics in our model.

6.1. Figure.

Bitcoin's growth is an example of how scarcity alone can increase its overall worth.



We propose improving this by providing incentives that reward earlier adopters more than later ones.

6.2. Wear & Tear.

Classic automobiles exhibit similar properties to our system. If ten people own a car, and nine use them extensively over a decade, then the one person who kept their car in a brand new condition will be able to sell it at a higher price. Likewise, the worth of a coin should increase as others coins collect wear and tear.

6.3. Scarcity.

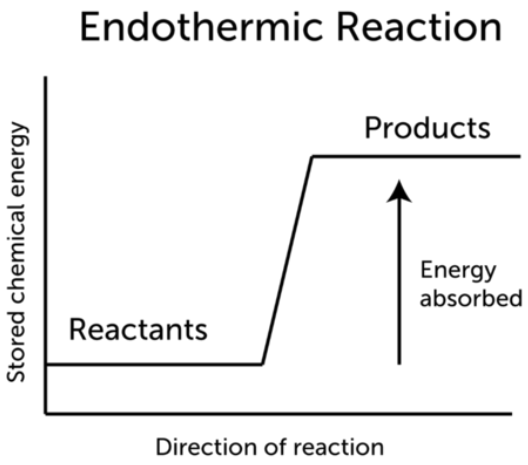
Scarcity is an easy mechanic to simulate. If we limit the number of initial coins available and algorithmically stipulate that each new generation of mined coins is worth less than the previous generation, then there is a high reward for those who join early and mine or own the most scarce coins.

7. Saturation

Unbounded growth would result in coins having no basis for their value, which could destabilize the economy. Instead, it would be ideal if the value of a coin would increase until it hit some saturating point. This would allow the economy to eventually become stable, after going through a fast initial growth.

7.1. Figure.

Endothermic reactions in chemistry are an example of a naturally occurring bounded growth curve.



Source: CK12.

7.2. Critical Mass.

The equilibrium of growth versus the saturating point starts when the number of potential new users is less than the number of current users. This is where critical mass is reached, and should be when we predict to see a diminishing return on the value of a coin compared to its prior gains.

7.3. Competition.

This is also the point where enough servers are in place to supply for the demand of the network, particularly because they are incentivized to mine new tokens before critical mass hits. Competition will then drive the token value of bandwidth to match the raw real world cost of physically sending bits over a wire or wirelessly through airwaves. This is the saturation point where the economy becomes stable.

7.4. Post-Scarcity.

We conjecture that the critical mass of the network will make it feasible for the token system to run on local routers, making internet access ubiquitous and cheap. Running it at

the physical layer will only be possible when the digital layer has become prevalent. We explore the implications of a post-scarcity economy for bandwidth in another paper[11], where we argue that data access is a human right.

8. Economics

Adoption, saturation, and proof of propagation must be possible even in the worse case of economic conditions, where people behave irrationally and greedily. This protects our system against failures, yet equips it for even better success if people do act rationally.

8.1. Greed.

Collecting more coins can only be done through mining or purchasing others' coin. Because others may be greedy, and only want to sell for above market prices, mining becomes a more cost efficient route. However, mining new coins depends upon providing bandwidth, which is itself a utility to others.

8.2. Selfishness.

Because every generation of newly mined coins is worth less than its parent's generation, the best way to accumulate wealth is to always be an earlier adopter than someone else. This incentivizes every new user to directly or indirectly evangelize the system, to maximize their own selfish opportunity. But it also increases the value of the network as a whole, by Metcalfe's Law.

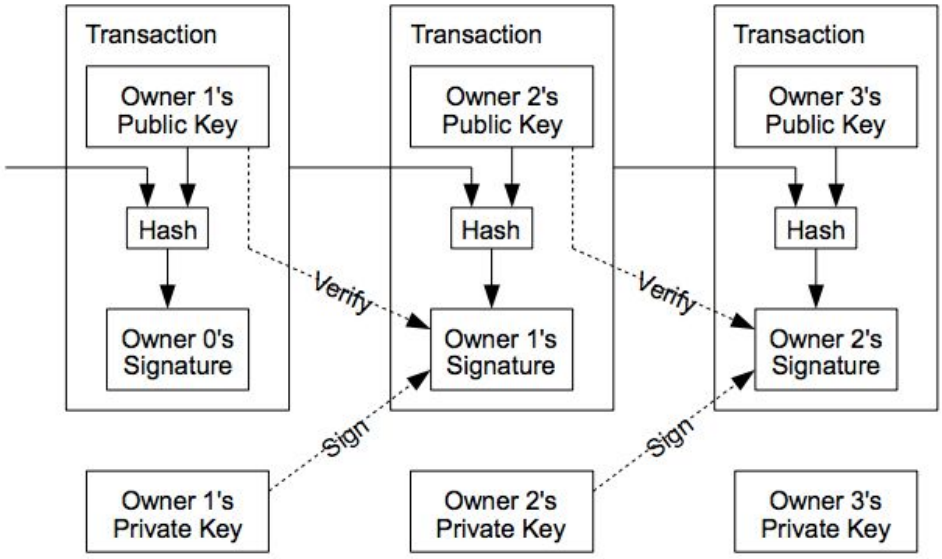
8.4. Fault Tolerance.

The more peers that are in the network, the more fault tolerant the network becomes against technical as well as behavioral failures. If there are too few servers, there is high reward for being one of the first. If there are servers charging too much, any peer has economic incentive to compete, and nothing to lose.

9. Transactions

Wear and tear of a coin is algorithmically tied to the overhead of transactions on a coin. Because there is no central authority, we rely upon a cryptographic audit trail that is immutably appended to the coin in the form of receipts. These receipts detail the change of public key ownership, which can be verified all the way back to the beginning of the economy.

9.1. Figure.



Source: Bitcoin.

9.2. Genesis Block.

The genesis block, known as G, is an authoritative set of data included in the code. It lists what coins are owned by what public keys, and has a well publicized checksum $G` = \Phi`(G)$ for peers to easily verify.

9.3. Signature Chain.

Take some public key $p_a`$ of its $p_a$ private key that is a known owner of an initial $c_{z0}$ coin in G. It can sign ownership over to $p_b$ by adding $owner = \sigma(p_b`, p_a)$ to the coin's ledger.

10. Preventing Counterfeit

The most basic attack on a currency, physical or digital, is counterfeited coins. Cryptographic primitives, such as verifying cryptographic primitives, prevents such attacks.

10.1. Verification.

Using the $p_a`$ from the G, we verify that $\sigma`(p_b`, p_a`) = p_b`$ is equal. We repeat $\sigma`(p_n, p_{n-1}`)$ down the signature chain until we get the last $p_n`$ which should match the public key that is claims ownership. If the coin cannot be traced from G, then it is a counterfeit.

10.2. Double Spending.

Each coin is technically its own blockchain, but double spending checks are managed by a separate blockchain, such as by being an ERC20 compatible token, or through some other method.

11. Network Waste

Because the signature chain must be transferred with a coin in order to verify ownership, there is overhead waste in the network. We address reducing this waste later through mining (Section 12).

11.1. Overhead.

The overhead of signatures subtract value from the coin. For instance, if a coin is worth 10KB of bandwidth, and the signature chain grows to 1KB in size, the coin only has 9KB of usable data transfer.

11.2. Worth.

Therefore, a coin with a longer signature chain is worth less than a coin with a shorter chain. This is the "wear & tear" on a coin, and may seem like an undesirable property, but it is a necessary mechanism to disincentivize fraudulent transactions (Section 13).

12. Mining

We propose taking the signature chains and hashing them to produce smaller tokens that will be more efficient to transfer on the network. These are new coins that get circulated into the network.

12.1. Hashing.

Only the peer that owns the coin can mine the signature chain, done via $\Phi`(chain) = coin$. The new coin must also link to its parent coin for verification purposes.

12.2. Circulation.

Other peers still have to do the work of verifying the coin, but the need only do this once. Then they can construct a second generation block, G_2, from the genesis block, and so on through G_n.

12.3. Scalable.

These coins are worth less due to the initial work involved to make sure they are not counterfeit, but they decrease network waste which helps keep the network scalable as a whole.

13. Collusion

If new coins can be mined for transacting bandwidth, colluders or fake accounts may attempt to trade bandwidth in order to reap reward. We discuss below why this should not be a problem.

13.1. Sybil Attacks.

Trading a coin back and forth between a bunch of fake accounts would rapidly depreciate the value of that coin, as outlined in Section 11. Because the value of any mined coin is calculated relative to its parent, the attackers would have a resulting net loss, not gain. Thus disincentivizing collusion.

13.2. End Peers.

There are no “exit nodes” to manipulate the system, and no peer can force another peer to send data. Even if an end peer colludes with a server, it still has to prove that it received data or else the sending peer would abandon the bid. The only interesting attack would require stealing the end peer’s private key.

13.3. Price Hikes.

Servers could collude to hike prices, and may be successful at it. But because servers are trustless and have no special authority, anybody else would have an economic incentive to deploy competitive servers.

14. Self-Balancing

Cryptoeconomics is about making bad behavior prohibitively expensive, not impossible. In practice, our system may exhibit many weaknesses, like a lack of initial servers or users, but those weaknesses are designed to incentivize high reward for those who fill them first.

14.1. Contract Violation.

For instance, if a server always accepted a token bid but never fulfilled its contract it could fraud peers out of money. But because the initial Proof of Propagation may only be 10bytes of test data, they would need to scam a million peers to make even 1 USD cent at

current rates. Their IP address would be blacklisted quickly, making it more expensive to deploy a new IP address than just being honest.

14.2. Incremental Payment.

A server may instead scam peers mid session, rather than at the beginning. Payments happen in incremental chunks though, so a sending peer would need to be streaming out 25 4K videos in parallel to get close to paying 1 USD cent at current rates up front for 200MB of bandwidth in the amount of time it takes to make a transaction. In this case, a server would profit more by continuing than scamming.

14.3. Two Dimensional Blockchains.

The worth of every coin is calculated by its age of generation and signature length as compared to the running average mode of tokens. For instance, a G_0 token of L_0 length is always the most valuable because it is the fastest and least work to verify. If on average tokens are at $G_{100}L_1$ then even a $G_{99}L_1$ may be too risky, but a $G_{67}L_{20}$ would be worth more than a $G_{42}L_{9792}$ token. It is an optimization tradeoff.

14.4. Marketplace Demand.

A server may lose a deal to other servers if it takes too long to calculate rates or if it charges too high of a price. As a result, there is an incentive for a server to pre-compute blockchain rates to provide faster responses.

15. Limitations

Every system has its tradeoffs, based on the goals of the system. We did not design the system to solve every problem, but it should be complementary to other solutions that address the limitations of ours.

15.1. Protocol Dependency.

Every peer must run the protocol, including the end peers.

15.2. Pseudo Anonymous.

Anonymity depends on whether a user leaks identifying information about their wallet. Additionally, the protocol works even with unencrypted data, so while it incentivizes privacy preserving capabilities, it does not enforce them.

15.3. Online Presence.

If a peer is not online, connectivity cannot be established. Our design can work well with other systems, such as remote storage, that may have their own incentive mechanisms for asynchronous exchange.

16. Conclusion

Because bandwidth has real physical cost, it would be useful to align revenue generating interests with privacy preserving mechanisms. Enabling propagating peers to mine tokens instead of mining data may be one way to solve for that, but will only work if there are strong economic incentives. We paired an improved early adopter reward with a mechanism for counteracting abuse, such that the system could have self-driving growth until the network reached critical mass, while still being scalable by mining its own waste. At this point the economy would stabilize around the physical cost of bandwidth, but have a competitive decentralized marketplace of trustless servers to supply for the demand. Our hope is that the success of the network would then enable routers to run the protocol, which would enable cheap ubiquitous bandwidth for even the most disadvantaged people. Our goal is that data access should be a human right for all.

A Decentralized Data Synchronization Protocol

Mark Nadal, Dr. Amber Cazzell
mark@gunDB.io, nadal@stanford.edu

Abstract. People come together in homes to share meals, to rest, do work, and have fun. The internet has let us extend our homes into global a community. But there are a unique set of problems that happen when we use electronically connected networks. First, there are the technical problems of creating live shared digital spaces, and doing it at scale across the world when physics stops information from moving fast enough[*]. Second, there is the social and political problems of needing an outside network that is no longer within the ownership our own home, and what that means for our privacy. We propose a solution for both of these problems by using an incentivized peer-to-peer network for sending end-to-end cryptographically secured data that will converge in the same way with any other conflicting information, as it would on any other peer in the network. By using a deterministic hybrid vector-timestamp lexicographical CRDT clock[*], we can simulate digital spaces that approximate physical ones while still protecting the privacy of individuals by not requiring a trusted third party host. This data synchronization protocol is what would allow for a decentralized digital infrastructure.

1. Introduction

By quoting Satoshi Nakamoto's introduction to the Bitcoin paper[*] with slight modification, we can highlight the problem: "*Collaboration* on the Internet has come to rely almost exclusively on *social networks run by* trusted third parties to *facilitate* electronic *productivity*." While the system works well enough for most communication, it still suffers from the inherent weakness in trust based models. The proven violation of privacy by governments, as disclosed by Edward Snowden[*], combined with the corporate incentive to propagandize alternate truth or discriminate against unprotected minorities through the censorship of freedom of speech[*], has created a demand for encrypted communication and cryptographically verifiable authorship. Bitcoin's algorithmic breakthrough and empirically observed adoption in finance[*] is evidence that such systems not only work, but are the inevitable economic models of the future. It is not this paper's place to make judgements about the "goodness" or "badness" of any political or social model, but it is in the interest of science to explore evolved systems, like Bitcoin or the one proposed in this paper, that can out compete existing solutions.

1.1. Motive.

What is needed is a data synchronization protocol based on deterministic convergence instead of a Single Source of Truth, allowing any two or more consenting parties to exchange data directly without the need for a trusted third party. The impacts of this is

that users could have truly private social networks, games, and collaborative productivity tools.

1.2. Applications.

Self driving cars could coordinate around blind corners and intersections, while smart homes could bid for their neighbor's solar panel energy, and robotic manufacturing could become offline and autonomous. Users would have full ownership of their data regardless of what devices they accessed it from. Data Scientists could design and publish distributed machine learning algorithms to an "App Store for ML/AI" that are cryptographically privacy preserving. This would let users run "AI Apps" against all their data, aggregating analytics over their phone, car, and computers to extrapolate sleep patterns, psychological well being, and guide self directed learning and improvement.

1.3. Utility.

Organizations could deploy these algorithms across a cluster that they own of a million IoT devices for autonomous self optimization or bidding out idle resources. Users could directly exchange specific aspects of their data, like their age or location, for entertainment such that advertisements are not needed, or scientific studies could ask users to donate their data for the greater common good.

2. Concurrency

Satoshi Nakamoto solved for the Double-Spending problem[*], which is one of the most difficult potential flaws in any economic system. There is an analog in computer science called the Split-Brain failure[*], which is one of worst problems within the general research field of data consistency.

2.1. Difficulty.

Data concurrency as a whole is known to be so challenging that there is a famous phrase making light of the subject, "There are only two hard things in Computer Science: cache invalidation and naming things"[*]. With the introduction and formalization of the CAP Theorem[*], scientists have been able to classify the Split-Brain problem to be well defined as either operating in optimistic mode or pessimistic mode[*].

2.2. History.

Industry use has historically and majoritively chosen centralization as a solution to the pessimistic mode in order to avoid having to solve for the harder optimistic mode. We propose a peer-to-peer solution for the harder mode of the Split-Brain failure by using a *consensus by determinism* mechanism alike the *proof-of-work* model in Bitcoin. Our solution also resolves conflicts for concurrency in general by using a simple push based cache invalidation method.

2.3. Formalization.

Given the following definitions:

- S is a member of a countable set, representing the current state of a Turing machine.
- s is a $\mathbb{R}$ real number that represents the state of s', a discrete unit of data on the Turing tape.
- We can define the function Δ delta, that takes another data state, x of $x`$, and outputs a v vector.
- The φ phi f function is the lexicographic value of the data as a cardinal number, a sum.
- The λ lambda function $f(y) = y$ writes $x`$, an atomic unit of data, to the Turing tape at x state.
- Because mathematics has operations that are not well defined in arithmetic, define them as 0.

We propose the following open-closed boundary function using piecewise notation:

$$
\begin{gathered}
\Delta(x) = \{ \\
\lambda(x - S) \; ; \; S < x, \\
\lambda(0) \; ; \; S >= x > s, \\
0 \; ; \; \varphi(s`) >= \varphi(x`), \\
\lambda(x - S - (\varphi(s`) - \varphi(x`)) / \varphi(x`)) \; ; \; \varphi(s`) < \varphi(x`), \\
\lambda(x - S) \; ; \; s > x
\end{gathered}
$$

2.3.1. Consensus by Determinism.

In simple words, it tells you the machine's distance from a change, given any two pieces of data and their state. This vector can independently be calculated by any machine at any point in time and produce accurate results without any further coordination or communication. That result gives us a guarantee that both machines will choose the same data in case of a conflict, this is what we call *consensus by determinism*.

2.3.2 Example Problem.

For example, suppose two machines, Alice and Bob, share a common state and they both send data at approximately the same Newtonian time with some latency in between. If the machines were to naively apply any update they received as being the "most recent"[*], they would then update themselves to the other's incoming data and thus be out of sync ($A \rightleftarrows B$, $B \neq A$).

2.3.3. Example Solution.

Instead, with our solution, one machine will compute a vector with 0 magnitude while the other a negative vector; this instructs one machine to apply the update as an immediate change, while the other may only apply the change to historical data, not the current state.

2.3.4. Commutative Operation.

This inverse calculation forms a commutative operation[*] that results in the two machines now being in sync ($A \rightleftarrows B$, $B = B$), guaranteeing strong eventual consistency[*].

2.3.5. Composable.

Other solutions, such as timestamps, have similar guarantees but are vulnerable to attack. We prove later that our solution is immune to such an attack. Additionally, our solution is generalizable compared to other CRDTs, such that other CRDTs can be implemented on top of ours.

2.4. Attacks.

We consider the scenario of an attacker trying to gain an unfair advantage of their data being preferred over any other peers' in the network. As a primer, let us first analyze the flaw of a naive solution for synchronization, that of timestamps[*], as an example of how an attack could work.

2.4.1. Timestamp Vulnerabilities.

Let us define the following:

$$t(x) = \{ 0 ; x < s, x ; x >= s$$

Note that this does not actually address conflict resolution, as noncommutative data can form if both Alice and Bob write data at the exact same t time. But that is far less problematic than the exploit, say a third party peer, Mallory, wants to gain an unfair advantage over the other peers, all she must do is the following:

$$Mx = s + 1$$

Such that Mal's timestamp is always larger than Alice's or Bob's, her update will always cause $t(Mx)$ to never be 0, such that non-0 output instructs a machine to apply an update, while 0 does not. Further, because t has no upper bound, we prove there is an infinitely large attack vector with the following:

$$\lim s \rightarrow \infty \; t(Mx) = \infty$$

2.4.2. Timestamp Solution.

Now compare this to our solution with the following limit:

$$\lim s \rightarrow \infty \; \Delta(Mx) = S$$

Our boundary function has an upper limit, preventing any immediate gain that Mal or other attackers could exploit. More specifically, if an attacker wanted to achieve S, they would need to get infinitely close to S. There may be no common S, so let us assume the worst case, that S is Newtonian as follows:

$$\lim s \rightarrow S^{-} \; \Delta(Mx) = 0$$

However, other peers that are honest are also operating within $S >= x > s$, causing Ax and Bx to have limits of 0, making the difference between honest and malicious peers to be ($Mx - Ax = 0$, $Mx - Bx = 0$). In effect, we have forced malicious peers to be honest if they want their updates applied immediately.

2.4.3. Out of Bounds.

But what about attacks outside of the operating bounds? Consider the scenario where an attacker always wants their data to be the next value, where they approach S from the right. Likewise, what if they want their data to be the last value, such that they always have final say? We can calculate the following limits:

$$\lim x \rightarrow S^{+} \lambda(x - S) = 0$$
$$\lim x \rightarrow \infty \lambda(x - S) = \infty$$

What this means is that if an attacker wants to have the next value, they need to be infinitely close to S which also converges to 0, giving them no advantage over any other peer. Same with wanting to be the last data, their machine would have to reach infinity to give them a gain over any other attacker.

2.4.4. In Bound.

Now consider practical attacks, bound between $\infty > x > S$, we can make the following two observations:

1. Honest peers' calculation of the vector ($v = Mx - S$) gives them v distance or time to preempt.
2. Any dishonest peer has v distance or time to also attack, competing with Mal's attempts.

2.4.4.1. Distance of Honesty.

The consequence of (1) is that it gives honest peers a magnitude on how much Mal is lying, such that it could arbitrarily decide, without disclosing a threshold to Mal, to reject Mal's updates. Having this λ write be delayed by v time, as well as the acknowledgment of the write, leaves Mal blind to whether her attack worked. Mal would have to sit, wait, and retry until she received a favorable acknowledgement, but given that it would take v time, she now has no advantage over other honest peers who have also arrived at the $S + v$ state.

Or if peers did disclose their threshold, they could use the v time to calculate a shared S upper boundary, which given the *consensus by inertial mass* (see Appendix) would let the peers reject any x above S. The only way for an attacker to win is if they have the largest cluster, which if they did, would mean they are either attacking themselves or that they are alone because they have been isolated from other networks. Either way disincentives the attacker.

2.4.4.2. Malicious Absurdity.

Meanwhile, (2) has the impact of reducing the possible reward from cheating a network, because in the unlikely situation it worked, *consensus by determinism* would favor lexical values infinitely less beneficial to any attacker. Each attacker would need a $\varphi(Mx`) = \varphi(s`) + 1$ than the other, but this itself has two problems.

First, if the data they are trying to attack has semantic properties, like the name of the owner of a coin, then there is no public key of $\lim s` \to \infty \varphi(s`)$ that corresponds to a cryptographically secure private key. That is to say, if there was a matching public key derivable from some private key, that private key would have well known and anomalous features, giving anybody, not just the attacker, access to that coin, which loses the reward of such an attack.

Or second, if the data has no semantic properties, than an attack of $\lim s` \to \infty \varphi(s`)$ would cause the attackers to have to generate infinitely larger values than the other, which would cost them both latency and physical energy in running their CPU and saturating their network bandwidth. This has to happen within a v vector of time which may not be achievable, and it is also a v distance away from a change that any honest peer would make, which increases the space that honest peers could use to block the attackers.

2.4.4. Summary.

All of these mechanisms are prohibitive factors in crackers trying to achieve an unfair advantage. We were able to prove that for several of the proposed scenarios, they would either be equal to forcing dishonest peers to behave honestly, or would result in infinitely decreasing odds that their gain is unfair. What this does not protect against, however, is malicious attackers who attempt to DDoS or spam the system for no gain at all. We will address this later by introducing trust into the peer-to-peer network with cryptographically verifiable sources. To conclude, our proposed solution is immune to unfair attacks in trustless environments by having boundaries that all peers operate under.

2.5. Split-Brain

We consider an example of a Split-Brain failure[*] to see whether data will converge correctly after the network is restored. There is an infinite number of ways that the failure could play out, so the following example is one of the simplest possible scenarios. It only shows the correctness for this specific case, but a generalizable proof can be formulated based on the equation. We propose the following example:

1. Suppose peer Alice, A has data at state s_0 of "hello", named as n_1, and S of 50.
2. Suppose peer Bob, B has data at state s_0 of "hello", named as n_1, and S of 50.
3. Peer A and B are separated by a one way L latency of 2.
4. At S of 50, A changes n_1 to "hello world" with s_{50} and pushes it across the network.
5. At S of 52, B receives n_1 at s_{50}, and calculates $v = \Delta(s_{50})$.
6. Where $S >= s_{50} > s_0$, then $v = \lambda(0) = 0$, the update is applied immediately and acknowledged.

7. At S of 52, now both A and B have n_1 of "hello world" at s_{50}.
8. At S of 54, A receives the acknowledgement from B.
9. At S of 55 the network fails, splitting A and B from communication.
10. At S of 56, A changes n_1 to be "hello bob" with s_{56} and fails to push it across the network.
11. At S of 59, B changes n_1 to be "hello alice" with s_{59} and fails to push it across the network.
12. At S of 62, A has n_1 conflicting with B of n_1, this is inconsistent from a Newtonian view.
13. At S of 63, the network is restored, both A and B ask the network for data on n_1.
14. At S of 65, A acknowledges the ask from B, and B acknowledges the ask from A.
15. At S of 67, A receives n_1 of "hello alice" at s_{59}, and calculates $v = \Delta(s_{59})$.
16. Where $S >= s_{59} > s_{56}$, then $v = \lambda(0) = 0$, Alice writes n_1 at s_{59} to be "hello alice".
17. A at S of 67 now has n_1 of "hello alice" at s_{59}.
18. At S of 67, B receives n_1 of "hello bob" at s_{56}, and calculates $v = \Delta(s_{56})$.
19. Bob already has s_{59}, so $s_{59} > s_{56}$ gives $v = \lambda(s_{56} - S) = -11$.
20. Because B trust A, B historically writes s_{56} as "hello bob", but keeps s_{59} as the operating state.
21. Now at S of 67, both A and B have n_1 as "hello alice" at s_{59}, consistent after 67 - 59 time.

The Split-Brain is now healed, and the data has converged as expected. To programmatically simulate these failures and check their resolution in our research, we developed a distributed testing framework called PANIC[*], where we tested more extensive scenarios. This enabled us to successfully verify the correctness of our solution, and is based on the work done by Kyle Kingsbury in his Jepsen Tests[*]. In conclusion, our proposed solution for a data synchronization protocol is able to correctly resolve conflicts even in the harshest conditions, such as the optimistic mode of Split-Brain failures. This, along with our considerations, allows us to synchronize information in a peer-to-peer network without the need for a trusted third party.

3. Cryptography

Social and political problems can sometimes be solved with logical processes, but because humans do not always act in deterministic ways, there is no guarantee logical solutions will be adopted. Therefore a system that can appeal to individual psychological preferences may have a higher chance of use than ones that expects normative behavior, as behaviors in social groups change over time.

3.1. Personal Identity.

As a result, we propose a solution that is based around personal identity, that way we incentive individual adoption of our solution such that even as cultural or political

changes occur, our solution is still applicable. This allows our work to integrate into existing political climates and evolve over time based on personalized goals[*] or the collective interests of a community. In order for this to work, it has to be useful in group settings as well as in isolation, which prevents us from utilizing external services for identity management or vetting.

3.2. Complexity.

Existing work in end-to-end cryptographic security[*] allows us to build a solution that depends upon no trusted third party. However, this introduces a lot of complexity into the system that could deter end user adoption, be vulnerable to DDoS or spam attacks, or increases the technical difficulty of implementing standard features, such as handling permissions between peers collaborating together in a group. We will discuss, in less technical terms than previous sections, a solution for these problems, what ramifications they have on society and political structures as a whole, and how to incentivizes trusted peer-to-peer networks.

3.3. Implications.

Personalized identity is only possible to achieve in a secure manner by using cryptographic methods, however this comes with the following technical tradeoffs that have some serious social and political implications:

- *Anonymity*. Any system that allows for unverified personalized identity also has the tradeoff of enabling anonymity. If no third party institution is required to vett an identity, then identities can be created independently. If they can be created independently, there is no limit on how many identities can be made. This, whether desirable or not, permits disposable identities that can be used for anonymous actions.
- *Privacy*. Allowing for unverified personalized identity also enables true privacy, because that cryptographic identity can be used to encrypt data without any other third party being able to decrypt the information. This privacy permits secrecy, which whether desirable or not can be used to exploit trust and give bad actors an unfair advantage over other open identities in a network.

Anonymity and privacy are controversial subjects[*], especially with respect to centralized authorities like governments. We discuss this further in the appendix.

3.4. Methods.

- **Proof of Work**. Because cryptography is cryptic in nature, and not all humans like dealing with complexity, it would be desirable to make cryptographic identities be accessible via mental models that users are already familiar with. Therefore, we propose it is possible to make a standard username-password combo cryptographically secure by using a proof of work based password stretching mechanism, such as PBKDF2[*]. Let the f function $\Phi(\rho) = \rho + \xi$, where ξ is an unchanging random value or ‘salt’ for each ρ to prevent rainbow attacks, and a Φ` function to represent checksum work, such as SHA[*]. This Φ proof of work makes brute force attacks

impractical because the time to compute each guess is non-trivial. We will then use the proof to symmetrically cypher a more secure key for our cryptographic identity.

- **Symmetric**. We propose using some symmetric mechanism of cyphering to keep users' data private, such as AES[*]. This symmetry is expressed as $\Omega(d, p) = d`$, and $\Omega`(d`, p) = d$.
- **Asymmetric**. The proof of work itself should not be the cryptographic identity because a user may want to change their password later. Instead, we propose using some asymmetric public-private key, like ECDSA[*], as a user's identity. We symmetrically cypher the private key with the proof of work, as $\Omega(p, \Phi`(p))$, so that way it can be stored without compromising the user. This will also let the user sign in from other devices, where their username or alias is a lookup mechanism that synchronizes their encrypted account information to that device. They then must use their password to decypher the private key via the proof of work. This asymmetric identity, or keys corresponding to it, can then be used for the following:
  - We are able to symmetrically cypher data as $\Omega(d, p)$.
  - Users can send encrypted messages, where $\sigma(m, p`) = m`$, and $\sigma`(m`, p) = m$ to read.
  - They can sign or verify sources of data as $\sigma(\Phi`(d), p) = s$ and $\sigma`(s, p`) = \Phi`(d)$.
  - Peers can secretly exchange other keys or group keys.
  - Users can independently enforce permissions on shared digital objects.

Combining these three cryptographic methods together provides robust end-to-end security for users. Further, it needs no third party middleman, yet emulates comparable mental models that users may already be accustomed to. In the successful implementation that we tested, we used PBKDF2, ECDSA, and AES with 256 bit sizes. We leave the specific algorithms or curves up to implementors based on their current day hardware and known broken systems, but consensus on what to use is needed. As such, we will reference everything as either a symmetric cypher for privacy, proof of work for disincentivizing attackers, or asymmetric encryption for verification or for privacy.

3.5. Model.

Because there are no centralized servers, we must design our logic to be enforced on every peer as if they were their own server. Initially, this may be difficult to reason about compared to the traditional client-server model[*] that most developers are used to, but in our research we have developed a simple yet effective model for dealing with this. An approximate yet innacurate summary is to invert the logical dependencies in a system, such that if $X \rightarrow Y$ in a client-server model, have $Y \rightarrow X$ in a peer-to-peer model. This becomes easier if we also model all the data as a graph, and treat peer connections as an ad-hoc mesh network[*]. We will go through several specific examples to address and explain how our model works.

1. *System Data*. We propose that all system level data, such as user lists and their associated public keys, have self-referential integrity. That is, rather than the code depending on an authorized Single Source of Truth for data, the data depends upon the

code to enforce a schema. While anybody can add a new user to the user list, the value of that user cannot be arbitrary, it can only be a table of public keys. Likewise, nobody can delete or change another user's list of users or public keys, and those keys must always reference themselves. Excess accounts could be made by a cracker in a Sybil attack[*], but because unlike Bitcoin, peers in our system do not need to store the full dataset, they only need to store the subset they are interested in or trust, this will not adversely affect the performance of system data. Specifically, lookup time for an account by its public key is O(1) in computational complexity[*], not O(N) where N is the number of users, because data can be stored in a radix tree such that the lookup time is constant based on the length of the public key.

2. *Shared data*. Again, in most client-server setups, clients make changes to a single common object, and an authorized server applies the updates based on whether client has permission to do so. In a peer-to-peer configuration, there would be N copies of the object under a common name, each owned by that Nth collaborative peer, where it contains only the changes made by that peer. These copies are then merged into a single view of the object using our proposed conflict resolution algorithm. This gives us concurrency for free, despite being an unusually hard problem to solve for, while also making complex permissions possible. By inverting the logical structure from having one object modified by multiple actors, to having multiple objects that each one actor merges together, we get secure shared data. We address performance and permissions next.
3. *Shared ownership*. An unexpected property of (2) is that it branches off a "multi-worlds" universe[*] of permissions, that is, each peer can independently choose who is or is not allowed to modify the object. Alice might collaborate with Bob and Carl, while Bob only collaborates with Carl and Dave, not Alice. This could be viewed as a glitch in our design, but it does not make any honest peer vulnerable to an attack, instead, we propose that this has feature opportunities, like being able to segregate cheaters from legitimate players in an online video game[*]. Specifically, this constraint on who can collaborate, based on who is trusted, also limits the computational complexity of lookup times and merge speed. It is unlikely a group would be so large to outpace the processing speed of merging copies, but even if it did, adjacent peers could store caches of the unified view. Peers could compare Merkle hashes[*] to determine with high probability that loading more copies would be unnecessary, or if it is needed, whose to load.

From these basic models, combined with the methods described above, it should be possible to implement any traditional application in a fully decentralized manner by using our data synchronization protocol. The most important component is that these are solvable problems, compared to problems that may be unsolvable. Our approach keeps end users protected and requires no change in behavior from users, however security alone does not incentivize adoption nor does it protect the network itself from being attacked. We will address those issues in the next section. In conclusion, internet access depends upon a network that is outside the ownership of one's home, this reliance makes

privacy impractical especially in the light of third party services that act as a host. These hosts are a centralized repository that may be required to give authorities unilateral access to an entire population's data[*], or are corporations that may develop incentives to censor or sell that information to untrusted parties[*]. Cryptographic identity allows us to restore privacy by enabling peer-to-peer applications to run logic within a user's home, this cleverly avoids any social or political pressures yet still interoperate within existing infrastructure.

4. Psychology

Bad habits often have a more effective reward schedule[*] than good habits, such that what may be better for an individual may not always be more appealing. Based on our research expertise in psychology and programming, we attempt to invert this model, by providing an incentivization mechanism that would assist in retention but also make the network immune to abuse. One of the most compelling features of Bitcoin is that it rewards adopters for participating in the network, which is a psychological touch that is often missing from other technical masterpieces.

4.1. Incentives.

We propose a series of incentives that will positively reinforce learning behavior[*] based on each level and scale that the system or a user may be at. In summary, we combine an individual's need for belongingness[*] and a groupthink sense of loss aversion[*] with a viral deflationary economic system, in order to bootstrap a network that is robust and efficient by penalizing attackers with inflation, such that individual ideals[*] or collective goals can be attained. This section also reaches the limit of our reference implementations and proposes a framework for our future work, based on our and our colleague's past and present research findings in psychology.

4.2. Individual.

Despite incentives, the usefulness of initial individual adoption is key. Data synchronization still has value in the context of the self, especially as individuals increasingly have more than one device. Work done on a mobile device while a user is away from home should automatically be in sync with their work station by the time they sit down at their desk, through their local wireless network or external services, without any manual intervention. Beyond that, the following are two important capabilities not possible with centralized solutions, that could attract users:

1. *Unified Data*. First, users could regain access to their data across all their federated and centralized communication tools, such as emails, social networks, search engines, and more. Applications based on personalized identity can run on the end user's device, meaning a user can have it automatically import or export data while the user is engaging with those external services in the same way visually impaired individuals

use secondary tools to assist in navigating those services. Overall, letting users re-mine data or behaviors that they generated restores ownership back into their control. This is important because it gives users the opportunity to self-learn for their own objectives, in ways that previously only advertising agencies could tap. Psychologists could build machine learning algorithms that users could securely run against their own data to detect whether they are being manipulated by third party services, or if they have addictions to social media, online shopping, or entertainment consumption. Users could then authorize these private decentralized programs to generate customized step-by-step therapy guides that provide positive or negative reinforcement when that user participates in behaviors they do or do not want. The amount of personal data that is not currently available for its own end user to analyze is a wasted opportunity that prevents the improved wellbeing and flourishing of individuals. Our proposed solutions not only open this up, but creates an entirely new marketplace and platform for people and scientists to participate in, enabling the creative collaboration of the commons.

2. *Computational Focus*. Another benefit that an individual would get in our system, is the collective harnessing of the user's computational resources. Because decentralized applications are designed to run concurrently across multiple machines, a user's local wireless network could be used to pool their devices' processing speed together. Take for example a game running on a user's mobile device, it is impractical for one user to use multiple devices simultaneously, so it would be an improved experience if their workstation could contribute graphics calculations to the phone. This would allow for graphics not possible on that device alone. The round trip local network time could be taken into account, as we described in our considerations on concurrency control, to provide hints for what the remote machine should calculate. Often times in games there is a parallax effect which intentionally delays the responsiveness of further away objects, this can be matched with the latency between devices to improve the graphics of distant objects, which are often not prioritized in a game. This would create a richer world for users to experience, by taking their currently underutilized computational resources and automatically refocusing them based on the attention that the user gives to any one particular device.

In summary, when application logic is no longer dependent upon a client-server model, individual users immediately benefit from advanced engineering and data modeling that can unify and saturate all of their hardware and information together. This incentivizes users to adopt a peer-to-peer model in order to access better self-improvement tools and new experiences, especially with the advancements happening in augmented and virtual reality[*]. These capabilities are not limited to individuals or gaming though. In one of our experiments, our collaborators successfully harnessed multiple devices to run distributed machine learning algorithm on our data synchronization protocol[*]. The usefulness of this, however, was found to be more applicable to larger groups than individuals, which we will discuss in the next sections.

4.3. Network.

The greatest value of data synchronization comes when multiple individuals are able to collaborate together. But because these individuals are not always in the same physical space, we have no guarantee that a network exists between them to facilitate communication. As a result, distributed groups often rely upon a third party to host their work, so that way as long as individuals have access to that service, they will always be in sync. These hosts expend a lot of capital to insure that their infrastructure is reliable, often times by internally using distributed mechanisms. To prevent against abuse on their network, they control and manage the identity of their users, so that way they can restrict access to anonymous actors or abusive accounts. A decentralized system does not have these luxuries, therefore it is only as robust as the peers in its network, and is only as reliable as whatever deterrents it has from abuse. So before we discuss the incentives for expanding the network, we first propose a mechanism to protect it.

1. *Trust Based*. By default, we propose that peers do not relay data through the network unless it is from an identity that they trust. This immediately eliminates any external abuse and incentivizes individuals to form larger trust networks in order to access better infrastructure by requiring them to also provide it to those they trust. Trust can be proven by verifying the signature of the data, $\sigma`(s, p`) = \Phi`(d)$, with a $p`$ public key that a peer has previously added to their encrypted trust table.
2. *Proof of Propagation*. Having the network ban all untrusted traffic removes the point of having a common network that users could rely upon when they do not have access to their trusted peers. As a result, we propose a *proof of propagation* mechanism that lets peers exchange tokens in order to transmit data through untrusted peers. This solves the problem of spam, DDoS, and abuse, because peers would not repropagate the user's data unless that user could prove they have already propagated other tokenized data, or can prove token ownership. This does not undermine the voice of underprivileged people though, because it still costs nothing within their trusted communities, and they are able to do work even when they are disconnected from that network because our data synchronization protocol has offline-first properties.
3. *Bidding Proofs*. A source peer offers a trivial token bid to its directly connected peers, which if one accepts the offer, the propagating peer adds their rate to the bid, signs the b bid with $s_n = \sigma(\Phi`(b_n), p)$ and syncs it back to that directly connected source peer so it can compare rates. Our synchronization protocol and data model allows them to form a provably shared Merkle tree[*] of $\mu(p) = \sigma(\Phi`(\Phi`(b_1 + s_1) + \ldots + \Phi`(b_n + s_n)), p)$, such that an equality of $\sigma`(\mu(p_a), p_a`) = \sigma`(\mu(p_b), p_b`)$ can be shown, which represent the escrow process. At any point, this contract can be committed to a secure blockchain ledger, but may be aggregated with other transactions for efficiency sake. Now the source peer can transmit through the propagator and verify this worked by receiving a signed acknowledgement from some intended end peer. If this does not happen, the source peer should stop signing the Merkle tree, preventing any significant loss. If it succeeds, the source peer can up its ante by signing the Merkle bid with more

value. Ownership of tokens and checking against double spending can be verified through the blockchain, using similar mechanisms to Bitcoin.

4. *Sybil Attacks*. Because a source peer has to already have a token that exists in the blockchain, it does not permit fake or colluding peers to generate tokens out of nowhere. The only option they have is to chain longer or sign larger bids, which makes that bid slower to move in the network and onto the blockchain, thus reducing the value and efficiency of the original token to the point that other legitimate peers would not want to trade for it. Legitimate peers have an incentive to compete for earlier tokens with fewer bids weighing them down, but as those become more scarce, the economy averages out as a whole over time. Our system is therefore immune to sybil attacks[*] because such an attack would create a provably large bid or long chain for each valid token, making it disproportionate in size compared to other tokens. This increasing size will not affect the long term performance of our system though, because the average mode of tokens sizes can be calculated in the blockchain, which will let peers incrementally shrink bid history with Merkle hashes, thus forming consensus around shorter tokens while still tracking their relative value.
5. *Weighted Trust*. Finally, it would be useful to have a mixed mode, where trust is not an all or nothing decision. We therefore propose weighted trust values, this lets users give discounts to acquaintances based on how many degrees of separation they have, or some other verifiable mechanism. It also allows individuals or groups to provide full discounts to charities or underprivileged minorities that they want to protect from discrimination, in ways that can be automatically vetted through a web of trust. A concrete example is that a homeless person, whom previously had a phone but that can no longer pay for a phone service, would be able to send asynchronous messages to their family whenever another individual walked by on the street, regardless of whether that stranger stopped to help them. Ad-hoc peer-to-peer mesh networks open up opportunities for the less fortunate in ways that centralized services cannot afford, because the improved efficiency in our system enables economies of scale. Data access should be a human right, because when we extend empathy to others outside of our in-group, we create social mobility that was previously unattainable for them, and from this diversification we get new insights that compete with the status quo.

In conclusion, we proposed a *proof of propagation* mechanism to protect against abuses on the network. This required us to design a cryptographic incentivization method similar to Bitcoin's, such that malicious or colluding peers cannot take advantage of the tokens. Our data synchronization protocol is able to achieve similar properties of robustness and reliability, as those of managed third party services, but without controlling the identity of users. Our model of ad-hoc mesh networks makes it capable of interoperating within existing client-server paradigms, but also enable novel use cases that previously would have been prohibitively expensive. When personalized identities exist within a network, we suggested how individual ideals or collective goals could be attained based on collaboration within networks of varying trust. As a whole, this *proof of propagation* mechanism makes the network scalable, at any level, because it incentivizes peers to

optimize for the shortest path across a network, thus reducing physical load in order to maximize each token value, while deterring abuse or attacks on an otherwise highly resilient system without the need for a third party centralized host.

4.5. Collective.

We propose incentivization mechanisms based on psychological understanding in order to expand the network as a whole, knowing that as it grows the network will be self adapting and resilient to attack. We also address the necessary problems any novel system confronts when competing against more established alternatives.

1. *Emergent Incentive*. All the benefits of (A) also have the emergent property of being usable in a group setting, this can help enable collective goal attainment or be used to pool together larger computational resources. As such, individual users would be incentivized to spread adoption to new trusted users in order to personally gain from their resources, while also improving their collective trusted network. This is separate from individuals' natural need for belongingness[*], which by itself provides some incentive, but can be complemented by providing inclinations that elicit those feelings.
2. *Economic Deflation*. While (1) may incentivize local adoption, it would be useful to have an incentivization mechanism that globally reinforces adoption. As such, we propose, based on the empirically observed success of Bitcoin, that deflation is an effective mean for generating collective adoption of our data synchronization protocol until Metcalfe's law[*] is held. The tokens owned by earlier users would increase in value over time as laggard adoptees join the network. Late comers must "mine" tokens via *proofs of propagation* into the blockchain ledger, which causes these tokens to decrease in value relative to their earlier tokens, thus rewarding the earliest adopting users while also preventing the attack vectors described in (B4). Based on the Metcalfe variant, tokens should decay through use at an approximately logarithmic rate.
3. *Loss Aversion*. The economic fact that earlier adopters will benefit more creates a constant sense of loss aversion[*] in the decision making process of an adopter. This is necessary in bootstrapping the network, because it provides cryptographic guarantees that as long as there is forward momentum in adoption, that then adopters will yield significant returns in the system, at least until a critical mass has been formed. Once that curve has been reached, the network no longer needs a bootstrapping mechanism, as it provides a service with intrinsic value to users. This makes it a compelling choice for individuals because the economy of scale it operates at make it significantly cheaper than centralized services, while still being more reliable as a whole. Incentivized adoption is a necessary mechanic to kill monopolies that already have critical mass, because competitive utility does not alone outweigh the value large conglomerates have accumulated through Metcalfe's law.
4. *Crossing Chasms*. If users have already sunk cost into centralized services, they need their loss aversion to outweigh the cognitive dissonance[*] associated with abandoning their existing choices. Once the escalation of commitment[*] has been

toppled in users, the undermining of a monopoly happens before it can even react. This is proportional to the exponential crossover happening into the new system, but the effects are delayed by the chasm of time while individuals are using both services. In retaliation, the subverted monopoly will use their established reputation to socially or politically delegitimize the newer system, which slows the crossover as users question social norms and evaluate whether being ostracized for their choices is worthwhile. As a result, there needs to be a continuous reward schedule to reinforce their new learning behavior and a community to make them feel included in it, which is why (1) is foundational.

In summary, because our data synchronization protocol enables cryptographically secure personalized identities, our system builds upon the improved value that individuals gain when using it for themselves, such that when we combine this with incentivized network effects, we can achieve collective goals that benefit society at large which previously were unattainable through centralized systems. Human flourishing is often lacking because bad habits use more effective psychological tactics. We invert this on its head, in the same way we previously inverted the models for data, concurrency, and security. By using verifiable *proof of propagation* mechanisms, we were able to design a deflationary economic system similar to Bitcoin that is immune to abuse and drives adoption, without the need for a third party to vett identity. The most important piece is the opportunity for individual ideals and collective goals to flourish, as personalized applications can be built to empower self learning, protect underprivileged minorities from social and economic discrimination, and dissolve monopolies into thriving competitive markets.

5. Conclusion

Collaboration on the Internet has come to rely almost exclusively on social networks run by trusted third parties to facilitate electronic productivity. We seek a solution that mirrors individuals' intuition of shared physical spaces but is capable of running on a decentralized digital infrastructure, such that communication can happen even when people are not present together.

The challenge of preserving privacy in an externally controlled network that people would ordinarily have with the warmth of their own home, is a war that has already been lost at the social and political level in society. Our proposed data synchronization protocol addresses these most pressing technical and cultural problems, by cryptographically securing personalized identities that can successfully converge encrypted data even if a Split-Brain failure occurs.

To minimize network failures, we use several psychological incentivization mechanisms to create a larger digital infrastructure that is more robust and reliable than

centralized third party services, yet is still provably immune to exploits and attacks from bad actors trying to cheat the system.

Our *proof of propagation* of data through the network generates a natural deflationary value for early adopters, which gives them continuous economic reward for helping bootstrap the network until it reaches critical mass, using the same approach as Bitcoin.

The most important emergent capability of the system though, is that once the properties of privacy and protection are restored into our digital homes, we can use our individual or collective idle resources to provide underprivileged minorities with valuable services that cost nothing, because crowdsourced peer-to-peer networks enable economies of scale. Free products, free peoples, flourish prosperity.

Appendix

A1. Synchronization Considerations.

We do not guarantee strong linearizable or serializable behavior[*], meaning we prefer high availability of data and partition resilient networks in the CAP Theorem over strong global consistency, which is commonly used for financial transactions. Bitcoin provides an alternative to this with a strongly ordered blockchain, and because our solution can be used to build blockchain technologies and data structures, it is possible to balance out the lack these behaviors through alternative approaches

A1.1. Consensus by Inertial Mass.

Our system still works even if there is no shared machine state across peers, which makes it ideal for decentralized networks but has the drawback of non-Newtonian time. Applications, such as video games, can simulate Newtonian time by running a fully peer-to-peer version of Network Time Protocol (NTP)[*] that we have implemented using the following damping differential equation:

$$\tau(t) = t - ((e - (e - s)/2 + a)/2)$$

It assumes the network routing is roughly symmetric. Each variable in the equation is defined as follows:

- t is time.
- s is the start time that this peer asked the other peer for their t, separate for each peer asked.
- e is the end time when this peer received the a acknowledgement from the other peer.
- Such that we approximate their current time by halving the latency and adding it to a.
- If each peer adds half the difference between their times, we approximate Newtonian time.
- Because clock skew will drift machines apart, incrementally running this will keep them in sync.

This algorithm has a clustering effect like gravity, meaning if every machine runs this against every other machine in a network, the largest number of machines with the closest drift will generate a larger pull on other machines. We call this *consensus by inertial mass*, but it is limited in usefulness and therefore should only be used sparingly. As a result, it is complementary to our generalizable solution because it simulates a Newtonian clock that could be used in a global system, but it itself is not resilient to network partitions and therefore should not be depended upon in any application with offline behavior.

A1.2. Counting.

Double-Spending problem is particularly hard for financial transactions which are zero sum in nature, where an accountant has to track that when Alice sends money to Bob, two operations need to be done, that of adding to Bob and subtracting from Alice. Either operation might fail or be duplicated because networks are unreliable, which makes guaranteeing exactly once semantics[*] difficult if not impossible.

Having all operations be performed by some centralized accountant removes the need to solve for these problems, but because we do not use centralization and do not depend upon Newtonian time, we propose the following simple CRDT sum[*] as a solution:

$$c = \Sigma_{n \in T} \lambda`(n)$$

This is to show that other CRDTs can be implemented on top of our generalizable solution, where we define the terms as follows:

- c is the count, which should be equal to the number that any Newtonian system would produce.
- n is a unique cardinal number or identifier.
- The $\lambda`$ lambda prime function reads an atomic unit of data from the T Turing tape at n space.
- Such that we sum every piece of data, expected to be an $\mathbb{R}$ real number, to get the final c count.

All this solution does is map operations that previously happened over time in a Newtonian system to space. Now that the data is represented in space, we can safely synchronize that information with our previously proposed solution. Both machines then perform the sum independently and rederive the same count as the other machine, or as would be produced with a centralized accountant. This shows that any other system, including blockchains or CRDTs, can be synchronized with our solution.

A1.3. Idempotency.

Another way to synchronize ledgers in our system is by borrowing the mechanism of physical currency. A coin is owned by someone and has a supposedly fixed economic value, so when we exchange physical coins the only operation needed is to change the name of the owner for each coin. Changing the name is an idempotent operation[*], meaning it is not susceptible to failure during network loss or duplication, and therefore can be retried safely using at least once semantics[*] over the network.

A2. Universal Foundation of Cryptographic Applicability.

Applying cryptography necessarily comes with controversial ground. Take for example a group of terrorists, they can use encryption to keep their plans secret, avoiding discovery by law enforcement. This is detrimental to society as a whole, as it allows for bad actors to commit violence against society.

It therefore may be suggested that banning cryptographic identities would be beneficial[*], because it would then force individuals to use a third party institution for identity management. This middleman could then make the decryption keys accessible to authorities, thus disincentivizing bad actors. However, we make the following observations:

1. *Transitive Law of Criminal Inference*. Suppose some claim, whether true or false, that "if criminals hide, and you are not a criminal, then you have nothing to hide" as $H \Rightarrow C \Rightarrow T$. It then suggests the reverse, $T \Rightarrow C$ as "if you have something to hide, then you are a criminal". We will see if $H \Leftrightarrow C \Leftrightarrow T$ can be verified as true, be shown to be false, or whether it is indeterminate.
2. *Cryptography Not Exclusive*. Encrypted communication has a unique property of not being dependent on the transport mechanism underneath. As long as any two parties can first cypher their messages with an external tool, they can still 'copy and paste' their messages through a third party. These decryption keys will not be accessible to the middleman or authorities, so while their identities may be known, they cannot be proven guilty of any crime.
3. *Identity Theft as Anonymity*. If encryption is made illegal on the justification of (2), then sensitive information about honest citizens cannot be cryptographically secured. This weakness makes it trivial for criminals to commit identity theft, as the entire system then relies upon trust during transactions. A single breach by a cracker at a large institution or a secondary swipe of credentials at a local store can be used to parasite honest citizens for decades to come.
4. *Government Security Precludes Accountability*. Suppose, as a result of (3), a government makes encryption available to authorized agents, such as large retail outlet or the government itself. This will enable citizens' information to be protected from crackers and further national security, which was the original intent of the initial policy. But this has the unintended consequence of incentivizing malicious peers to become government employees in order to use cryptography.
5. *Violation of Transitive Law*. And therefore, we arrive at a contradiction of criminal inference. If the government has something to hide, it suggests criminal intent ($T \Rightarrow C$). If the government by definition cannot be criminal, then the transitive property must be false ($T \not\Rightarrow C$); as in, it does not hold to be true in all cases. So in order for it to be true, the government would have to either risk national security, or have a paradoxical double standard with an indeterminant law.

In conclusion, the existence of anonymity and privacy are either not unique to criminals, or are equally possible to achieve in any other setting. This forms what we call the *universal foundation for cryptographic applicability*, it is a powerful tool that will inevitably be used by any individual or group. We cannot measure the subjective "goodness" or "badness" of any of those actors, but we can objectively instrument the use cases for them, and leave it to evolution for other systems to catch up.

A3. User Drawbacks.

There are several drawbacks to cryptographic based identities that need to be considered, and are discussed below in more detail.

A3.1. Password Resets.

Personalized cryptographic identities do not allow for traditional password reset mechanisms, as the password or keys are not known or derivable by any third party server. To compensate for this, we propose the following alternatives:

1. *Email derivable token*. Warning, this method is insecure! After an account is created, the user is prompted to email themselves a reset token in advance, with a subject line of "Your Application Password Reset Token". Then later when a user selects a password reset upon a failed sign in attempt, they are asked to search their email for that matching subject line. That reset token can then be used to sign in by deriving their cryptographic identity via $\Omega`(d, \Phi(r))$, which will let them change their password. This approach is dangerous as it leaks a token to an uncontrolled third party, who may or may not leak their users' information to other untrusted parties.
2. *Multi-token social sign on*. Warning, collusion makes this method insecure! After an account is created, the user is nagged to link their account with two or more social networks that they already have. We first lexically sort and then concatenate the social identity tokens we receive from the services, this lets us seed a proof of work to cypher the user's identity. This means no n of $n+1$ third parties can derive the identity, but the user who has authorized access to their $n+1$ services can. However, this assumes that no other actor, like a government, has a backdoor to all those other social networks.
3. *Extensible security questions*. Warning, social engineering makes this method insecure! After an account is created, the user is asked to fill in 5 security questions. Each question must be well designed to not be guessable or have limited or obvious answers, they must be memorable and irreversible to a user, and must be information that would never be published anywhere. Each answer should be extended into a proof of work, then concatenated together to cypher the identity via $\Omega(p, \Phi(q_1) + \Phi(q_2) + \Phi(q_3) + \Phi(q_4) + \Phi(q_5))$ which we can safely store. Now when a user forgets their password, we can prompt them to answer the security questions, and then decypher their identity through $\Omega`(d, \Phi(q_1) + ... + \Phi(q_5))$ to then enable a password change.

A3.2. Web of Trust.

Another consideration is that public key distribution is the weakest link[*], as usernames are not guaranteed to be unique, so malicious peers could try to pose as an imposter. We suggest using a verified contacts list where a user needs to add friends to it, which will form a web of trust between users. That way, asymmetric operations will rely upon the user's encrypted web of trust, rather than any randomly selected public key. This will greatly simplify most of the models in our design.

A3.3. Removing Access.

Finally, groups that use a common symmetric key for cyphering will need to remove members at some point. Traditionally, this is trivial because some centralized server stops permitting reads on that data by, for the removed user. But that is not possible in a peer-to-peer setup, because reads are determined by a trustless cryptographic method, not a trusted third party server. As such, it is necessary to generate a new symmetric key, share it with all the remaining members in the group, re-cypher all the old data, and then use it moving forward. While it is a hassle to implement, it at least is a known and solvable consideration.

A4. Thought Experiment.

We propose a thought experiment as the name for our generalizable solution to data synchronization, we call it the Hypothetical Amnesia Machine. Having theoretical experiments for novel proposals can be useful, as proven by Einstein's stories of chasing light beams on a train[*] as a way to ground our conceptualization of special relativity.

Similarly, if we were to remove time from our understanding of the universe, how would we explain the perceived linear flow of events in our life? One option is to imagine that all our life's experiences, past and future, are already stored as memories in our brain. Some memories are locked away for the future, isolated from everything else we know. Other memories are interconnected, and therefore seemingly present or in the past. Our experiences are triggered when one of the memories stored in isolation comes into contact with our memories that we can access.

The thing that which controls the connections of our memories in the graph of our mind, is a machine that can induce amnesia in our brain. Suppose such a machine is crawling around, connecting and disconnecting various ideas inside of us, it would be clever enough to always cause us to forget about the existence of such an amnesia machine.

Thereby, this machine would protect and ensure its own survival, by always remaining as a fictitious fragment in our minds, a hypothetical thought experiment. Instead, it feeding our brain a constant stream of ever connected memories would please our hypothalamus, thus distracting us with experiences such that we never notice the irrational illusion of time.

This is the Hypothetical Amnesia Machine, a thought experiment on how our solution works. If any update is within the boundary function of the equation, it is connected with the current live data set being operated on. If any update is outside of the boundary, it is isolated from memory until the machine arrives at that state, where it is then reconnected into the graph of data. This allows us to process information out of order and reconstruct a deterministic view of the data, even if duplicates or conflicts occur.