Towards global consensus on trust

Aggregation of temporal PageRank trust vectors

J. Harms

TU Delft

Contents

1 Introduction

The development of the personal computer and the internet as the digital infrastructure for global connectivity drastically impacted almost every feature of our lives. Many cannot imagine life without constant global acccess to knowledge, photos, social contacts and many more things. When Tim Berners-Lee developed the World Wide Web in the 1980s he imagined it as a globally distributed source of knowledge in form of a network where each node is equal, can both contribute and consume knowledge in order to improve cooperation and sharing of information. The huge success of the internet proved his idea right but at the same time led to problems: with global connectivity the openness of the internet's network invited malicious users. Also some services offered on the web were highly requested which required a highly advanced infrastructure to handle such traffic. Reasons such as these led to the centralization of the internet and the monopolization of services.

A similar trend can be found in many aspects of modern society: global production has replaced local production, global retailers replaced local retailers and mega cities replace rural villages. The strive for efficiency, growth and success leads to entities combining forces and creating ever stronger super-entities that improve processes to become more powerful. Mostly centralization has led to widespread welfare through more jobs, more affordable products and more efficient distribution of goods and services. The same is true for the internet: facebook has connected more than 2 billion people, google allows you to find information on anything and youtube is delivering entertainment; all for free, available to anyone willing to create an account, world wide.

But not everything about centralization is advanatgeous. Companies in powerful monopoly-like market positions can misuse their importance to influence politicians to act in their favor. German automobile manufacturers for example lobby against policies for reducing emissions of cars in European Parliament or speed-limits on the German Autobahn. In the digital world, Facebook misuses their user's data and allows political propaganda to be spread to millions of people. Furthermore, centralized application are prone to attacks by hackers as the servers are a single point of failure, and one data breach leads to the exposure of millions or even billions of users. Many such examples have proven in the past that centralized applications are not safe.

Banks are another example of such centralized systems that have their client's trust but are proven untrustworthy like during the financial crisis. This led to the development of Bitcoin, the first decentralized, digital currency which works completely without central authorities and is safe against double-spending, an attack on digital financial systems in which two confilcting transactions are submitted, resulting in spending some amount of currency twice. Bitcoin uses a single global hash chain of blocks (sets) of transactions. The hash chain leads to a chronology of transactions which makes double-spending impossible. All nodes on the Bitcoin network agree on this sequence of transactions using an algorithm called Proof-of-work(PoW) which solves a hard mathematical problem, expending time and resources. The first to solve the problem can publish a new block on the chain including a new (confirmed) set of transactions. Bitcoin is very secure, completely decentralized but it's Achilles heel is scalability: with a single global chain of transactions and the current block time of 10 minutes the theoretical throughput is 7 transactions per second.

Since the advent of Bitcoin many other digital currencies have appeard and some have found almost similar praise as Bitcoin, however the scalability problem still exists with most of the currencies. Vitalik Buterin, co-founder of the second largest digital currency network Ethereum, describes this as a trilemma: of the three desireable properties decentralization, scalability and security, at most two can be attained by one blockchain system at the same time. But in order to be usable for a global currency or as a global transaction storage, scalability will be neccessary.

This master thesis is concerned with TrustChain, a blockchain system developed at the Blockchain Lab at TU Delft. TrustChain has no global consensus on all transactions which removes the bottleneck for transaction throughput. Instead of a single blockchain for all transactions each entity on the network has their own blockchain. We have shown in previous work that this system achieves horizontal scalability, so additional nodes on the network lead to additional throughput. The scalability comes at the cost of security guarantees, most prominently the double spending attack and the sybil attack. The double spending attack is easy to perform because an agent can simply publish a second conflicting transaction with another node without sharing the original transaction. An attack is called a Sybil attack when an adversary creates many fake entities on the network to obtain a large part of the voting power or resources. This attack is usually prevented through verification of the agent, for example bitcoin verifies nodes by letting them perform work. Trustchain is built in a way that these attacks are not prevented, which is too costly to be scalable, but instead are detectable. Both transactions of a double-spend attack are stored on the blockchains of the two exploited agents. By spreading the records of these transactions in the network, eventually a node will have both versions of the same transaction and will identify the attacking node.

Next to the research on Trustchain the Blockchain Lab is concerned with real-world testing of a deployed system. The platform for those tests is Tribler, an anonymized onion-routing protected bittorrent client. A common problem with the bittorrent network is that it is not inherently protected against free-riders as there is a social dilemma in sharing resources with other agents on the network. Uploading data to other nodes is costly in terms of bandwidth without any direct reward, but if noone decides to upload nobody can download and the system breaks. This dilemma is a form of what is known as the Prisoner's dilemma in classical game theory. With BarterCast our research group has devleoped a system to prevent free-riding in the bittorrent network and recently this mechanism has been improved using Trustchain as a tamper-proof way to store transaction records. This work will also use the Tribler example as context.

Our contribution will be targeted at the dissemination mechanism of transaction records on the Trustchain network. Because every node has it's own chain, each node has only a subset of the data. This is different from blockchain systems with global consensus in which all agents act on the same data. This has considerable influence on the attack defence and accounting mechanisms. While Trustchain is made in a way that makes attacks detectable this is only possible if an agent collects data from other agents on the network. Thus, the dissemination and validation of data is of very high importance for the proper functioning of thu system. This mechanism has not yet been formerly been defined and it's implications for security and scalability of the Trustchain have not been researched in depth. Specifically this work contributes:

- We formally introduce *Pairwise endorsements* – a dissemination mechanism that records information exchange and validation on-chain, making information sharing strategy-proof and information tamper-proof
- We analyze the security, scalability and correctness of the dissemination mechanism
- We provide an implementation and experiment results to show the working of the mechanism.

The rest of this report will be structured as follows. In the next chapter, the problem will be discussed in more detail.

2 Problem description

In the introduction we make a case for the decentralization of applications that handle private information or resources and argues that scalability is one of the main problems of the promising blockchain technology to make such systems a reality. Trustchain is an approach that removes the main bottleneck that restricts the scalability of the most common blockchain fabrics, namely global consensus. However the lack of agreement on a single accepted set of transactions has many implications for attack resistance and correctness guarantess of the system. This chapter introduces these implications and defines the problem that this work is supposed to tackle.

2.1. Attacks

2.1.1. Double-spend attack

One of the most challenging attacks that exist in distributed systems is the *double-spend* attack in which an adversary creates two conflicting transactions with two different agents without telling each about the other, effectively using resources twice. In centralized systems this attack is prevented by the central server which processed transactions in order and realizes that the resources were spent in the first version of the transaction. Bitcoin was the first decentralized accounting system that solved this problem without a central, trusted entity. However the mining which creates a single accepted sequence for transactions is costly in terms of time and resources. Without global consensus Trustchain (discussed in more detail in section 3) is not able to prevent the double-spend attack. Instead, the double-spend attack will be recorded and therefore made detectable. The attacker sends two conflicting transactions to two different agents and keeps one, but both partners write the conflicting blocks on their chains. If those two agents share their blocks with each other or both share their blocks with a third agent, the attack becomes detectable because the two blocks are conflicting. The prevention of this attack therefore requires dissemination of transaction data across the network and constant checking for conflicting transactions by all agents.

2.1.2. Sybil attack

During a sybil attack an adversary takes control over many entities at the same time without making this known to the network. The attacker can then use those entities to gain influence without any real cost because the controlled entities can create proof of transactions without actually performing them.

This problem is very hard to detect because controlled entities can look like real agents to external observers. In centralized systems this is often prevented by requiring multiple authentication steps, for example scanning an identity card. Also if the creation of new agents has some costs, the adversary needs to evaluate the possible advantage against the cost of creating multiple agents. In Bitcoin and other proof-of-work based cryptocurrencies the attack is avoided because the power to create a new block is proportional to computational power, so whether the computational power is spread over multiple agents or not does not matter to the voting power in the system.

For other decentralized systems the sybil attack continues to be a challenging problem. Many solutions have been proposed which analyze the topology of the network. Also an initial negative balance has been proposed by some. Specifically for the Trustchain two algorithms, namely NetFlow and Temporal PageRank. Yet, while the two algorithms allow for sybil-resistant calculation of a metric which is related to the balance of agents. Also the accuracy of the algorithms depends on the amount of data that is available, making it neccessary to share data between agents in order to better be able to estimate the probability of sybils. The sybil attack will further be discussed in chapter 4.

2.1.3. Blockwithholding attack

In decentralized systems it can be advantageous for agents to not share some information about their transactions that would otherwise render them in a weaker position. This is not possible in centralized systems because users do not keep their own data which instead is stored on the central server. Thus it is not the user's decision to share or not share information with others.

In common blockchain fabrics all information is shared with everyone and only information that is accepted by everyone is true. By removing the global consensus this guarantee is no longer intact. If user's own their data, they can decide to share it or not. Agents can claim that information was lost during transactions or that a transaction did not take place.

2.1.4. Dishonest behaviour

Some application types may require agents to act according to a specific set of rules. For example in the Tribler application, if an agent (responder) receives two requests for contribution the agent should contribute to the one agent that has contributed the most in the past as that agent deserves to be rewarded for those past contributions. Without global consensus the agent determines the "goodness" of the requesters on the basis of an unobserved information set, which is a subset of the global network information. However the agent can also decide to not stick to the rules and contribute to the lesser of the two requesters. Without consensus on the information set on the basis of which the responder decides, this dishonest behaviour cannot be detected and punished by other agents.

2.2. Research question

From the above discussion it becomes clear that removing the global consensus from any blockchain farbics opens the system to many forms of attacks. The missing guarantees on information makes it hard to check the correct behaviour of other agents. This makes sharing of information and validation of transactions an essential building block of a blockchain system without global consensus. Yet, the question is how to enforce dissemination of transaction records without a trusted third party. Also which information is neccessary to distribute accross the network and how can we make sure that validation of that information is done by all nodes. Formally we can define the following research question:

How can we design a scalable, decentralized accounting system that ensures the distribution, correctness and honest usage of transaction records?

The research question entails some requirements for the system that we are trying to develop. In the following we will explain each of those in more detail.

2.2.1. Accounting system

The system we are trying to build is an accounting system. An accounting system keeps track of transactions of a resource of value between at least two parties. Accounting systems have many applications; two common examples are a banking system and a reputation system. Each entity in the accounting system has a unique identifier and from the history of the transactions recorded in the system a certain balance can be assigned to each identifier. When a new transaction is issued the balance is increased or decreased and usually some threshold is put inplace to restrict the infinite spending of resources. This implies that the order of transactions is of importance. As an example consider an entity A with the balance of 5 a minimum threshold of 0 and two transaction spending 4 units and 3 units two parties B and C, respectively. Obviously, it is not possible that both transactions are accepted. Either, A first spends 4 units on the interaction with B and cannot afford the transaction with C or the other way around. If entity A tries to submit both transactions at the exact same time, it is the task of the accounting system to create an order of two transactions and restrict the expenditure beyond the balance threshold.

2.2.2. Scalability

Accounting systems can exist in many different sizes and contexts, they do not even have to be digital for some applications. However in this work we are concerned with planet-scale accounting which even enables micro-transactions with high frequency. Therefore scalability is one of the main factors. Before the ascent of internet applications such dimensions were unheard of but in the last decade services such as Facebook, WeChat or YouTube have shown that an application can grow to have billions of users. Our ambition is to lay the theoretical and practical basis for future systems that scale to these sizes. In practice that means that the transaction throughput of the global systems needs to grow with the amount of users and that no global limit is in place that restricts further growth.

2.2.3. Decentralization

Ownership of all transaction data can, depending on the context, give the owner power, leverage and value. Furthermore, a central entity creates a target for attackers and with sufficient resources available an adversary will in the end be able to compromise the system. We see accounting systems as a part of the infrastructure that enables applications such as banking or reputation systems. No single entity should be owner of such infrastructure. That is why we are considering a decentralized solution. In the context of an internet application a centralized model assumes that one single (central) trusted entity has access to all information and all users know and connect to that single entity. In a decentralized model, we cannot assume that any other entity is trustworthy or omniscent. Instead entities are equal and communicate with each other. All users know about their own transactions and are owner of their data, with full control over whom to share them with.

2.2.4. Distribution

In a perfectly decentralized system each entity only knows about their own transactions. For an accounting system that means that each entity needs to check for themselves that they do not exceed the balance threshold. Yet, an entity's interest could be to spend as much as possbile, which makes the self-control mechanism ineffective. In the context of reputation systems, an entity's interest could be to show their good behaviour to others. In those situations a distribution mechanism needs to be put inplace because in a decentralized system we can no longer assume that information is simply available from the central entity. Perfect distribution of data would mean that each user is informed about each transaction happening on the accounting system's network. However in practice such a situation virtually impossible to uphold, especially when scaling to global high-frequency microtransactions. A balance needs to be found between the distribution of information, the scalability of the system and the storage and processing capabilities of each entity.

2.2.5. Correctness

In order to ensure the correctness of data multiple aspects need to be considered. First of all data needs to be stored in a tamper-proof manner, that is, once a transaction is accepted by all parties that transaction should not be changeable afterwards. Also the order of transactions needs to be definite, the reason for this was explained in Section 2.2.1. Finally, entities need to be able to validate the correctness of the state of the system. The distribution of data informs entities in the system about the behaviour of other agents, but without validation of that data, missing or wrong information cannot be found. This is another aspect that is often solved by a central entity that continuously analyzes the information received by users. In a decentralized system the validation has to be performed by each entity. For example entity A has a balance of 2 units but is trying to spend 3 units in a transaction to entity B. Without a central entity the only party to prevent A from transaction is entity B. B is only able to detect the invalid transaction if A has shared all it's transactions with B and if B uses some validation precedure before engaging in a new transaction. It is important to realize that validation is only possible if information is distributed.

2.2.6. Honest usage

Finally, the system should make it possible to ensure the honest behaviour of entities. To show how the previous two components are not enough to ensure this, we can continue with the example from the previous section. So even if B knows that the balance of A is insufficient to commit the transaction, both could collude and still commit the transaction. Afterwards, there is no way of knowing whether B was acting wrong on purpose or whether A did not share its information correctly.

In order to ensure correct usage of the given information it needs to be possible to distinguish good from bad behaviour. Without a central entity that knows the truth about every entity it is not straightforward to know which entity is the responsible one for a wrong transaction.

2.3. Limitations and assumptions

3 Related work

In the previous two chapters we have shown that a need exists for a decentralized accounting system in order to create a global infrastructure for secure, anonymous digital transactions that does not require control through a trusted third party. This need has been identified before and work has been performed both in the scientific community as well as the industry. In this chapter we will summarize those efforts, describe the short-comings of those approaches and define a basis for the work performed in this work.

3.1. Applications of decentralized accounting systems

The general concept of accounting is quite old as it is simply a recording of transactions between two or more parties. Before the digital age those recordings were simply written text on paper, nowadays those recordings are stored in databases. We are concerned with another type, namely decentralized accounting systems. We identified three types of applications for decentralized accounting systems: cryptocurrencies, distributed work systems and reputation systems.

3.1.1. Cryptocurrencies

In the years 2007 and 2008 the global financial crisis shattered the global economy, lead to many people loosing house and job and diminished the trust clients had in banks to keep their money safe. Politics discussed the problem and proposed to regulate the banks more but with little impact. However something else promised to change the banking world: the first white-paper for a decentralized digital currency without any need for a trusted third party, Bitcoin, was announced.

Bitcoin. Before the announcement of Bitcoin it was assumed that in order to verify the correctness of transactions between parties and prevent cheating with digital money a bank or credit card company was needed. Bitcoin proved them wrong by creating a hash-based chain of transaction blocks, a global ledger, that is shared among all users of the network. The acceptance of transactions is managed by a process called "mining" which ensures that only the majority of CPU power can publish new block. A blocks contains a fixed number of transactions and the Bitcoin network makes sure that a block is created once every 10 minutes. All mining node will execute the proof-of-work mechanism: in order to publish a block a value needs to be found that, when hashed with a certain hashing function like SHA-256, starts with a certain number of zeros. Depending on how many CPUs are active on the network the problem can be increased in difficulty by requiring more zeros at the beginning of the hashed value. Once a new block is published other nodes will validate the transactions and if they agree, will show their acceptance by working on creating the next block. This system ensures that as long as a majority of CPU power is owned by honest nodes, they will outpace the rest of the network in solving the hashing puzzle and creating valid blocks. Nodes will accept the longest chain and the transactions will be valid.

The Bitcoin approach solved many problems assuming that an honest majority exists: first and foremost the double-spending of funds is prevented because the Bitcoin blockchain creates one global order of valid transactions. Also the Sybil-attack is prevented by pairing the voting power to the available CPU power, which means Sybils can only run on real hardware, removing the advantage of fake identities. But these measures of attack prevention come at a price of efficiency. The surging price of Bitcoins especially in the year 2017 led to a surge in transactions, transaction fees and energy usage. The increasing price of Bitcoins makes mining them more profitable which means more nodes are joining the mining operation. Therefore the difficulty for the proof-of-work problem is increased, such that it takes more computing power to find a correct value. This again increases the amount energy consumed in the whole network. At the same time the number of transactions processed is a constant of the Bitcoin currency, approximately 7 transactions per second. At the time of writing the energy conusmption is at least 2.55 GW which makes it comparable to contries such as Ireland. Summarized Bitcoin was a large step towards decentralized accounting but unsolved scalability issues still prevent it from being actually useful as an infrastructure such as the one we envision.

Alternative coins and improvement measures. Bitcoin served as a first proof-of-concept for trustless digital currencies or for our purposes, a "secure" decentralized accounting system, but the shortcomings were also obvious. Once the populartiy increased, other enthousiasts, startups and incumbent companies started to create their own spin-off digital currency. Each of these so-called "alternative coins" used blockchains as a core technology to store transactions but tried to solve the scalability issues using different approaches. The discussion of all alternative coins goes beyond the scope of this chapter, therefore we will quickly introduce some of the main differences between the largest systems.

The block time is one parameter to tweak in order to increase transaction throughput. Ethereum, the second largest cryptocurrencies currently uses a block time of 15 seconds with a proof-of-work consensus. Also block size is a factor in the throughput rate, but increasing block time and size only creates a constant factor to the rate of transactions.

Ethereum is currently testing a proof-of-stake mechanism which should replace the energy intesive proof-of-work. In short this mechanism will require "minders" to put some amount of currency into a wallet in order to participate in the process. If a miner does not perform the validation of transactions correctly that "stake" will be lost for the miner. This will solve the energy consumption problem but it will not solve the overall scalability issue of the system.

Another feature in development in multiple currencies is the "Lightning network". The lightning network will allow two parties that expect to conduct multiple transactions with each other to create a "channel". Both parties store some funds in the channel and can then interact freely through this channel without needing to interact with the master network of the currency. Only the opening and netbalance at closing time will be writting to the chain while all other interactions are only recorded locally. This should increase the possible throughput significantly but due to the early stages of development the actual implications of large-scale use are not proven at the time of writing. But considering that Bitcoin has a transaction limit of 200000 transactions a day, it would still take 5000 days or 13.7 years to open one channel each for a billion people.

The IOTA project ...

Sharding ...

Conclusion

3.1.2. Distirubted work systems

In the field of distributed computing many applications include some mechanism in which a node is performing work for other nodes or the network in general. Seuken et al. call these distributed work systems. Some examples of distributed work systems are peer-to-peer file-sharing network, packet forwarding in mobile ad-hoc networks and volunteer scientific distributed computing. As our research group is mostly concerned with file-sharing networks and the concepts are similar in general we will stick to that example to discuss the latest developments.

Many different file-sharing networks have been built in the past, the most prominent being Napster, Gnutella and BitTorrent. In contrast to centralized file-sharing, in peer-to-peer systems there is no server that contains all data, but instead users share data directly, one peer downloading and one peer uploading. With no infrastructure needed, no costs and no single point of failure such a systems seems optimal. Talking in terms of distributed work systems, the act of uploading is equivalent of performing work while the act of downloading consumes work. There is, however, a social dilemma here: uploading to another node does not lead to an immediate reward for the uploading node, therefore, if we assume that bandwidth is a precious resource it is cheaper to not upload, yet if all agents on the network realize this, no agent will upload and thus no agent is able to download. The agents that do not upload any data are known as free-riders and free-rider protection in peer-to-peer file-sharing networks is a subject of ongoing research.

Accounting systems pose a possible solution to the free-riding problem. Let's first imagine a centralized accounting systems keeping track of all uploading and downloading behavior, uploading data increases the balance of agents, downloading decreases the balance. Now, the accounting system can enforce that agents keep their balance around 0, so they upload approximately as much as they download. Therefore, an accounting system can solve the free-riding problem, however as mentioned multiple times, a decentralized accounting system is hard to implement. Accounting mechanisms have first been related with this subject in the DropEdge paper, however a lot of work has been done on the very related subject of reputation systems, which will be discussed in the next section. Seuken et al. define an incentive-compatible accounting mechanism which removes any advantage for users that misreport their own contributions in the network. They present their DropEdge algorithm and show that it's possible to increase the efficiency of BitTorrent clients using accounting. A negative result of their work is that an accounting mechanism cannot prevent sybil attacks. Some short-comings of the approach is strategic manipulations of data and dissemination of data.

3.1.3. Reputation systems

One of the reasons that decentralized accounting systems are hard to create is that agents in peer-to-peer applications do not have a complete view of the network and thus also not all information of the network, at least not without a global consensus mechanism. In the file-sharing example from the previous section agents decide to upload to other agents based on some partial knowledge of the network and contributions of agents. It can be argued that an accounting mechanism cannot be correct if it acts on partial information and instead the particular balance of an agent as seen by another agent is rather a reputation. The goal is then to create trust between users in order to facilitate cooperation. Such a system will be called a reputation system.

Whether reputation systems can be called an application of accounting systems can be argued about. In general accounting systems track transactions between accounts, the full history of transactions determines the state of the network. According the framework of Mui et al. trust is the expectation of reciprocation for an agent given that agent's history of behavior. So a reputation system can act on the data of an accounting system and add additional conclusions. The previous example of agents uploading and downloading helps to understand this. An accounting system keeps track of the transactions and calculates the balance of an agent, for example +10MB, for an agent that has uploaded 10MB more than downloaded. Also it is possible to account the total uploaded and downloaded data, for example 1010MB and 1000MB respectively. A simple accounting system stops at this point, the system behaves correctly when no error has been done in calculating the balances and the data is correct. A reputation system adds another layer of interpretation to this data. The simplest reputation function only checks whether the balance is positive or not, or if the choice is between multiple agents, whose balance is the most positive. Another reputation function might weight agents with a 0 balance but 10GB of uploaded (and downloaded) data more trustworthy than an agent with 10MB positive balance but only 100MB uploaded data. Thus we can see a reputation system as a layer on top of an accounting system.

Describe some reputation systems ...

3.2. TrustChain

TrustChain was built as a system to create trust between two strangers

3.2.1. Data structure

3.2.2. Accounting mechanism

Definition of trust and reputation

3.2.3. Subjective graph

3.2.4. Consensus

4

Base model

In the previous chapter we have discussed the need for a decentralized accounting system and shown that no existing solution offers all the features needed for a globally scalable, secure system. Before we can describe the solution developed in this work it is important to first describe the model on the basis of which we can define our accounting mechanism. This chapter will therefore formally define a general model and its concepts in order to reason about them in the coming chapters.

4.1. Informal model definition

4.2. Formal model definition

The goal of our model is to record information of transactions between entities that are connected via some digital network infrastructure like the internet. The system should be application agnostic, thus not specialized for a single application context and allow the enforcement of application dependent rules. Therefore for the general model no assumptions can be made for the type of application but examples of different contexts will be given to show the applicability. The model presented in the following description is based on the definitions by Seuken et al. with some extensions.

Definition 1 (Network state model.) A *network state model* $M = \langle R, A, S \rangle$ consists of two sets and three functions.

- P, a finite set of agents
- I, a finite set of interactions
- $a : I \rightarrow P \times P$, a function mapping each interaction to the agents involved in it.
- $w : I \times P \rightarrow \mathbb{R}_{\leq 0}$, a function which describes the contribution of an agent in an interaction
- $k : I \times P \rightarrow \{true, false\}$, a function which describes whether an agent is aware of an interaction or not

The network state model defines the ingredients for a describing the actual state of the network. Almost any applications with transactions which can be described with a single numerical value and a single type of actor can be fit into such a model. For example, on the bitcoin network all public keys would make up the set R, the set of transactions I would contain the full blockchain, namely the transaction action. The function a obviously maps the bitcoin transactions to the public keys, the function w would map the interactions to the value of coins transferred. Finally, the function k is an extension of the original model defined by Seuken. We claim that in order to fully describe the state of the network we need to know for each agents of what transactions they are aware. In the Bitcoin network, or any network with global consensus mechanisms the function k always maps to $\{true\}$.

From the above definition we can simply extract the state definition of each agent.

Definition 2 (Agent state.) The state of the agent p can be fully described by the tuple $S_p = \langle I_p, a_p, w_p \rangle$ where:

- I_p, the subset of interactions that agent p is aware of, so $I_p = \{ i \in I \mid k(i,p) = \mathit{true} \}$

Definition 3 (Network state.)

Definition x (Network state observability.) Fully observable, partially observable (the complete state is theoretically observable but would require infinite work), not observable

Definition x (Network state validity.) The correctness of the state

Definition x (Application state validity.)

5 Attacks

6 Information dissemination

The strength of a reputation system largely pivots on the availability of data. Reputation is built through a history of interactions, but only those that know about the history can estimate the true nature of the agent. Also, unfair actions can only be detected if the information about those actions is widespread. In centralized reputation systems, this availability in guaranteed, as long as the central server with all data is available and not manipulated. In decentralized systems this guarantee does not exist, availability of data depends on the willingness of agents to share their private data.

Therefore our goal is to create a mechanism that gives agents an incentive to share their private data. More specifically, this dissemination mechanism should make the sharing of private data strategyproof, that is, sharing all private data should never be less advantageous than not sharing.

6.1. Pairwise auditing

agents group in pairs and assign a score(endorsement) to each other. The endorsement should increase with more data shared between the two parties. There is a maximum endorsement which can be calculated when all data is available. The score is used as another factor when determining the trustworthiness of an agent.

6.1.1. Definition

6.1.2. Incentive design

Endorsements without any endorsements, agents are seen as not trustworthy. Therefore agents need to exchange data with at least a few agents in order to become trustworthy. Endorsements should not be accepted by default but rather only be accepted

Strategy-proofness

6.1.3. Endorsement

6.2. Accusation

6.2.1. Proof

6.2.2. Untrue accusations

7 Experiments and results

7.1. Setup

7.1.1. Dataset

7.1.2. Experiments

7.2. Valid usage

More data leads to higher score

More audits, leads to higher average trust

7.3. Sybil attack

Sybil vs true agent

Detectability of lying

7.4. Accusations

8 Discussion

8.0.1. Strategy proofness

8.0.2. Attack resistance

8.0.3. Future research

Bibliography