ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Trust Is Risk:
Μία Πλατφόρμα Αποκεντρωμένης Οικονομικής Εμπιστοσύνης

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ορφέας Στέφανος Γ. Θυφρονίτης Λήτος

Διονύσης Σ. Ζήνδρος

Επιβλέπων : Αριστείδης Παγουρτζής
Αναπληρωτής Καθηγητής Ε.Μ.Π

Αθήνα, Ιανουάριος 2017

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Trust Is Risk:
Μία Πλατφόρμα Αποκεντρωμένης Οικονομικής Εμπιστοσύνης

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ορφέας Στέφανος Γ. Θυφρονίτης Λήτος

Διονύσης Σ. Ζήνδρος

Επιβλέπων : Αριστείδης Παγουρτζής
Αναπληρωτής Καθηγητής Ε.Μ.Π

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 23 Ιανουαρίου 2017.

.....	.....	.....
Αριστείδης Παγουρτζής	Δημήτριος Φωτάκης	Άγγελος Κιαγιάς
Αναπληρωτής Καθηγητής Ε.Μ.Π	Επίκουρος Καθηγητής Ε.Μ.Π	Αναπληρωτής Καθηγητής Ε.Κ.Π.Α

Αθήνα, Ιανουάριος 2017

.....
Ορφέας Στέφανος Γ. Θυφρονίτης Λήτος

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Trust Is Risk:
A Decentralized Financial Trust Platform

Orfeas Stefanos Thyfronitis Litos[1] and Dionysis Zindros[2,⋆]

[1] National Technical University of Athens
[2] National and Kapodistrian University of Athens
olitos@corelab.ntua.gr, dionyziz@di.uoa.gr

Abstract. Reputation in centralized systems uses stars and review-based trust. Such systems require manual intervention and secrecy to avoid manipulation. In autonomous and open source decentralized systems this luxury is not available. Previous peer-to-peer reputation systems do not allow for financial arguments pertaining to reputation. We propose a concrete Sybil-resilient decentralized reputation system where direct trust is defined as lines-of-credit using bitcoin's 1-of-2 multisig. We introduce a new model for bitcoin wallets in which user coins are split among trusted associates. Indirect trust is subsequently defined transitively. This enables formal game theoretic arguments pertaining to risk analysis. We prove that risk and max flows are equivalent in our model. Our system allows for concrete financial decisions on the monetary amount a pseudonymous party can be trusted with. Through algorithmic trust redistribution, the risk incurred from making a purchase from a pseudonymous party in this manner remains invariant.

Keywords: decentralized · trust · web-of-trust · bitcoin · multisig · line-of-credit · trust-as-risk · flow · reputation

⋆ Research supported by ERC project CODAMODA, project #259152

Table of Contents

List of Figures

List of Code Listings

1 Introduction

Modern online marketplaces can be roughly categorized as centralized and decentralized. Two major examples of each category are ebay and OpenBazaar. The common denominator of established online marketplaces is that the reputation of each vendor and client is either expressed in the form of stars and user-generated reviews that are viewable by the whole network, or not expressed at all inside the marketplace and instead is entirely built on word-of-mouth or other out-of-band means.

Our goal is to create a decentralized marketplace where the trust each user gives to the rest of the users is quantifiable, measurable and expressable in monetary terms. The central concept used throughout this paper is that trust is equivalent to risk, or the proposition that *Alice*'s *trust* to another user *Bob* is defined to be the *maximum sum of money* that *Alice* can lose when *Bob* is free to choose any strategy he wants. To flesh out this concept, we will use *lines of credit* as proposed by Washington Sanchez [1], not to be confused with the synonymous financial product. Joining the network will be done by explicitly entrusting a certain amount of money to another user, say *Bob*. If *Bob* has already entrusted an amount of money to a third user, *Charlie*, then we indirectly trust *Charlie* since if the latter wished to play unfairly, he could have already stolen the money entrusted to him by *Bob*. Thus we can engage in economic interaction with *Charlie*. The currency used is Bitcoin [2].

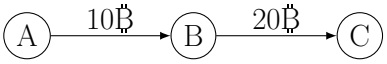

Fig.1: A trusts C 10฿

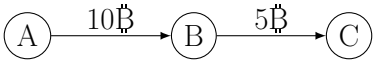

Fig.2: A trusts C 5฿

We thus propose a new kind of wallet where coins are not stored locally, but are placed in 1-of-2 multisigs, a bitcoin construction that permits any one of two pre-designated users to spend the coins contained therein [3]. We will use the notation $1/\{Alice, Bob\}$ to represent a 1-of-2 multisig that can be spent by either *Alice* or *Bob*.

Our approach changes the user experience in a subtle but drastic way. A user no more has to base her trust towards a store on stars, ratings or other dubious and non-quantifiable trust metrics. She can simply consult her wallet to decide whether the store is trustworthy and, if so, up to what value. This system works as follows: Initially *Alice* migrates her funds from P2PKH addresses in the UTXO [3] to 1-of-2 multisig addresses shared with friends she comfortably trusts. We call this direct trust. Our

system is agnostic to the means players use to determine who is trustworthy for these direct 1-of-2 deposits.

Suppose that *Alice* is viewing the item listings of vendor *Charlie*. Instead of *Charlie*'s stars, *Alice* will see a positive value that is calculated by her wallet and represents the maximum monetary value that *Alice* can safely use to complete a purchase from *Charlie*. We examine exactly how this value is calculated in Trust Flow theorem (2). This monetary value reported by our system maintains the desired security property that, if *Alice* makes this purchase, then she is exposed to no more risk than she was willing to expose herself towards her friends. We prove this result in the Risk Invariance theorem (3). Obviously it will not be safe for *Alice* to buy anything from *Charlie* or any other vendor if she has entrusted no value to any other player.

We see that in TrustIsRisk the money is not invested at the time of the purchase and directly to the vendor, but at an earlier point in time and only to parties that are trustworthy for out-of-band reasons. The fact that this system can function in a completely decentralized fashion will become clear in the following sections. We prove this result in the Sybil Resilience theorem (5).

There are several incentives for a user to join this network. First of all, she can have access to a store that is otherwise inaccessible. Moreover, two friends can formalize their mutual trust by entrusting the same amount to each other. A large company that casually subcontracts other companies to complete various tasks can express its trust towards them using this method. A government can choose to entrust its citizens with money and confront them using a corresponding legal arsenal if they make irresponsible use of this trust. A bank can provide loans as outgoing and manage savings as incoming trust and thus has a unique opportunity of expressing in a formal and absolute way its credence by publishing its incoming and outgoing trust. Last but not least, the network can be viewed as a possible field for investment and speculation since it constitutes a completely new area for financial activity.

It is worth noting that the same physical person can maintain multiple pseudonymous identities in the same trust network and that multiple independent trust networks for different purposes can coexist. On the other hand, the same pseudonymous identity can be used to establish trust in different contexts.

2 The Trust Graph

We now engage in the formal description of the proposed system, accompanied by helpful examples.

Definition 1 (Graph). *TrustIsRisk is represented by a sequence of directed weighted graphs* $(\mathcal{G}_j)$ *where* $\mathcal{G}_j = (\mathcal{V}_j, \mathcal{E}_j), j \in \mathbb{N}$. *Also, since the graphs are weighted, there exists a sequence of weight functions* (c_j) *with* $c_j : \mathcal{E}_j \to \mathbb{R}^+$.

The nodes represent the players, the edges represent the existing direct trusts and the weights represent the amount of value attached to the corresponding direct trust. As we will see, the game evolves in turns. The subscript of the graph represents the corresponding turn.

Definition 2 (Players). *The set* $\mathcal{V}_j = \mathcal{V}(\mathcal{G}_j)$ *is the set of all players in the network, otherwise understood as the set of all pseudonymous identities.*

Each node has a corresponding non-negative number that represents its capital. A node's capital is the total value that the node possesses exclusively and nobody else can spend.

Definition 3 (Capital). *The initial capital of A,* $Cap_{A,0}$*, is defined as the total value that belongs to A and exists in P2PKH in the UTXO at the beginning of the game.*

The capital of A is subsequently modified only during her turns, according to her actions. A rational player would like to maximize her capital in the long term. We also define a player's assets:

Definition 4 (Assets). *Sum of A's capital and outgoing trust.*

$$As_{A,j} = Cap_{A,j} + out_{A,j} \tag{1}$$

We consider the outgoing trust of a player as part of her assets. The formal definition of direct trust follows:

Definition 5 (Direct Trust). *Direct trust from A to B at the end of turn j,* $DTr_{A\to B,j}$*, is defined as the total amount of value that exists in 1/*$\{A, B\}$ *multisigs in the UTXO in the end of turn j, where the money is deposited by A.*

$$DTr_{A\to B,j} = \begin{cases} c_j(A,B), & if\ (A,B) \in \mathcal{E}_j \\ 0, & else \end{cases} \tag{2}$$

This definition agrees with the title of this paper and coincides with the intuition and experimental results of [4] that the trust *Alice* shows to *Bob* in real-world social networks corresponds with the extent of danger in which *Alice* is ready to expose herself to in order to help *Bob*. An example graph with its corresponding transactions in the UTXO can be seen below.

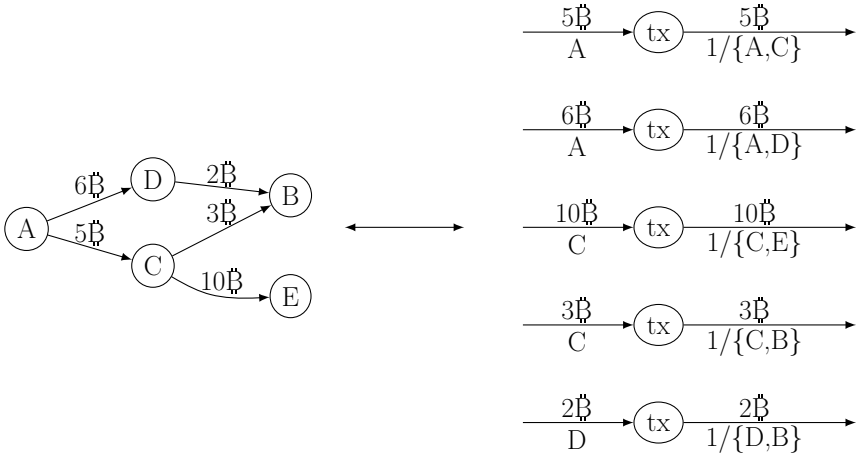

Fig.3: TrustIsRisk Game Graph and Equivalent Bitcoin UTXO

Any algorithm that has access to the graph $\mathcal{G}_j$ has implicitly access to all direct trusts of this graph. We use the notation $N^+(A)$ to refer to the nodes directly trusted by A and $N^-(A)$ for the nodes that directly trust A. We also use the notation $in_{A,j}, out_{A,j}$ to refer to the total incoming and outgoing direct trust respectively. For a reference of common definitions, see the Appendix.

3 Evolution of Trust

Definition 6 (Turns). *The game we are describing is turn-based. In each turn j exactly one player $A \in \mathcal{V}, A = Player\,(j)$, chooses one or more actions from the following two kinds:*
***Steal**(y_B, B): Steal value y_B from $B \in N^-(A)_{j-1}$, where $0 \leq y_B \leq DTr_{B\to A,j-1}$. Then:*

$$DTr_{B\to A,j} = DTr_{B\to A,j-1} - y_B$$

***Add**(y_B, B): Add value y_B to $B \in \mathcal{V}$, where $-DTr_{A\to B,j-1} \leq y_B$. Then:*

$$DTr_{A\to B,j} = DTr_{A\to B,j-1} + y_B$$

When $y_B < 0$, we say that A reduces her trust to B by $-y_B$. When $y_B > 0$, we say that A increases her trust to B by y_B. If $DTr_{A\to B,j-1} = 0$, then we say that A starts directly trusting B. If player A chooses no action in her turn, we say that she passes her turn. Also, let Y_{st}, Y_{add} be the total value to be stolen and added respectively by A in her turn, j. For a turn to be feasible, it must hold that

$$Y_{add} - Y_{st} \leq Cap_{A,j-1} \ . \tag{3}$$

The capital is updated in every turn:

$$Cap_{A,j} = Cap_{A,j-1} + Y_{st} - Y_{add} \ .$$

Moreover, player A is not allowed to choose two actions of the same kind against the same player in one turn. The set of actions the player makes in turn j is denoted by $Turn_j$. The new graph that emerges by applying the actions on $\mathcal{G}_{j-1}$ is $\mathcal{G}_j$.

We use $prev\,(j)$ and $next\,(j)$ to denote the previous and next turn respectively played by $Player(j)$. A formal definition can be found in the Appendix.

Definition 7 (Damage). *Let j be a turn such that $Player\,(j) = A$.*

$$Damage_{A,j} = out_{A,prev(j)} - out_{A,j-1} \tag{4}$$

We say that A has been stolen value $Damage_{A,j}$ between $prev\,(j)$ and j. We omit turn subscripts if they are implied from the context.

Definition 8 (History). *We define History, $\mathcal{H} = (\mathcal{H}_j)$, as the sequence of all tuples containing the sets of actions and the corresponding player.*

$$\mathcal{H}_j = (Player\,(j)\,, Turn_j) \tag{5}$$

Knowledge of the initial graph $\mathcal{G}_0$ and the history amount to full comprehension of the evolution of the game. Building on the example of figure 3, we can see the resulting graph when D plays

$$Turn_1 = \{Steal\,(1, A)\,, Add\,(4, C)\} \ . \tag{6}$$

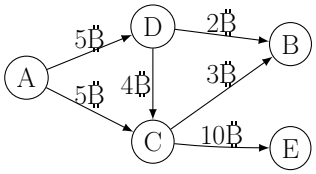

Fig.4: Game Graph after $Turn_1$ (6) on the Graph of figure 3

In the form presented here, TrustIsRisk is controlled by an algorithm that chooses a player, receives the turn that this player wishes to play and, if this turn is valid, executes it. These steps are repeated indefinitely. We assume players are chosen in a way that, after her turn, a player will eventually play again later.

```
TrustIsRisk Game
j = 0
while (True)
  j += 1
  v ←$ V_j
  Turn = vStrategy(G_0, v, (H)_{1...j−1})
  (G_j, Cap_{v,j}, H_j) = executeTurn(G_{j−1}, v, Cap_{v,j−1}, Turn)
```

`executeTurn()` checks the validity of `Turn` and substitutes it with an empty turn if invalid. Subsequently, it creates the new graph $\mathcal{G}_j$ and updates the history accordingly. For the routine code, see the Appendix.

4 Trust Transitivity

In this section we define some strategies and show the corresponding algorithms. Then we define the Transitive Game that represents the worst-case scenario for an honest player when another player decides to depart from the network with her money and all the money entrusted to her.

Definition 9 (Idle Strategy). *A player A is said to follow the idle strategy if she passes in her turn.*

```
Idle Strategy
Input : initial graph G_0, player A, history (H)_{1...j−1}
Output : Turn_j
idleStrategy(G_0, A, H) :
  return(∅)
```

The inputs and outputs are identical to those of `idleStrategy()` for the rest of the strategies, thus we avoid repeating them.

Definition 10 (Evil Strategy). *A player A is said to follow the evil strategy if she steals all incoming direct trust and nullifies her outgoing direct trust in her turn.*

```
1  evilStrategy(G_0, A, H) :
2    Steals = ⋃_{v∈N^-(A)_{j-1}} {Steal(DTr_{v→A,j-1}, v)}
3    Adds = ⋃_{v∈N^+(A)_{j-1}} {Add(−DTr_{A→v,j-1}, v)}
4    Turn_j = Steals ∪ Adds
5    return(Turn_j)
```

Definition 11 (Conservative Strategy). *Player A is said to follow the conservative strategy if she replenishes the value she lost since the previous turn, $Damage_A$, by stealing from others that trust her as much as she can up to $Damage_A$ and she takes no other action.*

```
1  consStrategy(G_0, A, H) :
2    Damage = out_{A,prev(j)} - out_{A,j-1}
3    if (Damage > 0)
4      if (Damage >= in_{A,j-1})
5        Turn_j = ⋃_{v∈N^-(A)_{j-1}} {Steal(DTr_{v→A,j-1}, v)}
6      else
7        y = SelectSteal(G_j, A, Damage) #y = {y_v : v ∈ N^-(A)_{j-1}}
8        Turn_j = ⋃_{v∈N^-(A)_{j-1}} {Steal(y_v, v)}
9    else
10     Turn_j = ∅
11   return(Turn_j)
```

`SelectSteal()` returns y_v with $v \in N^-(A)_{j-1}$ such that

$$\sum_{v \in N^-(A)_{j-1}} y_v = Damage_{A,j} \wedge \forall v \in N^-(A)_{j-1}, y_v \leq DTr_{v \to A, j-1} \ . \quad (7)$$

Each conservative player can arbitrarily define how `SelectSteal()` distributes the $Steal()$ actions each time she calls the function, as long as (7) is respected.

As we can see, the definition covers a multitude of options for the conservative player, since in case $0 < Damage_{A,j} < in_{A,j-1}$ she can choose to distribute the $Steal()$ actions in any way she chooses.

The rationale behind this strategy arises from a real-world common situation. Suppose there are a client, an intermediary and a producer. The client entrusts some value to the intermediary so that the latter can buy the desired product from the producer and deliver it to the client. The intermediary in turn entrusts an equal value to the producer, who needs the value upfront to be able to complete the production process. However the producer eventually does not give the product neither reimburses the value, due to bankruptcy or decision to exit the market with an unfair benefit. The intermediary can choose either to reimburse the client and suffer the loss, or refuse to return the money and lose the client's trust. The latter choice for the intermediary is exactly the conservative strategy. It is used throughout this work as a strategy for all the intermediary players because it models effectively the worst-case scenario that a client can face after an evil player decides to steal everything she can and the rest of the players do not engage in evil activity. Even if some of the rest of the players decide to reduce their outgoing trust as well, this will inhibit some possible damage to them, thus further guarding the client from damage.

We continue with a very useful possible evolution of the game, the Transitive Game. In turn 0, there is already a network in place. All players apart from A and E follow the conservative strategy. Furthermore, the set of players is not modified throughout the Transitive Game, thus we can refer to $\mathcal{V}_j$ for any turn j as $\mathcal{V}$. Moreover, each conservative player can be in one of three states: Happy, Angry or Sad. Happy players have 0 loss, Angry players have positive loss and positive incoming trust, thus are able to replenish their loss at least in part and Sad players have positive loss, but 0 incoming trust, thus they cannot replenish the loss. These conventions will hold whenever we use the Transitive Game.

```
Transitive Game
Input : graph G_0, A ∈ V idle player, E ∈ V evil player
1  Angry = Sad = ∅ ; Happy = V \ {A, E}
2  for (v ∈ V \ {E})
3    Loss_v = 0
4  j = 0
5  while (True)
6    j += 1
7    v <-$ V \ {A}
8    Turn_j = vStrategy(G_0, v, (H)_{1...j-1})
9    executeTurn(G_{j-1}, Cap_{v,j-1}, Turn_j)
```

```
10  for (action ∈ Turn_j)
11    action match do
12      case Steal(y, w) do
13        exchange = y
14        Loss_w += exchange
15        if (v != E)
16          Loss_v -= exchange
17        if (w != A)
18          Happy = Happy \ {w}
19          if (in_{w,j} == 0)
20            Sad = Sad ∪ {w}
21          else
22            Angry = Angry ∪ {w}
23  if (v != E)
24    Angry = Angry \ {v}
25    if (Loss_v > 0)
26      Sad = Sad ∪ {v}        #in_{v,j} should be zero
27    if (Loss_v == 0)
28      Happy = Happy ∪ {v}
```

An example execution follows:

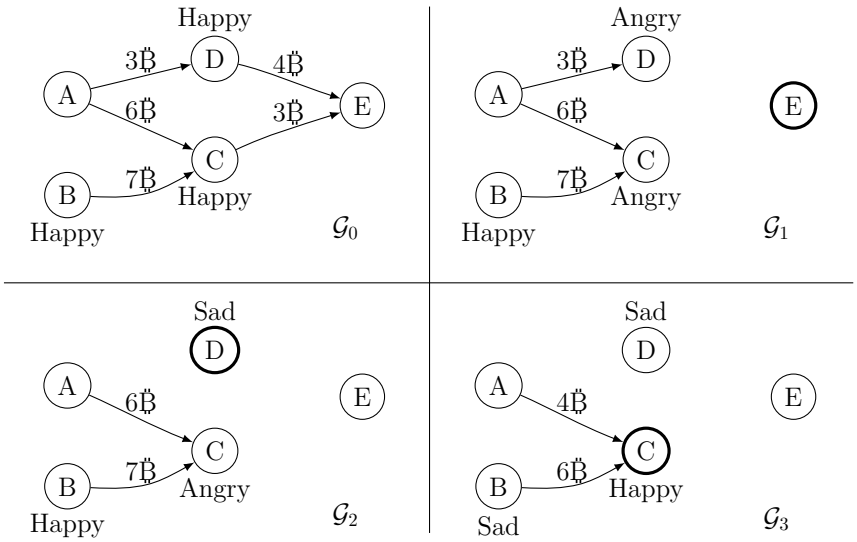

Fig.5: Turns of a `TransitiveGame`($\mathcal{G}_0, A, E$)

Let j_0 be the first turn on which E is chosen to play. Until then, all players will pass their turn since nothing has been stolen yet (see the

Appendix (theorem 11) for a formal proof of this simple fact). Moreover, let $v = Player(j)$ and $j' = prev\,(j)$. The Transitive Game generates turns:

$$Turn_j = \bigcup_{w \in N^-(v)_{j-1}} \{Steal\,(y_w, w)\} \ , \tag{8}$$

where

$$\sum_{w \in N^-(v)_{j-1}} y_w = \min\,(in_{v,j-1}, Damage_{v,j}) \ .$$

We see that if $Damage_{v,j} = 0$, then $Turn_j = \emptyset$.

From the definition of $Damage_{v,j}$ and knowing that no strategy in this case can increase any direct trust, we see that $Damage_{v,j} \geq 0$. Also, we can see that $Loss_{v,j} \geq 0$ because if $Loss_{v,j} < 0$, then v has stolen more value than she has been stolen, thus she would not be following the conservative strategy.

5 Trust Flow

Everything is in place to define the indirect trust, or simply trust, from one player to another.

Definition 12 (Indirect Trust). *The indirect trust from A to B after turn j is defined as the maximum possible value that can be stolen from A after turn j in the setting of* `TransitiveGame(`$\mathcal{G}_j$`,A,B)`*.*

It is $Tr_{A \to B} \geq DTr_{A \to B}$. The next theorem shows that $Tr_{A \to B}$ is finite.

Theorem 1 (Trust Convergence Theorem).
Consider a Transitive Game. There exists a turn such that all subsequent turns are empty:

$$\exists j' : \forall j \geq j', Turn_j = \emptyset \ .$$

Proof Sketch. If the game didn't converge, the $Steal\,()$ actions would continue forever without reduction of the amount stolen over time, thus they would reach infinity. However this is impossible, since there exists only finite total trust. □

Full proofs of all theorems and lemmas can be found in the Appendix.

In the setting of `TransitiveGame(`$\mathcal{G}$`,A,E)`, we make use of the notation $Loss_A = Loss_{A,j}$, where j is a turn that the game has converged. It is important to note that $Loss_A$ is not the same for repeated executions of this kind of game, since the order in which players are chosen may

differ between executions and the conservative players are free to choose which incoming trusts they will steal and how much from each.

Let G be a weighted directed graph. We will investigate the maximum flow on this graph. For an introduction to the maximum flow problem see [5] p. 708. Considering each edge's capacity as its weight, a flow assignment $X = [x_{vw}]_{\mathcal{V}\times\mathcal{V}}$ with a source A and a sink B is valid when:

$$\forall (v,w) \in \mathcal{E}, x_{vw} \leq c_{vw} \text{ and} \tag{9}$$

$$\forall v \in \mathcal{V} \setminus \{A,B\}, \sum_{w\in N^+(v)} x_{wv} = \sum_{w\in N^-(v)} x_{vw} \ . \tag{10}$$

We do not suppose any skew symmetry in X. The flow value is $\sum_{v\in N^+(A)} x_{Av}$, which is proven to be equal to $\sum_{v\in N^-(B)} x_{vB}$. There exists an algorithm that returns the maximum possible flow from A to B, namely $MaxFlow\,(A,B)$. This algorithm evidently needs full knowledge of the graph. The fastest version of this algorithm runs in $O\,(|\mathcal{V}||\mathcal{E}|)$ time [6]. We refer to the flow value of $MaxFlow\,(A,B)$ as $maxFlow\,(A,B)$.

We will now introduce two lemmas that will be used to prove the one of the central results of this work, the Trust Flow theorem.

Lemma 1 (MaxFlows Are Transitive Games).
Let $\mathcal{G}$ be a game graph, let $A, E \in \mathcal{V}$ and $MaxFlow\,(A,E)$ the maximum flow from A to E executed on $\mathcal{G}$. There exists an execution of `TransitiveGame`$(\mathcal{G}, A, E)$ such that

$$maxFlow\,(A,E) \leq Loss_A \ .$$

Proof Sketch. The desired execution of `TransitiveGame()` will contain all flows from the $MaxFlow\,(A,E)$ as equivalent $Steal\,()$ actions. The players will play in turns, moving from E back to A. Each player will steal from his predecessors as much as was stolen from her. The flows and the conservative strategy share the property that the total input is equal to the total output. □

Lemma 2 (Transitive Games Are Flows).
Let $\mathcal{H}$ =`TransitiveGame`$(\mathcal{G}, A, E)$ for some game graph $\mathcal{G}$ and $A, E \in \mathcal{V}$. There exists a valid flow $X = \{x_{wv}\}_{\mathcal{V}\times\mathcal{V}}$ on $\mathcal{G}_0$ such that

$$\sum_{v\in\mathcal{V}} x_{Av} = Loss_A \ .$$

Proof Sketch. If we exclude the sad players from the game, the *Steal* () actions that remain constitute a valid flow from A to E. □

Theorem 2 (Trust Flow Theorem).
Let $\mathcal{G}$ be a game graph and $A, E \in \mathcal{V}$. It holds that

$$Tr_{A\to E} = maxFlow\,(A, E) \quad .$$

Proof. From lemma (1) we see that there exists an execution of the Transitive Game such that $Loss_A \geq maxFlow\,(A, E)$. Since $Tr_{A\to E}$ is the maximum loss that A can suffer after the convergence of the Transitive Game, we see that

$$Tr_{A\to E} \geq maxFlow\,(A, E) \quad . \tag{11}$$

Moreover, there exists an execution of the Transitive Game such that $Tr_{A\to E} = Loss_A$. From lemma (2), this execution corresponds to a flow. Thus

$$Tr_{A\to E} \leq maxFlow\,(A, E) \quad . \tag{12}$$

The theorem follows from (11) and (12). □

We note that the maxFlow is the same in the following two cases: When a player chooses the evil strategy and when the same player chooses a variation of the evil strategy where she does not nullify her outgoing direct trust.

Here we see another important theorem that gives the basis for risk-invariant transactions between different, possibly unknown, parties.

Theorem 3 (Risk Invariance Theorem). *Let $\mathcal{G}$ game graph, $A, B \in \mathcal{V}$ and l the desired value to be transferred from A to B, with $l \leq Tr_{A\to B}$. Let also $\mathcal{G}'$ with the same nodes as $\mathcal{G}$ such that*

$$\forall v \in \mathcal{V}' \setminus \{A\}, \forall w \in \mathcal{V}', DTr'_{v\to w} = DTr_{v\to w} \quad .$$

Furthermore, suppose that there exists an assignment for the outgoing trust of $A, DTr'_{A\to v}$, such that

$$Tr'_{A\to B} = Tr_{A\to B} - l \quad . \tag{13}$$

Let another game graph, $\mathcal{G}''$, be identical to $\mathcal{G}'$ except for the following change:

$$DTr''_{A\to B} = DTr'_{A\to B} + l \quad .$$

It then holds that

$$Tr''_{A\to B} = Tr_{A\to B} \quad .$$

Proof. The two graphs $\mathcal{G}'$ and $\mathcal{G}''$ differ only on the weight of the edge (A, B), which is larger by l in $\mathcal{G}''$. Thus the two $MaxFlow$s will choose the same flow, except for (A, B), where it will be $x''_{AB} = x'_{AB} + l$. □

It is intuitively obvious that it is possible for A to reduce her outgoing direct trust in a manner that achieves (13), since $maxFlow\,(A, B)$ is continuous with respect to A's outgoing direct trusts. We leave this calculation as part of further research.

6 Sybil Resilience

One of the primary aims of this system is to mitigate the danger for Sybil attacks [7] whilst maintaining fully decentralized autonomy.

Here we extend the definition of indirect trust to many players.

Definition 13 (Indirect Trust to Multiple Players). *The indirect trust from player A to a set of players, $S \subset \mathcal{V}$ is defined as the maximum possible value that can be stolen from A if all players in S follow the evil strategy, A follows the idle strategy and everyone else ($\mathcal{V} \setminus (S \cup \{A\})$) follows the conservative strategy. More formally, let choices be the different actions between which the conservative players can choose, then*

$$Tr_{A \to S,j} = \max_{j':j'>j,choices} \left[out_{A,j} - out_{A,j'}\right] \tag{14}$$

We now extend Trust Flow theorem (2) to many players.

Theorem 4 (Multi-Player Trust Flow).
Let $S \subset \mathcal{V}$ and T auxiliary player such that $\forall B \in S, DTr_{B \to T} = \infty$. It holds that

$$\forall A \in \mathcal{V} \setminus S, Tr_{A \to S} = maxFlow\,(A, T) \quad .$$

Proof. If T chooses the evil strategy and all players in S play according to the conservative strategy, they will have to steal all their incoming direct trust since they have suffered an infinite loss, thus they will act in a way identical to following the evil strategy as far as $MaxFlow$ is concerned. The theorem follows thus from the Trust Flow theorem. □

We now define several useful notions to tackle the problem of Sybil attacks. Let Eve be a possible attacker.

Definition 14 (Corrupted Set). *Let $\mathcal{G}$ be a game graph and let Eve have a set of players $\mathcal{B} \subset \mathcal{V}$ corrupted, so that she fully controls their outgoing direct trusts to any player in $\mathcal{V}$ and can also steal all incoming*

direct trust to players in $\mathcal{B}$. We call this the corrupted set. The players $\mathcal{B}$ are considered to be legitimate before the corruption, thus they may be directly trusted by any player in $\mathcal{V}$.

Definition 15 (Sybil Set). *Let $\mathcal{G}$ be a game graph. Since participation in the network does not require any kind of registration, Eve can create any number of players. We will call the set of these players $\mathcal{C}$, or Sybil set. Moreover, Eve can arbitrarily set the direct trusts of any player in $\mathcal{C}$ to any player and can also steal all incoming direct trust to players in $\mathcal{C}$. However, players $\mathcal{C}$ can be directly trusted only by players $\mathcal{B} \cup \mathcal{C}$ but not by players $\mathcal{V} \setminus (\mathcal{B} \cup \mathcal{C})$, where $\mathcal{B}$ is a set of players corrupted by Eve.*

Definition 16 (Collusion). *Let $\mathcal{G}$ be a game graph. Let $\mathcal{B} \subset \mathcal{V}$ be a corrupted set and $\mathcal{C} \subset \mathcal{V}$ be a Sybil set, both controlled by Eve. The tuple $(\mathcal{B}, \mathcal{C})$ is called a collusion and is entirely controlled by a single entity in the physical world. From a game theoretic point of view, players $\mathcal{V} \setminus (\mathcal{B} \cup \mathcal{C})$ perceive the collusion as independent players with a distinct strategy each, whereas in reality they are all subject to a single strategy dictated by the controlling entity, Eve.*

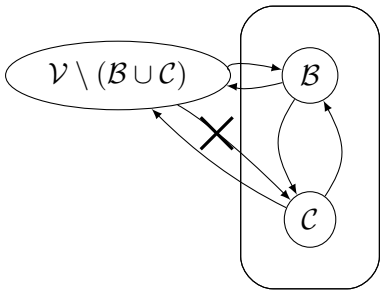

Fig.6: Collusion

Theorem 5 (Sybil Resilience).
Let $\mathcal{G}$ be a game graph and $(\mathcal{B}, \mathcal{C})$ be a collusion of players on $\mathcal{G}$. It holds that

$$Tr_{A \to \mathcal{B} \cup \mathcal{C}} = Tr_{A \to \mathcal{B}} \ .$$

Proof Sketch. The incoming trust to $\mathcal{B} \cup \mathcal{C}$ cannot be higher than the incoming trust to $\mathcal{B}$ since $\mathcal{C}$ has no incoming trust from players outside the collusion. □

We have proven that controlling $|\mathcal{C}|$ is irrelevant for Eve, thus Sybil attacks are meaningless.

7 Risk Invariance Algorithms

We will now focus our discussion on the act of a riskless purchase. According to the Risk Invariance theorem (3), if *Alice* wants to buy something that costs l from the vendor *Bob*, she should first find a new distribution of her outgoing direct trust such that $maxFlow'(A,B) = maxFlow(A,B) - l$ and then entrust l to *Bob*. The theorem then ensures that the initial risk is equal to the final. In the following section we discuss several algorithms that calculate possible new distributions for *Alice*'s outgoing trust.

Let $A \in \mathcal{V}$ source, $B \in \mathcal{V}$ sink. For the following, we suppose that $Turn_{j-1}$ has just finished and $A = Player(j)$ is currently deciding $Turn_j$. We use the following notation:

$$C \text{ capacity configuration with } c_{Av} = DTr_{A\to v,j-1}$$
$$C' \text{ capacity configuration with } c'_{Av} = DTr_{A\to v,j}$$

We will use the following notation for clarity:

$$\begin{cases} X = MaxFlow_{\mathcal{G}_{j-1}}(A,B) \\ X' = MaxFlow_{\mathcal{G}_j}(A,B) \end{cases}, \text{ for some } MaxFlow \text{ execution}$$
$$\begin{cases} F = maxFlow_{\mathcal{G}_{j-1}}(A,B) \\ F' = maxFlow_{\mathcal{G}_j}(A,B) \end{cases}$$

Any subscripts or superscripts applied to X and F refer to the capacity configuration with the exact same subscripts and superscripts.

Furthermore, we suppose an arbitrary ordering of the members of $N^+(A)$. We set $n = |N^+(A)|$. Thus

$$N^+(A) = \{v_1, ..., v_n\}$$

We use these subscripts to refer to the respective capacities (a.k.a. direct trusts) and flows. Thus

$$\begin{matrix} x_i = x_{Av_i} \\ c_i = c_{Av_i} \end{matrix}, \text{ where } i \in [n]$$

Definition 17 (Trust Reduction).
Trust Reduction on neighbour i is defined as $\delta_i = c_i - c'_i$.
Flow Reduction on neigbour i is defined as $\Delta_i = x_i - c'_i$.

We will also use the standard notation for 1-norm and ∞-norm:

$$||\delta_i||_1 = \sum_{i=1}^{n} \delta_i$$
$$||\delta_i||_\infty = \max_{1 \leq i \leq n} \delta_i$$

Theorem 6 (Saturation Theorem).

$$(\forall i \in [n], c'_i \leq x_i) \Rightarrow (\forall i \in [n], x'_i = c'_i)$$

Proof. From the flow definition we know that

$$\forall i \in [n], x'_i \leq c'_i \ . \tag{15}$$

In turn $j-1$, there exists some valid flow Y such that

$$\forall i \in [n], y_i = c'_i$$

with a flow value $\sum_{i=1}^{n} y_i$, which can be created as follows: We start from X and for each (A, v_i) edge we reduce the flow along paths starting from this edge for a total reduction of $x_i - c'_i$ on all those paths. Y is also obviously valid for turn j and, since all capacities c'_i are saturated, there can be no more outgoing flow from the source, thus Y is a maximum flow in $\mathcal{G}_j$. □

Theorem 7 (Trust Transfer Theorem).
Let $V \in [0, F]$. We create a C' where

$$\forall i \in [n], c'_i \leq x_i \text{ and}$$
$$\sum_{i=1}^{n} c'_i = F - V \ .$$

It then holds that $F' = F - V$.

Proof. From the Saturation theorem (6) we can see that $x'_i = c'_i$. It holds that

$$F' = \sum_{i=1}^{n} x'_i = \sum_{i=1}^{n} c'_i = F - V \ .$$

□

Definition 18 (Restricted Flow).
Let $i \in [n]$. Let $F_{A_i \to B}$ be x'_i when:

$$c'_i = c_i \text{ and}$$
$$\forall k \in [n] \setminus \{i\}, c'_k = 0 \ .$$

This definition can be rephrased equivalently as follows:
Let $v \in N^+(A)$. Let $F_{A_v \to B}$ be x'_{Av} when:

$$c'_{Av} = c_{Av} \text{ and}$$
$$\forall w \in N^+(A) \setminus \{v\}, c'_{Aw} = 0 \ .$$

Let $L \subset [n]$. Let $F_{A_L \to B}$ be $\sum_{i \in L} x'_i$ when:

$$\forall i \in L, c'_i = c_i \text{ and}$$
$$\forall i \in [n] \setminus L, c'_i = 0 \ .$$

The latter definition can be rephrased equivalently as follows:
Let $S \subset N^+(A)$. Let $F_{A_S \to B}$ be $\sum_{v \in S} x'_{Av}$ when:

$$\forall v \in S, c'_{Av} = c_{Av} \text{ and}$$
$$\forall v \in N^+(A) \setminus S, c'_{Av} = 0 \ .$$

The choice of the definition will depend on whether K in $F_{A_K \to B}$ is a node, an index or a set of nodes or indices.

Lemma 3 (Flow Limit Lemma).

$$\forall i \in [n], x_i \leq F_{A_i \to B}$$

Proof. Suppose a flow where $\exists i \in [n] : x_i > F_{A_i \to B}$. If for any $k \neq i$ we choose $c'_k < c_k$, then $x'_i \geq x_i$. We set the new capacities as follows:

$$\forall k \neq i, c'_k = 0 \text{ and}$$
$$c'_i = c_i \ .$$

Then for X' we will have

$$\forall k \neq i, x'_k = 0 \text{ and}$$
$$x'_i = x_i \ ,$$

which is also a valid flow for $\mathcal{G}_{j-1}$ and thus by definition

$$F_{A_i \to B} = x'_i = x_i > F_{A_i \to B} \ ,$$

which is a contradiction. Thus the proposition holds. □

Theorem 8 (Trust Saving Theorem).
Suppose some $i \in [n]$ and two alternative capacities configurations, say C_1' and C_2' such that

$$c'_{1,i} = F_{A_i \to B} \ ,$$
$$c'_{2,i} = c_i \ ,$$
$$\forall k \in [n] \setminus \{i\}, c'_{1,k} = c'_{2,k} \ .$$

Then $F_1' = F_2'$.

Proof. From the Flow Limit lemma (3) we know that $x_i \leq F_{A_i \to B}$, thus we can see that any increase in c_i' beyond $F_{A_i \to B}$ will not influence x_i and subsequently will not incur any change on the rest of the flows. □

Theorem 9 (Invariable Trust Reduction with Naive Algorithms).

If $\forall i \in [n], c_i' \leq x_i$, then $||\delta_i||_1$ and $||\Delta_i||_1$ are independent of x_i', c_i'.

Proof. Since $\forall i \in [n], c_i' \leq x_i$, by applying the Saturation theorem (6) we see that $x_i' = c_i'$, thus $\delta_i = c_i - x_i'$ and $\Delta_i = x_i - x_i'$. We know that $\sum_{i=1}^{n} x_i' = F - V$, so we have

$$||\delta_i||_1 = \sum_{i=1}^{n} \delta_i = \sum_{i=1}^{n} (c_i - x_i') = \sum_{i=1}^{n} c_i - F + V \text{ and}$$
$$||\Delta_i||_1 = \sum_{i=1}^{n} \Delta_i = \sum_{i=1}^{n} (x_i - x_i') = \sum_{i=1}^{n} x_i - F + V \ .$$

thus $||\delta_i||_1, ||\Delta_i||_1$ are independent from x_i' and c_i'. □

Until now $MaxFlow$ has been viewed purely as an algorithm. This algorithm is not guaranteed to always return the same flow when executed muliple times on the same graph. However, the corresponding flow value, $maxFlow$, is always the same. Thus $maxFlow$ can be also viewed as a function from a matrix of capacities to a non-negative real number. Under this perspective, we prove the following theorem. Let $\mathcal{C}$ be the family of all capacity matrices $C = [c_{vw}]_{V(\mathcal{G}) \times V(\mathcal{G})}$.

Theorem 10 (maxFlow Continuity).
Suppose a directed weighted graph with a source A, a drain B and a capacity configuration $C \in \mathcal{C}$. Let also $p \in \mathbb{N} \cup \{\infty\}$. The function $maxFlow : \mathcal{C} \to \mathbb{R}^+$ is continuous with respect to the $||\cdot||_p$ norm.

Proof Sketch. An infinitesimal modification to any capacity leads to a no more than infinitesimal modification to the total $maxFlow$. □

Here we show three naive algorithms for calculating new direct trusts so as to maintain invariable risk when paying a trusted party. Let $F = \sum_{i=1}^{n} x_i$. To prove the correctness of the algorithms, it suffices to prove that

$$\forall i \in [n], c'_i \leq x_i \text{ and} \tag{16}$$

$$\sum_{i=1}^{n} c'_i = F - V \ . \tag{17}$$

Proofs of correctness and complexity can be found in the Appendix.

```
First Come First Served Trust Transfer
Input : old flows x_i, value V
Output : new capacities c'_i
1  fcfs((x_i), V) :
2    n = length(x_i)
3    F_cur = F = sum_{i=1}^{n} x_i
4    if (F < V)
5      return(⊥)
6    for (i = 1 to n)
7      c'_i = x_i
8    i = 1
9    while (F_cur > F - V)
10     reduce = min(x_i, F_cur - (F - V))
11     F_cur = F_cur - reduce
12     c'_i = x_i - reduce
13     i += 1
14   return( ⋃_{i=1}^{n} {c'_i})
```

This algorithm simply chooses to nullify all outgoing trust to one player after another in the order they are given at the input, until the desired indirect trust is achieved. The complexity of this algorithm is $O(n)$.

```
Absolute Equality Trust Transfer ( ||Δ_i||_∞ minimizer)
Input : old flows x_i, value V
Output : new capacities c'_i
1  abs((x_i), V) :
2    n = length(x_i)
3    F_cur = F = Σ_{i=1}^{n} x_i
4    if (F < V)
5      return(⊥)
6    X = preprocess(x_i)
7    empty = 0
8    reduction = 0
9    while (F_cur > F - V)
10     (i, X) = popMin(X)
11     F_prov = F_cur - (n - empty)*(x_i - reduction)
12     if (F_prov > F - V)
13       reduction = x_i
14       empty += 1
15       F_cur = F_prov
16     else
17       aux = reduction
18       reduction += (F_cur - (F - V)) / (n - empty)
19       F_cur -= (n - empty)*(reduction - aux)
20       #lines 17 & 19 can be replaced by break. In this
21       #case, the loop (line 9) can become while (TRUE).
22   for (i = 1 to n)
23     c'_i = max(0, x_i - reduction)
24   return(⋃_{i=1}^{n} {c'_i})
```

The function `preprocess(`x_i`)` returns a data structure `X` containing the set of flows (x_i), such that the corresponding function `popMin(X)` is able to repeatedly return a tuple consisiting of the index of the minimum element and a new data structure missing exactly the minimum element. Examples of such pairs of functions are:

$$\begin{cases} \texttt{preprocess = quickSort} \\ \texttt{popMin = (}x_1\texttt{, X} \setminus x_1\texttt{)} \end{cases} \quad \text{and}$$

$$\begin{cases} \texttt{preprocess = FibonacciHeap} \\ \texttt{popMin = (find-min(X),delete-min(X))} \end{cases} .$$

In the general case, the complexity of the algorithm `abs` is proven to be $O(preprocess)+O(n)\,O(popMin)$. In both specific cases, the complexity is $O(n \log n)$. This algorithm also minimizes the $||\Delta_i||_\infty$ norm for the specific set of old flows x_i given as input. A proof of this fact can be found in the Appendix.

```
Proportional Equality Trust Transfer
Input : old flows x_i, value V
Output : new capacities c'_i
prop((x_i), V) :
  n = length(x_i)
  F = sum_{i=1}^{n} x_i
  if (F < V)
    return(⊥)
  for (i = 1 to n)
    c'_i = x_i - V/F * x_i
  return( ⋃_{i=1}^{n} {c'_i} )
```

We can see that this algorithm is simpler than the previous two. The complexity of this algorithm is proven to be $O(n)$.

Naive algorithms result in $c'_i \leq x_i$, thus according to the Invariability Theorem (9), $||\delta_i||_1$ is invariable for any of the possible results C' of these algorithms and the resulting norm is not necessarily the minimum. The following algorithms concentrate on finding a configuration C' that achieves $F' = F - V$ while minimizing two δ_i norms, $||\delta_i||_\infty$ and $||\delta_i||_1$. We start with the $||\delta_i||_\infty$ minimizer.

```
||δ_i||_∞ minimizer
Input : old capacities c_i, source A, drain B, value V, ε_1, ε_2
Output : new capacities c'_i
dinfmin(c_i, V, ε_1, ε_2) :
  n = length(c_i)
  F = maxFlow(A, B, C)
  if (F < V)
    return(⊥)
  if ((ε_1 < 0) or (ε_2 < 0) or (F - V - ε_1 < 0) or
     (F - V + ε_2 > F))
    return(⊥)
  δ_max = max(C)
  δ* = BinSearch(0, δ_max, F - V, n, A, B, C, ε_1, ε_2)
```

```
  for (i = 1 to n)
    c'_i = max(0, c_i - δ*)
  return(⋃_{i=1}^{n} {c'_i})
```

Since trust should be considered as a continuous unit and binary search bisects the interval containing the solution on each recursive call, inclusion of the ϵ-parameters in `BinSearch` is necessary for the algorithm to complete in a finite number of steps.

```
Binary Search Function for ||δ_i||_∞ minimizer
Input : bot, top, F', n, source A, drain B, capacities C,
                                                    ε_1, ε_2
Output : δ*
BinSearch(bot, top, F', n, C, ε_1, ε_2) :
  if (bot == top)
    return(bot)
  else
    mid = (top + bot)/2
    for (i = 1 to n)
      c'_i = max(0, c_i - mid)
    if (maxFlow(A, B, C') < F' - ε_1)
      return(BinSearch(bot, mid, F', n, A, B, C, ε_1, ε_2))
    else if (maxFlow(A, B, C') > F' + ε_2)
      return(BinSearch(mid, top, F', n, A, B, C, ε_1, ε_2))
    else
      return(mid)
```

Let $\delta \in \left[0, \max\limits_{1\leq i\leq n}\{c_i\}\right]$. Furthermore, let C' such that $\forall i \in [n], c'_i = \max(0, c_i - \delta)$. We define $maxFlow(\delta) = maxFlow(A, B, C') = F'$ and $MaxFlow(\delta) = X'$. Conventions similar to F, X and C hold for F', X' and δ as far as subscripts are concerned.

Lemma 4 (maxFlow Monotonicity).
It holds that $\forall \delta \in \left[0, \max\limits_{1\leq i\leq n}\{c_i\}\right]$ such that $maxFlow(\delta) < F$, the function $maxFlow(\delta)$ is strictly decreasing with respect to δ.

Proof Sketch. Proof by contradiction is utilised: If we suppose that $maxFlow(\delta)$ is not always strictly decreasing, then we can find a flow for a specific δ which is larger than the expected $maxFlow$. □

From the previous lemma we deduce that, given a $V \in (0, F)$, if we determine a δ such that $maxFlow(\delta) = F - V$, this δ is unique. Furthermore,

it is

$$||\delta_i||_\infty = \max_{1\leq i\leq n} \delta_i = \max_{1\leq i\leq n} (c_i - c'_i) = \max_{1\leq i\leq n} (c_i - \max(0, c_i - \delta)) = \delta \ .$$

It is proven that the two algorithms work as expected, that is an invocation of `dinfmin` with valid inputs returns a capacity C' that yields the desired $maxFlow$ and minimizes $||\delta_i||_\infty$. The complexity of `BinSearch` is $O\left((maxFlow + n)\log_2\left(\frac{top-bot}{\epsilon_1+\epsilon_2}\right)\right)$ and the complexity of `dinfmin` is $O\left((maxFlow + n)\log_2\left(\frac{\delta_{max}}{\epsilon_1+\epsilon_2}\right)\right)$.

We will now concentrate on finding a capacity configuration C' such that $F' = F - V$ and that minimizes $\sum_{i=1}^{n} (c_i - c'_i) = ||\delta_i||_1$ as well. We treat the flow problem as a linear programming problem. Next we see the formulation of the problem in this form, along with a breakdown of each relevant matrix and vector. In matrix form, the maximum flow problem can be seen as:

$$AX \leq B$$
$$\max CX$$

Where:

$$X = \left.\begin{array}{l} \underbrace{\left[c'_{11} \ldots c'_{1j} \ldots c'_{1n}\right.}_{n} \\ \underbrace{f'_{11} \ldots f'_{1j} \ldots f'_{1n}}_{n} \\ \vdots \\ f'_{i1} \ldots f'_{ij} \ldots f'_{in} \\ \vdots \\ \left. f'_{n1} \ldots f'_{nj} \ldots f'_{nn}\right]^\intercal \end{array}\right\} n$$

$$X : \left(n^2 + n\right) \times 1$$

$$
\begin{aligned}
B = [&\underbrace{c_{11} \dots c_{1j} \dots c_{1n}}_{n} \\
&\left.\begin{array}{l}
\underbrace{0 \dots 0}_{n} \\
\underbrace{c_{21} \dots c_{2j} \dots c_{2n}}_{n} \\
\vdots \\
c_{i1} \dots c_{ij} \dots c_{in} \\
\vdots \\
c_{n1} \dots c_{nj} \dots c_{nn}
\end{array}\right\} n \\
&\underbrace{0 \dots 0}_{n-2} \\
&F - V]^{\intercal}
\end{aligned}
$$

$$
B : \left(n^2 + 2n - 1\right) \times 1
$$

$$
C = [\underbrace{0 \dots 0}_{n} \underbrace{1 \dots 1}_{n} \underbrace{0 \dots 0}_{n(n-1)}]
$$

$$
C : 1 \times \left(n^2 + n\right)
$$

A is implied by the following constraints.

$$
A : \left(n^2 + 2n - 1\right) \times \left(n^2 + n\right)
$$

Let $n = |\mathcal{V}|$, A represented by 1 and B by n. The constraints are:

$$
\begin{aligned}
\forall j \in [n]\,, \quad & c'_{1j} \leq c_{1j} && (n \text{ constraints}) \\
\forall j \in [n]\,, & f'_{1j} - c'_{1j} \leq 0 && (n \text{ constraints}) \\
\forall i \in [n] \setminus \{1\}\,, \forall j \in [n]\,, \quad & f_{ij} \leq c_{ij} && (n^2 - n \text{ constraints}) \\
\forall i \in [n] \setminus \{1, n\}\,, \quad & \sum_{j \in [n]} f'_{ij} = \sum_{j \in [n]} f'_{ji} && (n - 2 \text{ constraints}) \\
& \sum_{j \in [n]} f'_{1j} = F - V && (1 \text{ constraint})
\end{aligned}
\tag{18}
$$

The constraints are $n^2 + 2n - 1$ in total. The special constraints are:

$$\forall j \in [n]\,, c'_{1j} \geq 0$$
$$\forall i, j \in [n]\,, f'_{ij} \geq 0$$

The desired optimization is

$$\max \sum_{j \in [n]} f'_{1j} \ .$$

We would like to find a solution that, except for maximizing the flow, also minimizes $||\delta_i||_1$ at the same time. More precisely, we would like to optimize

$$\min \sum_{v \in \mathcal{V}} (c_{Av} - c'_{Av})$$

as well. Since we wish to optimize with regards to two objective functions, we approach the problem as follows: Initially, we ignore the minimization and derive the dual of the previous problem with respect to the maximisation. We then substitute the two problems' optimisations with an additional constraint that equates the two objective functions. Due to the Strong Duality theorem of linear programming [32], this equality can be achieved only by the common optimal solution of the two problems. Next we treat the combination of constraints and variables of the primal and the dual problem, along with the newly introduced constraint and the previously ignored $||\delta_i||_1$ minimisation as a new linear problem. For every $j \in [n]$, the solution to this problem contains a c'_{1j}. These capacities will comprise the new configuration that player A requires.

We will now describe the dual problem in detail. In matrix form, the dual problem can be seen as:

$$A^\intercal Y \geq C^\intercal$$
$$\min B^\intercal Y$$

Where $A^\intercal, B^\intercal$ and $C^\intercal$ are known and

$$
\begin{aligned}
Y = [& \underbrace{y_{c11} \cdots y_{c1j} \cdots y_{c1n}}_{n} \\
& \left.\begin{array}{l}
\underbrace{y_{fc11} \cdots y_{fc1j} \cdots y_{fc1n}}_{n} \\
\vdots \\
y_{fci1} \cdots y_{fcij} \cdots y_{fcin} \\
\vdots \\
y_{fcn1} \cdots y_{fcnj} \cdots y_{fcnn}
\end{array}\right\} n \\
& \underbrace{y_{f2} \cdots y_{fi} \cdots y_{f(n-1)}}_{n-2} \, y_F]^\intercal
\end{aligned}
$$

$$
Y : \left(n^2 + 2n - 1\right) \times 1
$$

The constraints of the dual problem are:

$$
\begin{aligned}
\forall j \in [n], \quad & y_{c1j} - y_{fc1i} \geq 0 \ (n \text{ constraints}) \\
& y_{fc11} + y_F \geq 1 \ (1 \text{ constraint}) \\
\forall j \in [n] \setminus \{1, n\}, \ & y_{fc1j} - y_{fj} + y_F \geq 1 \ (n-2 \text{ constraints}) \\
& y_{fc1n} + y_F \geq 1 \ (1 \text{ constraint}) \\
\forall i \in [n] \setminus \{1, n\}, \quad & y_{fci1} + y_{fi} \geq 0 \ (n-2 \text{ constraints}) \\
\forall i, j \in [n] \setminus \{1, n\}, \ & y_{fcij} + y_{fi} - y_{fj} \geq 0 \ (n^2 - 4n + 4 \text{ constraints}) \\
\forall i \in [n] \setminus \{1, n\}, \quad & y_{fcin} + y_{fi} \geq 0 \ (n-2 \text{ constraints}) \\
& y_{fcn1} \geq 0 \ (1 \text{ constraint}) \\
\forall j \in [n] \setminus \{1, n\}, \quad & y_{fcnj} - y_{fj} \geq 0 \ (n-2 \text{ constraints}) \\
& y_{fcnn} \geq 0 \ (1 \text{ constraint})
\end{aligned}
\tag{19}
$$

The constraints are $n^2 + n$ in total. The special constraints are:

$$
\begin{aligned}
\forall j \in [n], \ & y_{c1j} \geq 0 \\
\forall i, j \in [n], & y_{fcij} \geq 0 \\
\forall i \in [n] \setminus \{1, n\}, \quad & y_{fi} \in \mathbb{R} \\
& y_F \in \mathbb{R}
\end{aligned}
$$

The desired optimization is

$$
\min \left\{ \sum_{j \in [n]} c_{1j} y_{c1j} + \sum_{i \in [n]} \sum_{j \in [n]} c_{ij} y_{fcij} + (F - V) \, y_F \right\} .
$$

Everything is now in place to define the linear problem whose solution C' yields $maxFlow\,(C') = F - V$ and minimizes $||\delta_i||_\infty$. The constraints are (18) and (19) supplemented with a constraint that equates the two problems' optimality functions:

$$\sum_{j\in[n]} f'_{1j} = \sum_{j\in[n]} c_{1j} y_{c1j} + \sum_{i\in[n]} \sum_{j\in[n]} c_{ij} y_{fcij} + (F - V)\, y_F \ .$$

The desired optimisation is

$$\min \sum_{j\in[n]} \left(c_{1j} - c'_{1j} \right) \ .$$

The final linear program consists of $2n^2 + 2n - 1$ variables and $2n^2 + 2n$ constraints. There exist a variety of algorithms that solve linear programs, such as simplex and ellipsoid algorithm. The ellipsoid algorithm requires polynomial time in the worst-case scenario, but for practical purposes algorithms of the simplex category seem to exhibit better behavior [8].

With this we have successfully delivered our promise for a Sybil-resilient decentralized financial trust system with invariant risk when making purchases, supplemented with a set of algorithms that calculate new trust such that risk is really kept invariant on a purchase.

8 Related Work

Trust is a wide topic that exhibits very interesting properties and can be defined in several, often competing manners. Here we will present briefly several alternative approaches that have been followed in pursuit of a satisfactory model of trust and another tightly related and equally elusive concept, reputation.

The topic of trust has been repeatedly attacked with several approaches: Purely cryptographic infrastructure where trust is rather binary and transitivity is not possible is explored in PGP [9]. A transitive web-of-trust for fighting spam is explored in Freenet [10]. Other systems require central trusted third parties, such as PKI [11] and Bazaar [12], or, in the case of BFT, authenticated membership [13].

Mui and Halberstadt [14] have proposed an elaborate model based on the triptych "trust, reciprocity, reputation", where reciprocal actions of an agent A generate a corresponding reputation, which in turn influences other agents' trust to A. Trusting A inspires other agents to reciprocate, thus completing the cycle. In this model, actions are limited to *cooperate*

and $defect$, reciprocity and reputation between two agents are real numbers in $[0, 1]$, the latter also depending on the context of interest. Lastly trust is derived as a mean value based on the agent's reputation and the known history. The variables are connected using the Beta distribution from statistics.

This model has little resemblance with Trust Is Risk not only in the formalities, but mainly in the approach taken. Trust Is Risk proposes a new financial game, whereas [14] attempts to model and predict all kinds of conceivable trust. Trust Is Risk does not use statistics nor scales trust to $[0, 1]$ and thus can provide strong results, such as the Risk Invariance theorem.

One relevant proposal is [20] which proposes a set of definitions and mathematical manipulations pertaining to trust, essentially providing the core for other integrated trust and reputation systems. Once more the Beta distribution is used to model the expected behavior of others, a fact that results in two major drawbacks. First of all, each agent's actions are confined to exactly two options, a constraint not applicable to real-world applications. Secondly, expecting people to act according to a certain distribution function is inviting them to trick and circumvent this assumption for personal gain. Furthermore, Sybil attacks are not addressed. Lastly, the system proposed has a centralized structure, however its core components could also function in a decentralized manner.

FIRE [15] constitutes another attempt to tackle trust, this time in a practical setting. FIRE aims to create a decentralized rating system for services provided. It essentially calculates trust as "the sum of all the available ratings weighted by the rating relevance and normalized to the range of $[-1, 1]$." This setup needs two very disputable assumptions: Firstly that "[a]gents are willing to share their experiences with others" and secondly that "[a]gents are honest in exchanging information with one another." One side effect of the above assumptions is that FIRE is susceptible to Sybil attacks. Trust Is Risk does not make these assumptions, but can function even when each player follows any strategy she desires.

CORE [16] proposes a reputation protocol for MANET to avoid non collaboration of nodes. It uses the terms trust and reputation almost interchangeably. It is designed in a manner such that reputation in several different settings can be expressed through it, for example it can be applied and improve network speeds on the DSR Route Discovery function and the Packet Forwarding function.

No results relating to Sybil attacks are derived, thus Sybil attacks may be possible, albeit pointless, since this protocol refers to setups without other incentives other than reducing the time for completing various jobs. The trust model required in such settings is considerably simpler and rigorous results are not as necessary for the system to function at a satisfactory level and to improve on similar systems without reputation.

[18] delivers a protocol that claims to cover the reputation needs of MAS, or Multi Agent Systems. The setting of the problem is an open environment where each agent can freely come and go, retaining her reputation in the process. The context of the reputation should be decided according to the needs of each particular distributed task. The reputation data itself is also distributed with the use of a DHT and agents are uniquely identifiable. The system keeps track of all transactions to avoid ratings that do not correspond to a transaction. Furthermore, all transactions must be rated. Platform reputation is also implemented to differentiate between end users and software platforms built for users. Each agent's credentials must be signed by at least one more agent apart from the original one. A layered implementation is used to abstract from the communication details and the backend complexity to the frontend interface.

Since no hard restrictions are placed on reputation form, scale and metric, the system retains a context agnostic stance that puts the burden of avoiding whitewashing and other attacks on those that will implement and use [18] as a reputation mechanism for specific distributed tasks. Some simple attacks are discussed and specific countermeasures are proposed. For example, agent cooperation is rather fruitless since each pair of agents can have only one rating and this single rating does not weigh much. However, Sybil attacks are not even discussed and there are no privacy considerations: all ratings are public and traceable. Staying true to the context agnostic nature of this project, no concrete definitions of reputation or trust are given.

The same problem is attacked in [19], where a middleware trust management system consisting of several protocols is introduced to provide a decentralized, self-organized way for reducing or even eliminating malicious and dishonest behavior by peers. No user intervention is one of the design goals, another one being resilience against peers' collusions. Trust from *Alice* to *Bob* is defined as the probability with which *Bob* will act in *Alice*'s desired manner, as calculated by *Alice*. *Bob*'s reputation is defined as the sum of all ratings he has received, one rating for each object he has offered. This definition results in a unique, global reputation for

each peer. *Bob*'s reputation is stored by all his neighbors. For *Bob* to be deemed trustworthy by *Alice* as far as a certain object is concerned, *Bob*'s reputation must exceed a trust threshold chosen by *Alice* specifically for this kind of object. A protocol similar to IP is used for routing object requests. A certain peer reputation is itself assigned more confidence when more peers respond with this particular reputation. This policy encourages peers to remain connected and makes dishonest reputation reporting riskier.

Several simple attacks such as reputation altering are explored and mitigated with various measures. One attack that is only partially avoided is the situation where *Eve* chooses to have only colluding neighbors that report falsely high reputation values for *Eve*. The confidence value proposed earlier does not mitigate this attack, since *Eve* is in principle not discouraged from creating an arbitrarily large number of fake peers, thus reinforcing her forged positive reputation even more. SybilGuard [37] is proposed as a possible remedy for this type of attacks.

Pace [21] promises to provide a trust management model that can be readily incorporated in decentralized applications that use event-based software architectures. The model concentrates on the internal architecture of each node of the network. Trust relationships between digital, not physical, identities are considered. Internal knowledge and perceptions of each node are carefully separated from externally reported information. The four layers of the internal architecture are the communication layer consisting of the protocol handler, the communication manager and the signature manager, the information layer consisting of the internal information and the external information component, the trust layer is comprised of the key manager, the credential manager and the trust manager and finally the application layer, which consists of the application trust rules component and the application's subarchitecture itself. All the components are implicitly trusted, except for the communication layer, because it relays external, possibly untrusted messages. These messages require an explicit trust value. Several simple attacks such as impersonation are mitigated with cryptographic and other commonly used constructs. Fraudulent actions however must be discovered and highlighted at the application layer, thus once more the burden of solving the more complex trust issues is handed over to each specific use case of this general infrastructure. Bootstrapping new users' trust is accomplished through out-of-band means. According to [21], successful prototype implementations of Decentralized Auctioning, of Common Operational Picture and

of Decentralized File Sharing have been developed using the proposed architecture.

[22] proposes a decentralized hierarchical structure where the most trusted peer is the root of the network. Data routing is decided based on the structure of the network, which in turn is connected to the trust value of each peer. A global trust scheme arises from this setup. This system concentrates on supporting a decentralized network of content distribution, thus no financial trust is discussed. A trust value of a peer is comprised of two numbers: $[TV]$, which represents the number of feedbacks this peer has received and $\{TV\}$, which represents the trust value associated to the peer. After a new feedback is given, $[TV]$ is incremented by 1 and $\{TV\}$ is set to be the mean of all feedbacks, including the new one. It is pointed out that old feedbacks may have to be eliminated in order to maintain a satisfactory level of weight for the new feedbacks. Each new peer is assigned a set of trust managers chosen by her parent peer. The set of trust managers contains more than 1 peer to ostensibly avoid peer collusion and is considered to be unknown to the new peer, or to any other peer except for the parent peer. The parent peer is expected to randomly choose trust managers. Knowledge of the approximate neighbourhood of the trust managers is proposed to protect the anonymity of the trust managers and in the same time avoid broadcasts when requesting trust information. The details of requesting, sending, signing and updating trust for a specific peer are discussed and some experimental setups are assessed with mostly encouraging results.

This architecture is not suitable for Trust Is Risk because of the hierarchical structure that provides a small set of peers the ability to cause major network partitions. Furthermore, it does not provide any hard proof for Sybil resilience or resistance to other kinds of attacks.

[24] is a formal attempt to quantify trust over identity. In this work, trust from *Alice* to *Bob* is represented by a percentage that expresses the level of confidence *Alice* has that *Bob* will only sign public PGP keys whose corresponding private keys are rightly and uniquely owned by the person stated in the public key. This is a special use case for trust, much like financial trust which is a largely unrelated kind of trust. This type of trust has some transitive properties that are derived from statistical models and do not have a connection with the concept of flow. One drawback of such a design is that there is no single indirect trust measure and as a consequence there are no strong results such as the Trust Flow theorem, but each user must experiment and choose one or more trust metrics of her liking based on subjective factors.

A practical solution for determining chains of trust over PGP is [25]. This project has a simple web interface that takes a start and an end public key and returns trust paths from the first to the second consisting of a chain of PGP signatures of public keys freely available on public keyservers. It is up to the user to decide how much trust a particular chain inspires. This project shows how a previously end-to-end construction with no transitive properties such as PGP can be used as basis for a network with transitive properties.

Yet another proposal is Open Reputation [17]. This project provides a general framework for rating generation and dissemination in the era of IoT. Reputers attribute reputation to reputees and make them available through their own reputation. Reputees are rated with certain reputes over predefined virtues. For example, the reputer *Alice* can create a repute of 98 over the virtue "speed" that is defined as an integer between 1 and 100 for a reputee washing machine she bought. *Alice* will publish this repute along with her own reputation.

[17] Does not provide any kind of Sybil resilience, nor does it diverge from the often revisited theme where reputation is projected on an arbitrary bounded scale. This model may serve well under the assumption of honest reputers, but can be tricked if selfish motives require so.

[23] design an ambitious framework that claims to cover all needs of decentralized trust models based on reputation. It consists of the 4C's: Content, Communication, Computation and Counteraction. Content refers to the agents' network structure (hierarchical, nested, etc.), the representation of the reputation (discrete, continuous, etc.), the context of the reputation (financial, medical, etc.) and the period of validity of the reputation. Communication refers to the protocol used to collect and transmit information (hierarchical or ad-hoc, etc.), to the allowed hop count of information and to the actual content of the messages exchanged, possibly of many different types (informational, revocation, confirmation, etc.) Computation engages with the mathematical and design details of trust derivation, such as whether a simple average of recommendations is used, how to combine external information with personal experience and how the period of validity influences the computation. Lastly counteraction expresses the particular model's method of feedback dissemination, with two methods being proposed, namely active and passive dissemination. An XML specification is proposed for all the aforementioned aspects of the desired trust and reputation system.

It is currently unclear whether our model can be expressed in the terms of [23]. More importantly, there is little insight into whether the

task of expressing it in such terms will be a valuable asset for Trust Is Risk.

[4] applies the MaxFlow algorithm into real-world situations of informal contract enforcement and money borrowing schemes. Their approach combines the algorithmic and sociological aspects of the issue in a productive way and their results constitute a strong confirmation of the validity of our assumption that trust is risk. More precisely, they show that money borrowing between residents in an area of Peru can be correctly predicted by deriving direct trust from the time residents spend together and calculating indirect trust with the MaxFlow algorithm. Their results show that our central design choice corresponds to real-world trust dynamics. The one important difference of their model with ours is that the graphs used in [4] are directionless because of the way direct trust is derived, thus making their case a special case of the Trust Is Risk graph, where all direct trust is obligatorily mutual and equal.

In [34] it is stated that "willingness to take risks may be one of the few characteristics common to all trust situations" and [35] cites the same passage, adding "Trust is not taking risk *per se*, but rather it is a *willingness* to take risk." These observations corroborate our choice to define trust as risk.

[35] proposes a concrete model for trust that incorporates several notions. For example, trust from agent A to agent B is a factor of A's Propensity and B's Ability, Benevolence and Integrity. Once more, the target of [35] is different from ours in that [35] attempts to explain the inner workings of trust, whereas we define trust as risk and build a financial game atop of this assumption.

[33] constitutes a thorough and highly informative overview of trust and reputation systems up to the time of writing. Flow models are briefly discussed, however our case is not covered.

Bartercast [36] uses the maximum flow algorithm in an innovative way to calculate trust towards unknown BitTorrent peers. MaxFlow usage there closely resembles to ours, however the abscence of a blockchain leaves room for sybil attacks. Nevertheless, simulation results show that freeriders that selfishly take advantage of the network obtain a progressively worse reputation, a fact that strengthens our reliance on MaxFlow as a suitable algorithm for trust calculation.

Bazaar [12] proposes an encanhement to existing centralized marketplaces where subjective trust is calculated using the MaxFlow algorithm. The bootstrapping process of the network is extremely similar to how players join the Trust Is Risk network. However, their approach contains

a weak point in the way new trust is calculated after each transaction. Furthermore, trust between parties is commutative because non-directed graphs are used. This may be viewed as a crucial restriction for Bazaar. Nevertheless, inclusion of this system as an additional fraud detection system would probably decrease fraud cases and as a result insurance fees would diminish and customer satisfaction would increase.

Beaver [26] proposes an integrated decentralized marketplace solution that provides all the functionality of eBay or other centralized marketplaces. Up to one public review per transaction is permitted and user ratings are globally calculated and not subjective. On the downside, adhoc fees must be attached to several reputation generating actions to deter fraudulent merchants from arbitrarily improving their ratings through Sybil or other attacks. Our system promises to automate and integrate several comparable parts of Beaver in a more intuitive system with less hand-tuned parameters and arbitrary fees that, while discouraging fraudulent action, they also reduce vendors' and customers' desire to participate.

A very different direction is chosen by [29], where an economy based on personal IOUs is proposed. According to this scheme, a payment from *Alice* to *Bob* can be completed by *Alice* offering some of her IOUs. If *Bob* trusts her, that is a valuable enough payment for him. Otherwise they can find a chain of trust, comprised by other intermediary agents, the first of which trusts *Alice* and is trusted by the second and so on until the last one trusts the second last and is trusted by *Bob*. This model of economy has some interesting implications, namely that conventional currency is simply viewed as government IOUs and checks as bank IOUs. Unfortunately, this proposal was made prior to the advent of bitcoin and thus had no concrete basis to be built upon, leaving room for malicious intermediary nodes to fake or disclose contradictory trust amounts to different parties. Furthermore, the distributed nature of the system and its resemblance to contemporary bank relations could sharply increase the time needed for a simple transaction, because active agreement of many intermediate parties would be required.

[28] proposes a system closely related to ours, the mechanics of which are very similar to Trust Is Risk. The work is accomplished in a more economic vein, considering the dynamics that arise from charging a premium for the references (essentially direct trust) parties provide. Trust in itself is not strictly defined, it is just suggested that references be given only to trusted parties. Since this work came before the advent of blockchains, it relies on the honesty of parties to stay true to the references they have

published, given that these references are a type of insurance. The assumption of honesty for most of the players in the long term is backed by the positive social result such a behavior would yield. Long term stabilization of the network and maximum gains for parties can be achieved if there are no fraudulent players. One further implication of the lack of a blockchain is that for every failed transaction a potentially great amount of time may be needed for the defrauded party to collect insurance money. Furthermore, a relatively small number of intermediate players refusing to return the insured money can result in the defrauded party losing money, thus very complex investment strategies are required to minimize risk while maintaining gains.

[27] describes and analyzes the OpenBazaar infrastructure. As Trust Is Risk can be a natural extension of OpenBazaar, the aforementioned work provides valuable insight on how closely related decentralized marketplace systems function. More precisely, its game theoretic analysis constitutes a basis for the future corresponding analysis of our work and the elaborate attacks described and mitigated solve a range of problems that could arise in Trust Is Risk. However, the concept of trust is not rigorously defined in [27], thus Trust Is Risk fills that vacuum in an elegant manner.

The discussion in Synereo [31] does not revolve around trust, but it aims to describe a decentralized social network. As a result, trust is not rigorously defined, but reputation is and its measure is called `Reo`. Risking oversimplification, we could say that the posts of users flow through the network like *current* flows through cables. Some *charge* dissipates for every node that receives the post until no charge is left. A user may pay with `AMP`s to help her post travel further. `AMP` is social capital created by previous popular posts the user has generated. Furthermore, content viewable by *Alice* is personalized through her *engagement* with other users. *Alice*'s engagement with *Bob* is a measure of her amount of time/energy spent on *Bob*'s posts. For example, engagement increases when *Alice* reads and likes a post by *Bob*. *Bob*'s `Reo` is another factor that determines how high *Bob*'s posts will be placed in *Alice*'s stream. *Bob*'s `Reo` as viewed by *Alice* is calculated as the mean engagement of *Alice*'s community with *Bob*'s posts. Putting it all together, current is calculated as the product of charge, `AMP`s, engagement and `Reo`.

While [31] does not qualify as a pure trust model, it undoubtedly contains a host of interesting ideas and approaches on decentralized reputation and mathematical manipulation of arbitrarily many independent event and content generators. It also belongs in the same extended family of ideas as Trust Is Risk in that it attempts to definitively port to the

decentralized setting a popular service that currently exists only through centralized solutions, namely the service of social networking.

Freenet [10] proposes a decentralized platform for secure, deniable file storage and retrieval. Files and file identifiers are hashed and files themselves are additionally encrypted and stored reduntantly in all nodes that receive them, whether said nodes initiated the file request or not. An efficient routing protocol similar to IP is used and a Least Recently Used protocol is used for file retention.

While not directly related with the concept of trust, Freenet is an interesting example of a functional decentralized system that provides specific positive properties and can be trusted as a system from its users, requiring little to no trust between users themselves.

A similar property is exhibited by Bitcoin [2] which assures the existence of a currency with no central issuing authority where users do not have to trust anyone else, just the infrastructure itself. More precisely, the advent of blockchains put the parties that would otherwise be able to forge coins in a situation where they compete amongst themselves to generate a valid block, which will in turn be verified by everybody else. This circumvents elegantly the need for trust to any external party and confines trust to each one's local machine. This is a reasonable trust to demand, since everyone can read and understand [2] and furthermore verify their clients execute the correct code, or even develop their own implementation of the protocol. What Bitcoin achieves is a trustless consensus.

Consensus is a problem loosely related to trust and its best expression is in [13] as the problem of the Byzantine generals, who want to reach a common agreement (e.g. attack or retreat) in the presence of faulty, malicious and dishonest parties who can report inconsistent values to different generals or fail to reply whatsoever. A strong result achieved in [13] and put into revolutionary use in Bitcoin is that in the presence of unforgeable signatures, the honest generals can achieve a consensus no matter how many generals are corrupted.

9 Further Research

While our trust network can form a basis for risk-invariant transactions in a pseudonymous and decentralized setting, more research is required to achieve other desirable properties. Some directions for future research are outlined below.

If *Alice* trusts *Bob* enough to make a purchase from him, she should not directly pay him the value of the good because then she will increase

her trust towards *Bob*. She first has to reduce her outgoing direct trust in a manner such that the supposition (13) of Risk Invariance theorem is satisfied. The methods *Alice* can use to recalculate her outgoing trust will be discussed in a future paper.

The current description of TrustIsRisk refers to a static setting where the game evolves in turns. In each turn only one user changes the state of the network and the game is controlled by a central algorithm, the TrustIsRisk Game. In the dynamic setting, users should be able to play simultaneously, freely join, depart or disconnect temporarily from the network.

Our network evaluates indirect trust by computing the max flow in the graph of lines-of-credit. In order to do that, complete information about the network is required. However, disclosing the network topology may be undesirable, as it subverts the identity of the participants even when participants are treated pseudonymously, as deanonymisation techniques can be used [30]. To avoid such issues, exploring the ability to calculate flows in a zero knowledge fashion may be desirable. However, performing network queries in zero knowledge may allow an adversary to extract topological information. More research is required to establish how flows can be calculated effectively in zero knowledge and what bounds exist in regards to information revealed in such fashion.

Our game theoretic analysis is simple. An interesting analysis would involve modelling repeated purchases with the respective edge updates on the trust graph and treating trust on the network as part of the utility function.

We are proposing a concrete financial game and not a theoretical concept. Thus its implementation as a wallet on any blockchain will be most welcome.

A simulation or actual implementation of TrustIsRisk, combined with analysis of the resulting dynamics can yield interesting experimental results. Subsequently, our trust network can be used in other applications, such as decentralized social networks [31].

1-of-2 multisigs correspond intuitively to simple directed weighted graphs. However it can be interesting to explore the trust relations that can arise by using other types of multisig, such as 1-of-3, as vessel for multi-party trust schemes. Our results do not necessarily hold for other multisigs and the simple relations now represented by directed weighted graphs have to be revised under a new kind of representation.

Appendix

1 Common Notation

Definition 19 (Neighbourhood).

1. *Let $N^+(A)_j$ be the set of players B that A directly trusts with any positive value at the end of turn j. More formally,*

$$N^+(A)_j = \{B \in \mathcal{V}_j : DTr_{A\to B,j} > 0\} \ . \tag{20}$$

$N^+(A)_j$ is called out neighbourhood of A on turn j. Let $S \subseteq \mathcal{V}_j$. Then

$$N^+(S)_j = \bigcup_{A\in S} N^+(A)_j \ . \tag{21}$$

2. *Let $N^-(A)_j$ be the set of players B that directly trust A with any positive value at the end of turn j. More formally,*

$$N^-(A)_j = \{B \in \mathcal{V}_j : DTr_{B\to A,j} > 0\} \ . \tag{22}$$

$N^-(A)_j$ is called in neighbourhood of A on turn j. Let $S \subseteq \mathcal{V}_j$. Then

$$N^-(S)_j = \bigcup_{A\in S} N^-(A)_j \ . \tag{23}$$

3. *Let $N(A)_j$ be the set of players B that either directly trust or are directly trusted by A with any positive value at the end of turn j. More formally,*

$$N(A)_j = N^+(A)_j \cup N^-(A)_j \ . \tag{24}$$

$N(A)_j$ is called neighbourhood of A on turn j. Let $S \subset \mathcal{V}_j$. Then

$$N(S)_j = N^+(S)_j \cup N^-(S)_j \ . \tag{25}$$

Definition 20 (Total Incoming/Outgoing Trust).

$$in_{A,j} = \sum_{v\in N^-(A)_j} DTr_{v\to A,j} \tag{26}$$

$$out_{A,j} = \sum_{v\in N^+(A)_j} DTr_{A\to v,j} \tag{27}$$

We consider the outgoing trust of a player as part of her assets.
Here we add some concrete $Turn_j$ examples. Let $A = Player(j)$.

1.
$$Turn_j = \emptyset$$

2.
$$Turn_j = \{Steal\,(y, B)\,, Add\,(w, B)\}\ ,$$
given that
$$0 \leq y \leq DTr_{B\to A,j-1} \wedge -DTr_{A\to B,j-1} \leq w \wedge w - y \leq Cap_{A,j-1}\ .$$

3.
$$Turn_j = \{Steal\,(x, B)\,, Add\,(y, C)\,, Add\,(w, D)\}\ ,$$
given that
$$0 \leq x \leq DTr_{B\to A,j-1} \wedge -DTr_{A\to C,j-1} \leq y \wedge$$
$$\wedge - DTr_{A\to D,j-1} \leq w \wedge y + w - x \leq Cap_{A,j-1}\ .$$

4.
$$Turn_j = \{Steal\,(x, B)\,, Steal\,(y, B)\}$$
is not a valid turn because it contains two $Steal\,()$ actions against the same player. If
$$0 \leq x \wedge 0 \leq y \wedge x + y \leq DTr_{B\to A,j-1}\ ,$$
the correct alternative would be
$$Turn_j = \{Steal\,(x + y, B)\}\ .$$

Definition 21 (Previous/Next Turn). *Let $j \in \mathbb{N}$ a turn with $Player\,(j) = A$. We define $prev\,(j)\,, next\,(j)$ as the previous and next turn that A is chosen to play respectively. If j is the first turn that A plays, $prev\,(j) = 0$. More formally, if*
$$P = \{k \in \mathbb{N} : k < j \wedge Player\,(k) = A\}\ \textit{and}$$
$$N = \{k \in \mathbb{N} : k > j \wedge Player\,(k) = A\}\ ,$$
then we define $prev\,(j)\,, next\,(j)$ as follows:
$$prev\,(j) = \begin{cases} \max P, & P \neq \emptyset \\ 0, & P = \emptyset \end{cases} \tag{28}$$
$$next\,(j) = \min N \tag{29}$$
$next\,(j)$ is always well defined with the assumption that after each turn eventually everybody plays.

2 Proofs, Lemmas and Theorems

Lemma 5 (*Loss* Equivalent to *Damage*).
Consider a Transitive Game. Let $j \in \mathbb{N}$ and $v = Player(j)$ such that v is following the conservative strategy. It holds that

$$\min(in_{v,j}, Loss_{v,j}) = \min(in_{v,j}, Damage_{v,j}) \ .$$

Proof.
Case 1: Let $v \in Happy_{j-1}$. Then

1. $v \in Happy_j$ because $Turn_j = \emptyset$,
2. $Loss_{v,j} = 0$ because otherwise $v \notin Happy_j$,
3. $Damage_{v,j} = 0$, or else any reduction in direct trust to v would increase equally $Loss_{v,j}$ (line 14), which cannot be decreased again but during an Angry player's turn (line 16).
4. $in_{v,j} \geq 0$

Thus

$$\min(in_{v,j}, Loss_{v,j}) = \min(in_{v,j}, Damage_{v,j}) = 0 \ .$$

Case 2: Let $v \in Sad_{j-1}$. Then

1. $v \in Sad_j$ because $Turn_j = \emptyset$,
2. $in_{v,j} = 0$ (line 25),
3. $Damage_{v,j} \geq 0 \wedge Loss_{v,j} \geq 0$.

Thus

$$\min(in_{v,j}, Loss_{v,j}) = \min(in_{v,j}, Damage_{v,j}) = 0 \ .$$

If $v \in Angry_{j-1}$ then the same argument as in cases 1 and 2 hold when $v \in Happy_j$ and $v \in Sad_j$ respectively if we ignore the argument (1). Thus the theorem holds in every case. □

Proof of Theorem 1: Trust Convergence
First of all, after turn j_0 player E will always pass her turn because she has already nullified her incoming and outgoing direct trusts in $Turn_{j_0}$, the evil strategy does not contain any case where direct trust is increased or where the evil player starts directly trusting another player and the other players do not follow a strategy in which they can choose to $Add()$ trust to E. The same holds for player A because she follows the idle strategy. As far as the rest of the players are concerned, consider the Transitive Game. As we can see from lines 3 and 14 - 16, it is

$$\forall j, \sum_{v \in \mathcal{V}_j} Loss_v = in_{E,j_0-1} \ .$$

In other words, the total loss is constant and equal to the total value stolen by E. Also, as we can see in lines 1 and 26, which are the only lines where the Sad set is modified, once a player enters the Sad set, it is impossible to exit from this set. Also, we can see that players in $Sad \cup Happy$ always pass their turn. We will now show that eventually the $Angry$ set will be empty, or equivalently that eventually every player will pass their turn. Suppose that it is possible to have an infinite amount of turns in which players do not choose to pass. We know that the number of nodes is finite, thus this is possible only if

$$\exists j' : \forall j \geq j', |Angry_j \cup Happy_j| = c > 0 \wedge Angry_j \neq \emptyset \ .$$

This statement is valid because the total number of angry and happy players cannot increase because no player leaves the Sad set and if it were to be decreased, it would eventually reach 0. Since $Angry_j \neq \emptyset$, a player v that will not pass her turn will eventually be chosen to play. According to the Transitive Game, v will either deplete her incoming trust and enter the Sad set (line 26), which is contradicting $|Angry_j \cup Happy_j| = c$, or will steal enough value to enter the $Happy$ set, that is v will achieve $Loss_{v,j} = 0$. Suppose that she has stolen m players. They, in their turn, will steal total value at least equal to the value stolen by v (since they cannot go sad, as explained above). However, this means that, since the total value being stolen will never be reduced and the turns this will happen are infinite, the players must steal an infinite amount of value, which is impossible because the direct trusts are finite in number and in value. More precisely, let j_1 be a turn in which a conservative player is chosen and

$$\forall j \in \mathbb{N}, DTr_j = \sum_{w,w' \in \mathcal{V}} DTr_{w \to w',j} \ .$$

Also, without loss of generality, suppose that

$$\forall j \geq j_1, out_{A,j} = out_{A,j_1} \ .$$

In $Turn_{j_1}$, v steals

$$St = \sum_{i=1}^{m} y_i \ .$$

We will show using induction that

$$\forall n \in \mathbb{N}, \exists j_n \in \mathbb{N} : DTr_{j_n} \leq DTr_{j_1 - 1} - nSt \ .$$

Base case: It holds that

$$DTr_{j_1} = DTr_{j_1 - 1} - St \ .$$

Eventually there is a turn j_2 when every player in $N^-(v)_{j-1}$ will have played. Then it holds that

$$DTr_{j_2} \leq DTr_{j_1} - St = DTr_{j_1-1} - 2St \ ,$$

since all players in $N^-(v)_{j-1}$ follow the conservative strategy, except for A, who will not have been stolen anything due to the supposition.

Induction hypothesis: Suppose that

$$\exists k > 1 : j_k > j_{k-1} > j_1 \Rightarrow DTr_{j_k} \leq DTr_{j_{k-1}} - St \ .$$

Induction step: There exists a subset of the $Angry$ players, S, that have been stolen at least value St in total between the turns j_{k-1} and j_k, thus there exists a turn j_{k+1} such that all players in S will have played and thus

$$DTr_{j_{k+1}} \leq DTr_{j_k} - St \ .$$

We have proven by induction that

$$\forall n \in \mathbb{N}, \exists j_n \in \mathbb{N} : DTr_{j_n} \leq DTr_{j_1-1} - nSt \ .$$

However

$$DTr_{j_1-1} \geq 0 \wedge St > 0 \ ,$$

thus

$$\exists n' \in \mathbb{N} : n'St > DTr_{j_1-1} \Rightarrow DTr_{j_{n'}} < 0 \ .$$

We have a contradiction because

$$\forall w, w' \in \mathcal{V}, \forall j \in \mathbb{N}, DTr_{w \to w', j} \geq 0 \ ,$$

thus eventually $Angry = \emptyset$ and everybody passes. □

Proof of Lemma 1: MaxFlows Are Transitive Games

We suppose that the turn of $\mathcal{G}$ is 0. In other words, $\mathcal{G} = \mathcal{G}_0$. Let $X = \{x_{vw}\}_{\mathcal{V}\times\mathcal{V}}$ be the flows returned by $MaxFlow\,(A, E)$. For any graph G there exists a $MaxFlow$ that is a DAG. We can easily prove this using the Flow Decomposition theorem [32], which states that each flow can be seen as a finite set of paths from A to E and cycles, each having a certain flow. We execute $MaxFlow\,(A, E)$ and we apply the aforementioned theorem. The cycles do not influence the $maxFlow\,(A, E)$, thus we can remove these flows. The resulting flow is a $MaxFlow\,(A, E)$ without cycles, thus it is a DAG. Topologically sorting this DAG, we obtain a total order of its nodes such that $\forall$ nodes $v, w \in \mathcal{V} : v < w \Rightarrow x_{wv} = 0$ [5]. Put

differently, there is no flow from larger to smaller nodes. E is maximum since it is the sink and thus has no outgoing flow to any node and A is minimum since it is the source and thus has no incoming flow from any node. The desired execution of Transitive Game will choose players following the total order inversely, starting from player E. We observe that $\forall v \in \mathcal{V} \setminus \{A, E\}, \sum\limits_{w \in \mathcal{V}} x_{wv} = \sum\limits_{w \in \mathcal{V}} x_{vw} \leq maxFlow\left(A, E\right) \leq in_{E,0}$. Player E will follow a modified evil strategy where she steals value equal to her total incoming flow, not her total incoming trust. Let j_2 be the first turn when A is chosen to play. We will show using strong induction that there exists a set of valid actions for each player according to their respective strategy such that at the end of each turn j the corresponding player $v = Player\left(j\right)$ will have stolen value x_{wv} from each in-neighbour w.

Base case: In turn 1, E steals value equal to $\sum\limits_{w \in \mathcal{V}} x_{wE}$, following the modified evil strategy.

$$Turn_1 = \bigcup_{v \in N^-(E)_0} \{Steal\left(x_{vE}, v\right)\}$$

Induction hypothesis: Let $k \in [j_2 - 2]$. We suppose that $\forall i \in [k]$, there exists a valid set of actions, $Turn_i$, performed by $v = Player\left(i\right)$ such that v steals from each player w value equal to x_{wv}.

$$\forall i \in [k], Turn_i = \bigcup_{w \in N^-(v)_{i-1}} \{Steal\left(x_{wv}, w\right)\}$$

Induction step: Let $j = k + 1, v = Player\left(j\right)$. Since all the players that are greater than v in the total order have already played and all of them have stolen value equal to their incoming flow, we deduce that v has been stolen value equal to $\sum\limits_{w \in N^+(v)_{j-1}} x_{vw}$. Since it is the first time v plays, $\forall w \in N^-\left(v\right)_{j-1}, DTr_{w \to v, j-1} = DTr_{w \to v, 0} \geq x_{wv}$, thus v is able to choose the following turn:

$$Turn_j = \bigcup_{w \in N^-(v)_{j-1}} \{Steal\left(x_{wv}, w\right)\}$$

Moreover, this turn satisfies the conservative strategy since

$$\sum_{w \in N^-(v)_{j-1}} x_{wv} = \sum_{w \in N^+(v)_{j-1}} x_{vw} \ .$$

Thus $Turn_j$ is a valid turn for the conservative player v.

We have proven that in the end of turn $j_2 - 1$, player E and all the conservative players will have stolen value exactly equal to their total incoming flow, thus A will have been stolen value equal to her outgoing flow, which is $maxFlow(A, E)$. Since there remains no Angry player, j_2 is a convergence turn, thus $Loss_{A,j_2} = Loss_A$. We can also see that if E had chosen the original evil strategy, the described actions would still be valid only by supplementing them with additional $Steal\,()$ actions, thus $Loss_A$ would further increase. This proves the theorem. □

Proof of Lemma 2: Transitive Games Are Flows

Let $Sad, Happy, Angry$ be as defined in the Transitive Game. Let $\mathcal{G}'$ be a directed weighted graph based on $\mathcal{G}$ with an auxiliary source. Let also j_1 be a turn when the Transitive Game has converged. More precisely, $\mathcal{G}'$ is defined as follows:

$$\mathcal{V}' = \mathcal{V} \cup \{T\}$$

$$\mathcal{E}' = \mathcal{E} \cup \{(T, A)\} \cup \{(T, v) : v \in Sad_{j_1}\}$$

$$\forall (v, w) \in \mathcal{E}, c'_{vw} = DTr_{v \to w,0} - DTr_{v \to w,j_1}$$

$$\forall v \in Sad_{j_1}, c'_{Tv} = c'_{TA} = \infty$$

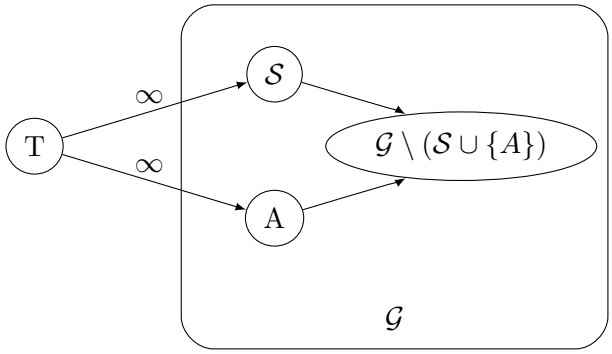

Fig.7: Graph $\mathcal{G}'$, derived from $\mathcal{G}$ with Auxiliary Source T.

In the figure above, $\mathcal{S}$ is the set of sad players. We observe that $\forall v \in \mathcal{V}$,

$$\begin{aligned} &\sum_{w \in N^-(v)' \setminus \{T\}} c'_{wv} = \\ &= \sum_{w \in N^-(v)' \setminus \{T\}} (DTr_{w \to v,0} - DTr_{w \to v,j_1}) = \\ &= \sum_{w \in N^-(v)' \setminus \{T\}} DTr_{w \to v,0} - \sum_{w \in N^-(v)' \setminus \{T\}} DTr_{w \to v,j-1} = \\ &= in_{v,0} - in_{v,j_1} \end{aligned} \tag{30}$$

and

$$\begin{aligned} &\sum_{w \in N^+(v)' \setminus \{T\}} c'_{vw} = \\ &= \sum_{w \in N^+(v)' \setminus \{T\}} (DTr_{v \to w,0} - DTr_{v \to w,j_1}) = \\ &= \sum_{w \in N^+(v)' \setminus \{T\}} DTr_{v \to w,0} - \sum_{w \in N^+(v)' \setminus \{T\}} DTr_{v \to w,j-1} = \\ &= out_{v,0} - out_{v,j_1} \ . \end{aligned} \tag{31}$$

We can suppose that

$$\forall j \in \mathbb{N}, in_{A,j} = 0 \ , \tag{32}$$

since if we find a valid flow under this assumption, the flow will still be valid for the original graph.
Next we try to calculate $MaxFlow\,(T, E) = X'$ on graph $\mathcal{G}'$. We observe that a flow in which it holds that $\forall v, w \in \mathcal{V}, x'_{vw} = c'_{vw}$ can be valid for the following reasons:

- $\forall v, w \in \mathcal{V}, x'_{vw} \leq c'_{vw}$ (Capacity flow requirement (9) $\forall e \in \mathcal{E}$)
- Since $\forall v \in Sad_{j_1} \cup \{A\}, c'_{Tv} = \infty$, requirement (9) holds for any flow $x'_{Tv} \geq 0$.
- Let $v \in \mathcal{V}' \setminus (Sad_{j_1} \cup \{T, A, E\})$. According to the conservative strategy and since $v \notin Sad_{j_1}$, it holds that

  $$out_{v,0} - out_{v,j_1} = in_{v,0} - in_{v,j_1} \ .$$

  Combining this observation with (30) and (31), we have that

  $$\sum_{w \in \mathcal{V}'} c'_{vw} = \sum_{w \in \mathcal{V}'} c'_{wv} \ .$$

  (Flow Conservation requirement (10) $\forall v \in \mathcal{V}' \setminus (Sad_{j_1} \cup \{T, A, E\})$)

– Let $v \in Sad_{j_1}$. Since v is sad, we know that

$$out_{v,0} - out_{v,j_1} > in_{v,0} - in_{v,j_1} \ .$$

Since $c'_{Tv} = \infty$, we can set

$$x'_{Tv} = (out_{v,0} - out_{v,j_1}) - (in_{v,0} - in_{v,j_1}) \ .$$

In this way, we have

$$\sum_{w \in \mathcal{V}'} x'_{vw} = out_{v,0} - out_{v,j_1} \text{ and}$$

$$\sum_{w \in \mathcal{V}'} x'_{wv} = \sum_{w \in \mathcal{V}' \setminus \{T\}} c'_{wv} + x'_{Tv} = in_{v,0} - in_{v,j_1} +$$
$$+ (out_{v,0} - out_{v,j_1}) - (in_{v,0} - in_{v,j_1}) = out_{v,0} - out_{v,j_1} \ .$$

thus

$$\sum_{w \in \mathcal{V}'} x'_{vw} = \sum_{w \in \mathcal{V}'} x'_{wv} \ .$$

(Requirement 10 $\forall v \in Sad_{j_1}$)

– Since $c'_{TA} = \infty$, we can set

$$x'_{TA} = \sum_{v \in \mathcal{V}'} x'_{Av} \ ,$$

thus from (32) we have

$$\sum_{v \in \mathcal{V}'} x'_{vA} = \sum_{v \in \mathcal{V}'} x'_{Av} \ .$$

(Requirement 10 for A)

We saw that for all nodes, the necessary properties for a flow to be valid hold and thus X' is a valid flow for $\mathcal{G}$. Moreover, this flow is equal to $maxFlow(T, E)$ because all incoming flows to E are saturated. Also we observe that

$$\sum_{v \in \mathcal{V}'} x'_{Av} = \sum_{v \in \mathcal{V}'} c'_{Av} = out_{A,0} - out_{A,j_1} = Loss_A \ . \tag{33}$$

We define another graph, $\mathcal{G}''$, based on $\mathcal{G}'$.

$$\mathcal{V}'' = \mathcal{V}'$$

$$E(\mathcal{G}'') = E(\mathcal{G}') \setminus \{(T, v) : v \in Sad_j\}$$
$$\forall e \in E(\mathcal{G}''), c''_e = c'_e$$

If we execute $MaxFlow(T, E)$ on the graph $\mathcal{G}''$, we will obtain a flow X'' in which

$$\sum_{v \in \mathcal{V}''} x''_{Tv} = x''_{TA} = \sum_{v \in \mathcal{V}''} x''_{Av} \ .$$

The outgoing flow from A in X'' will remain the same as in X' for two reasons: Firstly, using the Flow Decomposition theorem [32] and deleting the paths that contain edges $(T, v) : v \neq A$, we obtain a flow configuration where the total outgoing flow from A remains invariant, [3] thus

$$\sum_{v \in \mathcal{V}''} x''_{Av} \geq \sum_{v \in \mathcal{V}'} x'_{Av} \ .$$

Secondly, we have

$$\left.\begin{array}{l} \sum\limits_{v \in \mathcal{V}''} c''_{Av} = \sum\limits_{v \in \mathcal{V}'} c'_{Av} = \sum\limits_{v \in \mathcal{V}'} x'_{Av} \\ \sum\limits_{v \in \mathcal{V}''} c''_{Av} \geq \sum\limits_{v \in \mathcal{V}''} x''_{Av} \end{array}\right\} \Rightarrow \sum_{v \in \mathcal{V}''} x''_{Av} \leq \sum_{v \in \mathcal{V}'} x'_{Av} \ .$$

Thus we conclude that

$$\sum_{v \in \mathcal{V}''} x''_{Av} = \sum_{v \in \mathcal{V}'} x'_{Av} \ . \tag{34}$$

Let $X = X'' \setminus \{(T, A)\}$. Observe that

$$\sum_{v \in \mathcal{V}''} x''_{Av} = \sum_{v \in \mathcal{V}} x_{Av} \ .$$

This flow is valid on graph $\mathcal{G}$ because

$$\forall e \in \mathcal{E}, c_e \geq c''_e \ .$$

Thus there exists a valid flow for each execution of the Transitive Game such that

$$\sum_{v \in \mathcal{V}} x_{Av} = \sum_{v \in \mathcal{V}''} x''_{Av} \overset{(34)}{=} \sum_{v \in \mathcal{V}'} x'_{Av} \overset{(33)}{=} Loss_{A,j_1} \ ,$$

which is the flow X. □

[3] We thank Kyriakos Axiotis for his insights on the Flow Decomposition theorem.

Theorem 11 (Conservative World Theorem).
If everybody follows the conservative strategy, nobody steals any amount from anybody.

Proof. Let $\mathcal{H}$ be the game history where all players are conservative and suppose there are some $Steal\,()$ actions taking place. Then let $\mathcal{H}'$ be the subsequence of turns each containing at least one $Steal\,()$ action. This subsequence is evidently nonempty, thus it must have a first element. The player corresponding to that turn, A, has chosen a $Steal\,()$ action and no previous player has chosen such an action. However, player A follows the conservative strategy, which is a contradiction. □

Proof of Theorem 5: Sybil Resilience
Let $\mathcal{G}_1$ be a game graph defined as follows:

$$\mathcal{V}_1 = \mathcal{V} \cup \{T_1\} \ ,$$

$$\mathcal{E}_1 = \mathcal{E} \cup \{(v, T_1) : v \in \mathcal{B} \cup \mathcal{C}\} \ ,$$

$$\forall v, w \in \mathcal{V}_1 \setminus \{T_1\}, DTr^1_{v\to w} = DTr_{v\to w} \ ,$$

$$\forall v \in \mathcal{B} \cup \mathcal{C}, DTr^1_{v\to T_1} = \infty \ ,$$

where $DTr_{v\to w}$ is the direct trust from v to w in $\mathcal{G}$ and $DTr^1_{v\to w}$ is the direct trust from v to w in $\mathcal{G}_1$.
Let also $\mathcal{G}_2$ be the induced graph that results from $\mathcal{G}_1$ if we remove the Sybil set, $\mathcal{C}$. We rename T_1 to T_2 and define $\mathcal{L} = \mathcal{V} \setminus (\mathcal{B} \cup \mathcal{C})$ as the set of legitimate players to facilitate comprehension.

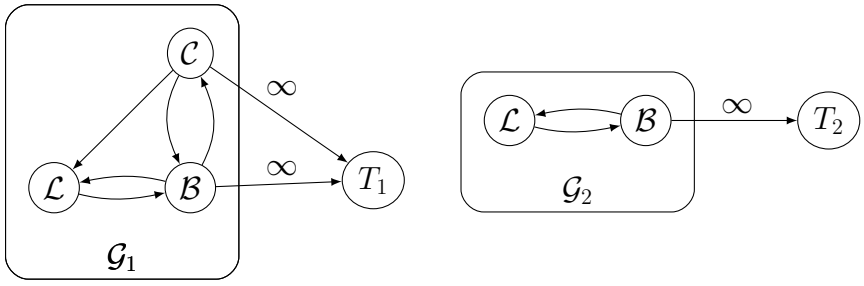

Fig.8: Graphs $\mathcal{G}_1$ and $\mathcal{G}_2$

According to theorem (4),

$$Tr_{A\to\mathcal{B}\cup\mathcal{C}} = maxFlow_1\,(A, T_1) \wedge Tr_{A\to\mathcal{B}} = maxFlow_2\,(A, T_2) \ . \quad (35)$$

We will show that the $MaxFlow$ of each of the two graphs can be used to construct a valid flow of equal value for the other graph. The flow $X_1 = MaxFlow\left(A, T_1\right)$ can be used to construct a valid flow of equal value for the second graph if we set

$$\begin{aligned} \forall v \in \mathcal{V}_2 \setminus \mathcal{B}, \forall w \in \mathcal{V}_2, x_{vw,2} = x_{vw,1} \ , \\ \forall v \in \mathcal{B}, x_{vT_2,2} = \sum_{w \in N_1^+(v)} x_{vw,1} \ , \\ \forall v, w \in \mathcal{B}, x_{vw,2} = 0 \ . \end{aligned}$$

Therefore

$$maxFlow_1\left(A, T_1\right) \leq maxFlow_2\left(A, T_2\right)$$

Likewise, the flow $X_2 = MaxFlow(A, T_2)$ is a valid flow for $\mathcal{G}_1$ because $\mathcal{G}_2$ is an induced subgraph of $\mathcal{G}_1$. Therefore

$$maxFlow_1\left(A, T_1\right) \geq maxFlow_2\left(A, T_2\right)$$

We conclude that

$$maxFlow\left(A, T_1\right) = maxFlow\left(A, T_2\right) \ , \tag{36}$$

thus from (35) and (36) the theorem holds. □

Proof of Theorem 10: maxFlow Continuity
Let $C_0 \in \mathcal{C}$. We want to prove that

$$\forall \epsilon > 0, \exists \delta > 0 : 0 < ||C - C_0||_p < \delta \Rightarrow |F - F_0| < \epsilon \ .$$

We will prove it by contradiction. Suppose that

$$\exists \epsilon > 0 : \forall \delta > 0, 0 < ||C - C_0||_p < \delta \Rightarrow |F - F_0| \geq \epsilon \ .$$

Let $v_1, u_1 \in V\left(\mathcal{G}\right)$. Let C such that

$$\begin{aligned} c_{v_1 u_1} = c_{0, v_1 u_1} + \frac{\epsilon}{2} \\ \forall \left(v, u\right) \in E\left(\mathcal{G}\right) \setminus \{(v_1, u_1)\}, c_{vu} = c_{0,vu} \ . \end{aligned}$$

Due to the construction, for $\delta = \epsilon$ we have

$$0 < ||C - C_0||_p < \delta \ . \tag{37}$$

Any valid flow for C_0 is also valid for C, thus

$$F_0 \leq F \ . \tag{38}$$

Also, it is obvious by the way that C was constructed that

$$F \leq F_0 + \frac{\epsilon}{2} \tag{39}$$

From (38) we have $F_0 \leq F + \frac{\epsilon}{2}$, which, in combination with (39), gives

$$|F - F_0| \leq \frac{\epsilon}{2} < \epsilon \ ,$$

which, together with (37) contradicts our supposition. Thus $maxFlow$ is continuous on C_0. Since C_0 is arbitrary, the result holds for all $C_0 \in \mathcal{C}$, thus $maxFlow$ is continuous with respect to $||\cdot||_p$ for any $p \in \mathbb{N} \cup \{\infty\}$. □

Proof of correctness for algorithm `fcfs`

We will first show that at the end of the execution, $i \leq n+1$. Suppose that $i > n+1$ on line 14. This means that $F_{cur,n}$ exists and $F_{cur,n} = F - \sum_{i=1}^{n} x_i = 0 \leq F - V$ since, according to the condition on line 4, $F - V \geq 0$. This means however that the `while` loop on line 9 will break, thus $F_{cur,n+1}$ cannot exist and $i = n + 1$ on line 14, which is a contradiction, thus the first proposition holds. We can also note that, even if $i = n + 1$ at the end of the execution, the `while` loop will break right after the last incrementation, thus the algorithm will never try to read or write the nonexistent objects x_{n+1}, c'_{n+1}.

We will now show that $\forall i \in [n], c'_i \leq x_i$, as per the requirement (16). Let $i \in [n]$. In line 7 we can see that $c'_i = x_i$ and the only other occurence of c'_i is in line 12 where it is never increased ($reduce \geq 0$), thus we see that the requirement (16) is satisfied.

We will finally show that $\sum_{i=1}^{n} c'_i = F - V$. From line 3 we see that $F_{cur,0} = F$. Let $i \in [n]$ such that $F_{cur,i}$ exists. If $F_{cur,i} \leq F - V$, then $F_{cur,i+1}$ does not exist because the `while` loop (line 9) breaks after calculating $F_{cur,i}$. Else

$$F_{cur,i+1} = F_{cur,i} - \min\left(x_{i+1}, F_{cur,i} - F + V\right) \ . \text{ (lines 10- 11)}$$

If $\nexists i \in [n] : \min\left(x_i, F_{cur,i-1} - (F - V)\right) = F_{cur,i-1} - (F - V)$, then $\forall i \in [n], \min\left(x_i, F_{cur,i} - (F - V)\right) = x_i$, thus from line 12 it will be

$\forall i \in [n], c'_i = 0$ and from line 11, $F_{cur,n} = 0$. However, we have

$$\left.\begin{array}{l} min\left(x_n, F_{cur,n-1} - (F-V)\right) \neq F_{cur,n-1} - (F-V) \\ F_{cur,n-1} = x_n \end{array}\right\} \Rightarrow$$
$$\Rightarrow x_n < x_n - (F-V) \Rightarrow F < V$$

which is a contradiction, since if this were the case the algorithm would have failed on lines 4 - 5. Thus

$$\exists i \in [n] : \min\left(x_{i+1}, F_{cur,i} - (F-V)\right) = F_{cur,i} - (F-V)$$

That is the only $i \in [n]$ such that $F_{cur,i+1} = F - V$, so

$$\forall 0 < k < i, F_{cur,k} = F_{cur,k-1} - x_k \Rightarrow F_{cur,i} = F - \sum_{k=1}^{i-1} x_k \ .$$

Furthermore, since $F_{cur,i+1} = F - V$, it is

$$c'_{i+1} = x_{i+1} - F_{cur,i} + F - V = x_i - F + \sum_{k=1}^{i-1} x_k + F - V \Rightarrow$$
$$\Rightarrow c'_{i+1} = \sum_{k=1}^{i} x_k - V \ ,$$
$$\forall k \leq i, c'_k = 0 \text{ and}$$
$$\forall k > i+1, c'_k = x_k \ .$$

In total, we have

$$\sum_{k=1}^{n} c'_k = \sum_{k=1}^{i} x_k - V + \sum_{k=i+1}^{n} x_k = \sum_{k=1}^{n} x_k - V \Rightarrow \sum_{k=1}^{n} c'_k = F - V \ .$$

Thus the requirement (17) is satisfied. □

Complexity of algorithm `fcfs`
Since i is incremented by 1 on every iteration of the `while` loop (line 13) and $i < n + 1$ at the end of the execution, the complexity of the `while` loop is $O(n)$ in the worst case. The complexity of lines 4 - 5 and 8 is $O(1)$ and the complexity of lines 2 - 3, 6 - 7 and 14 is $O(n)$, thus the total complexity of algorithm 7 is $O(n)$. □

Note that we choose to calculate the complexity of the `length()` function as $O(n)$. Whereas this complexity is implementation-dependent, even with the most naive approaches it cannot be higher than $O(n)$, thus we use this worst-case complexity in our analysis to cover all cases. This approach is implicitly used in all subsequent complexity analyses, but since all complexities are greater than $O(n)$, this approach does not influence them.

Proof of correctness for algorithm `abs`

First of all, we can note that, if $F_{prov} \leq F - V$ in line 12, then the execution will enter the `else` clause of line 16. Therefore, in line 19, F_{cur} will get the value $F - V$, as we can see by executing the lines 17 - 19 by hand. This in turn means that the loop in line 9 will break right after the `else` clause is executed. Furthermore, the assignment in line 15 in combination with the truth of the statement $F_{prov} > F - V$ in line 12 shows that, if the execution enters the `if` clause of line 12, then the loop of line 9 will be executed at least once more. These two observations amount to the fact that the `else` clause will be executed exactly one time and afterwards the `while` loop will break.

We use the notation $F_{cur,0}$, $reduction_0$ and $empty_0$ to refer to the initial values of the corresponding variables, as set in lines 3, 8 and 7 respectively. Furthermore, the notation $empty_j$, $reduction_j$, $F_{cur,j}$ and i_j is used to refer to the values of the corresponding variables after the j-th iteration of the `while` loop. i_j is chosen in line 10. From lines 11, 13, 15, 12 and 17 to 19 we see that

$$F_{cur,j} = \begin{cases} F_{prov,j}, & F_{prov,j} > F - V \\ F - V, & F_{prov,j} \leq F - V \end{cases} \text{, where}$$

$$F_{prov,j} = F_{cur,j-1} - (n - empty_{j-1}) \left(x_{i_j} - x_{i_{j-1}} \right), j \geq 1 \text{ and } x_{i_0} = 0 \ .$$

It is worth noting that the maximum number of iterations is n, or else $j \leq n$. This holds because, if we suppose that $F_{cur,n+1}$ exists, it is

$$F_{cur,n} > F - V \geq 0 \tag{40}$$

However, we can easily see that in this case

$$F_{cur,n} = F_{cur,0} - \sum_{j=1}^{n} (n-(j-1)) \left(x_{i_j} - x_{i_{j-1}}\right) =$$
$$= \sum_{j=1}^{n} x_{i_j} - \sum_{j=1}^{n} (n-(j-1))\, x_{i_j} + \sum_{j=1}^{n-1} (n-j)\, x_{i_j} =$$
$$= \sum_{j=1}^{n} x_{i_j} - \sum_{j=1}^{n-1} [(n-(j-1)) - (n-j)] x_{i_j} - (n-(n-1))\, x_{i_n} =$$
$$= \sum_{j=1}^{n} x_{i_j} - \sum_{j=1}^{n-1} x_{i_j} - x_{i_n} = 0 \ ,$$

which is a contradiction to (40), thus $F_{cur,n+1}$ does not exist and $j \leq n$. This means that `popMin()` will never fail.

We will now show that $\forall j \in [n], empty_j < n$. At line 7, it is $empty_0 = 0 < n$. *empty* is again modified in line 14, where it is incremented by at most 1 at each iteration of the `while` loop (line 9). As we saw above, the iterations cannot exceed n and *empty* is not incremented in the last iteration which consists of the `else` clause, thus $\forall j \in [n], empty_j < n$.

Next, we will show that $\forall i \in [n], c'_i \leq x_i$, as per the requirement (16). From line 23, we see that it suffices to prove that $reduction \geq 0$. In line 8, *reduction* is initialized to 0. In line 13, *reduction* is set to x_i, which is always a non-negative value. The last line where *reduction* is modified is 18. In this line, it is $F_{cur} > F - V$ or else the `while` loop would have broken before beginning this iteration and $n > empty$ as we previously saw. Thus the non-negative variable *reduction* is increased and the resulting value is always positive.

We will finally show that $\sum_{i=1}^{n} c'_i = F - V$, which satisfies the requirement (17). Let $k, 0 \leq k \leq n$ be such that at the end of the execution

$$\forall j \leq k, c_{i_j} = 0 \wedge \forall j > k, c_{i_j} > 0 \ .$$

The following holds:

$$\sum_{j=1}^{n} c'_{i_j} = \sum_{j=k+1}^{n} c'_{i_j} = \sum_{j=k+1}^{n} \left(x_{i_j} - reduction_{k+1} \right) =$$
$$\sum_{j=k+1}^{n} \left(x_{i_j} - \left(x_{i_k} + \frac{F_{cur,k} - (F - V)}{n-k} \right) \right)$$
$$\sum_{j=k+1}^{n} \left(x_{i_j} - \left(x_{i_k} + \frac{F_{cur,0} - \sum_{l=1}^{k} (n - (l-1)) \left(x_{i_l} - x_{i_{l-1}} \right) - F + V}{n-k} \right) \right)$$
$$\sum_{j=k+1}^{n} \left(x_{i_j} - \left(x_{i_k} + \frac{F - \sum_{l=1}^{k} (n-l+1)\, x_{i_l} + \sum_{l=1}^{k-1} (n-l)\, x_{i_l} - F + V}{n-k} \right) \right)$$
$$\sum_{j=k+1}^{n} x_{i_j} - (n-k)\, x_{i_k} - \frac{n-k}{n-k} \left(- \sum_{l=1}^{k-1} x_{i_l} - (n-k+1)\, x_{i_k} + V \right)$$
$$\sum_{j=k+1}^{n} x_{i_j} - (n-k)\, x_{i_k} + \sum_{j=1}^{k-1} x_{i_j} + (n-k+1)\, x_{i_k} - V$$
$$= \sum_{j=1}^{n} x_{i_j} - V = F - V \quad ,$$

thus the desired property holds. □

Complexity of algorithm `abs`

Lines 4 - 5, 7 - 8 and 11 - 19 have a complexity of $O(1)$. Lines 2 - 3 and 22 - 24 have a complexity of $O(n)$. The `while` loop of line 9 is repeated at most n times, as we saw in the proof of correctness. Thus the total complexity is

$$O(preprocess) + O(n)\, O(popMin) \quad .$$

If the flows are first sorted, it would be

$$O(preprocess) = O(quicksort) = O(nlogn) \text{ and}$$
$$O(popMin) = O(1) \quad ,$$

amounting to a total complexity of $O(nlogn)$. In the case a Fibonacci heap is used, it is

$$O(preprocess) = O(FibonacciHeap) = O(n) \text{ and}$$
$$O(popMin) = O(find\text{-}min) + O(delete\text{-}min) = O(logn) \ ,$$

thus the total complexity is again $O(nlogn)$. □

Proof that algorithm `abs` minimizes $||\Delta_i||_\infty$
Let *reduction* be the final value of the corresponding variable. It holds that

$$\forall i \in [n] : c'_i > 0, \Delta_i = reduction \ ,$$
$$\forall i \in [n] : c'_i = 0, \Delta_i = x_i \text{ and}$$
$$\forall i \in [n] : c'_i = 0, reduction \geq x_i \ ,$$

thus we deduce that

$$||\Delta_i||_\infty = \max_{1 \leq i \leq n} (x_i - c'_i) = reduction \ .$$

With the capacity configuration C' resulting from `abs()`, it holds that $\sum_{i=1}^{n} c'_i = F - V$. Suppose that there exists a configuration C_1 that maintains that property:

$$\sum_{i=1}^{n} c_{1,i} = F - V \tag{41}$$

and furthermore

$$||\Delta_{1,i}||_\infty = b < reduction \ . \tag{42}$$

Then it must be

$$\forall i \in [n], \Delta_{1,i} \leq b \Rightarrow \forall i \in [n], c_{1,i} \geq x_i - b \ .$$

Without loss of generality, suppose that x_i are sorted in ascending order. Then

$$\exists k' \in [n] \cup \{0\} : \begin{cases} \forall i \leq k', x_i \leq reduction \\ \forall i > k', x_i > reduction \end{cases}$$
$$\text{and } \exists k_1 \in [n] \cup \{0\} : \begin{cases} \forall i \leq k_1, x_i \leq b \\ \forall i > k_1, x_i > b \end{cases} \ .$$

Since $b < reduction$, it is $k_1 \leq k'$. It is:

$$\forall i \in [k_1], 0 \leq c_{1,i} \leq x_i \text{ and}$$
$$\forall i \in [n] \setminus [k_1], x_i - b \leq c_{1,i} \leq x_i \ .$$

Let all $c_{1,i}$ assume the smallest possible value according to the above restriction. Then

$$\forall i \in [k'] \setminus [k_1], x_i > b \text{ and} \tag{43}$$

$$\sum_{i=1}^{n} c_{1,i} = \sum_{i=k_1+1}^{n} (x_i - b) \overset{(43)}{\geq} \sum_{i=k'+1}^{n} (x_i - b) \overset{(42)}{>}$$

$$> \sum_{i=k'+1}^{n} (x_i - reduction) = \sum_{i=1}^{n} c'_i = F - V \ .$$

We see that even with the minimum possible C_1 configuration, the hypothesis (41) is violated, thus the existence of C_1 is a contradiction. We have thus proven the proposition. □

Proof of correctness for algorithm `prop`

We will first show that $\forall i \in [n], c'_i \leq x_i$. Let $i \in [n]$. According to line 7, which is the only line where c'_i is modified, it is

$$c'_i = x_i - \frac{V}{F} x_i = x_i \left(1 - \frac{V}{F}\right) \ . \tag{44}$$

Since $0 < V \leq F$, it is

$$0 < \frac{V}{F} \leq 1 \Rightarrow 0 \leq 1 - \frac{V}{F} < 1 \ . \tag{45}$$

From (44) and (45), along with the fact that $x_i \geq 0$, it is straightforward to see that $c'_i \leq x_i$, thus the requirement (16) is satisfied.

We will now show that $\sum_{i=1}^{n} c'_i = F - V$. At the end of the execution it is

$$\sum_{i=1}^{n} c'_i = \sum_{i=1}^{n} \left(x_i - \frac{V}{F} x_i\right) = \sum_{i=1}^{n} x_i - \frac{V}{F} \sum_{i=1}^{n} x_i \overset{\sum_{i=1}^{n} x_i = F}{=} F - V \ ,$$

thus the requirement (17) is satisfied. □

Complexity of algorithm `prop`

The complexity of lines 2 - 3, 6 - 7 and 8 is $O(n)$ and the complexity of lines 4 - 5 is $O(1)$, thus the total complexity of the algorithm is $O(n)$. □

Proof of Lemma 4: maxFlow Monotonicity

Supppose that

$$\exists \delta_1, \delta_2 : \delta_1 < \delta_2 \wedge maxFlow(\delta_1) \leq maxFlow(\delta_2) < F \ .$$

We choose X'_1, X'_2 such that

$$\forall i \in [n], x'_{1,i} \leq x_i \wedge x'_{2,i} \leq x_i \ .$$

This is always possible because we can derive the maximum flows X'_1 and X'_2 starting with X and reducing the flow along paths beginning from edges with capacities $c'_{1,i} < x_i$ and $c'_{2,i} < x_i$ respectively, until the flows become valid.

Define $MinCut\,(\delta)$ as the minimum cut set of the $MaxFlow\,(\delta)$ configuration. Let

$$S_j = \{i \in [n] : v_i \in N^+\,(A) \cap MinCut\,(\delta_j)\}, j \in \{1, 2\} \ .$$

It holds that $S_j \neq \emptyset$. Suppose that $S_j = \emptyset$. Since $F > F'_j$, there exists a path from A to B on G'_j with positive flow not used, thus X'_j is not the maximum flow, which is a contradiction. Thus $S_j \neq \emptyset, j \in \{1, 2\}$.

Moreover, it holds that $S_1 \subseteq S_2$, since $\forall i \in [n], c'_{2,i} \leq c'_{1,i}$. More precisely, it is

$$\forall i \in [n] : c'_{2,i} > 0, c'_{2,i} < c'_{1,i} \ .$$

Every node in the $MinCut\,(\delta_j)$ is saturated, thus

$$\forall i \in S_1, x'_{j,i} = c'_{j,i}, j \in \{1, 2\} \ .$$

Thus $\sum\limits_{i \in S_1} x'_{2,i} < \sum\limits_{i \in S_1} x'_{1,i}$ and, since $maxFlow(\delta_1) \leq maxFlow(\delta_2)$, we conclude that for X'_1, X'_2 it is $\sum\limits_{i \in [n] \setminus S_1} x'_{2,i} > \sum\limits_{i \in [n] \setminus S_1} x'_{1,i}$. However, since $x'_{i,j} \leq x_i, j \in \{1, 2\}$, the configuration X'' such that

$$\forall i \in S_1, x''_i = x'_{1,i}$$
$$\forall i \in [n] \setminus S_1, x''_i = x'_{2,i}$$

is a valid flow configuration for C'_1 and then

$$F'_1 \geq \sum_{i \in S_1} x''_i + \sum_{i \in [n] \setminus S_1} x''_i = \sum_{i \in S_1} x'_{1,i} + \sum_{i \in [n] \setminus S_1} x'_{2,i} > maxFlow\,(\delta_1) \ \ ,$$

which is a contradiction because $F'_1 = maxFlow\,(\delta_1)$ by the hypothesis. Thus $maxFlow\,(\delta_1) > maxFlow\,(\delta_2)$ and, since δ_1, δ_2 were chosen arbitrarily with the restriction $\delta_1 < \delta_2$, we deduce that the proposition holds. □

We will now prove that `BinSearch` returns the desired δ^* when we provide it with an appropriate interval as input.

Proof of correctness for algorithm `BinSearch`
Suppose that

$$\left[F' - \epsilon_1, F' + \epsilon_2\right] \subset \left[maxFlow\left(top\right), maxFlow\left(bot\right)\right] \quad .$$

We will prove that

$$maxFlow\left(\delta^*\right) \in \left[F' - \epsilon_1, F' + \epsilon_2\right] \quad .$$

First of all, we should note that if an invocation of `BinSearch` returns without calling `BinSearch` again (line 3 or 13), its return value will be equal to the return value of the initial invocation of `BinSearch`, as we can see on lines 9 and 11, where the return value of the invoked `BinSearch` is returned without any modification. The case where `BinSearch` is called recursively is analyzed next:

If $maxFlow\left(\frac{top+bot}{2}\right) < F' - \epsilon_1$ (line 8) then, by the maxFlow monotonicity lemma, $\delta^* \in \left[bot, \frac{top+bot}{2}\right)$. As we see on line 9, the interval $\left(\frac{top+bot}{2}, top\right]$ is discarded when the next `BinSearch` is recursively called. Since $F' + \epsilon_2 \leq maxFlow\left(bot\right)$, we have

$$\left[F' - \epsilon_1, F' + \epsilon_2\right] \subset \left[maxFlow\left(\frac{top + bot}{2}\right), maxFlow\left(bot\right)\right]$$

and the length of the available interval is divided by 2.

Similarly, if $maxFlow\left(\frac{top+bot}{2}\right) > F' + \epsilon_2$ (line 10) then, by the maxFlow monotonicity lemma, it holds that $\delta^* \in \left(\frac{top+bot}{2}, top\right]$. According to line 11, the interval $\left[bot, \frac{top+bot}{2}\right)$ is discarded when the next `BinSearch` is recursively called. Since $F' - \epsilon_1 \geq maxFlow\left(top\right)$, we have

$$\left[F' - \epsilon_1, F' + \epsilon_2\right] \subset \left[maxFlow\left(top\right), maxFlow\left(\frac{top + bot}{2}\right)\right]$$

and the length of the available interval is divided by 2.

As we saw, it holds that

$$\left[F' - \epsilon_1, F' + \epsilon_2\right] \subset \left[maxFlow\left(top\right), maxFlow\left(bot\right)\right]$$

in every recursive call and $top - bot$ is divided by 2 in every call. It is $|[F' - \epsilon_1, F' + \epsilon_2]| = \epsilon_1 + \epsilon_2$. Let bot_0, top_0 the input values given to the initial invocation of `BinSearch`, bot_j, top_j the input values given to the j-th recursive call of `BinSearch` and $len_j = |[bot_j, top_j]| = top_j - bot_j$. We have

$$\forall j > 0, len_j = top_j - bot_j = \frac{top_{j-1} - bot_{j-1}}{2} \Rightarrow$$
$$\forall j > 0, len_j = \frac{top_0 - bot_0}{2^j} \ .$$

We understand that in the worst case

$$len_j = \epsilon_1 + \epsilon_2 \Rightarrow 2^j = \frac{top_0 - bot_0}{\epsilon_1 + \epsilon_2} \Rightarrow j = \log_2(\frac{top_0 - bot_0}{\epsilon_1 + \epsilon_2}) \ .$$

Also, as we saw earlier, δ^* is always in the available interval, thus it holds that

$$maxFlow\left(\delta^*\right) \in \left[F' - \epsilon_1, F' + \epsilon_2\right] \ .$$

□

Complexity of algorithm `BinSearch`
Lines 2 - 3 have complexity $O\left(1\right)$, lines 6 - 7 have complexity $O\left(n\right)$, lines 8 - 13 have complexity $O\left(maxFlow\right) + O\left(BinSearch\right)$. As we saw in the proof of correctness for `BinSearch`, we need at most $\log_2\left(\frac{top-bot}{\epsilon_1+\epsilon_2}\right)$ recursive calls of `BinSearch`. Thus the function `BinSearch` has worst-case complexity

$$O\left(\left(maxFlow + n\right)\log_2\left(\frac{top - bot}{\epsilon_1 + \epsilon_2}\right)\right) \ .$$

□

Proof of correctness for algorithm `dinfmin`
Let $C' =$ `dinfmin(`c_i`, V, `ϵ_1`, `ϵ_2`)`. We will show that

$$F' \in [F - V - \epsilon_1, F - V + \epsilon_2] \ .$$

We can easily see that

$$maxFlow\left(0\right) = F \text{ and}$$
$$maxFlow\left(\max_{i\in[n]}\{c_i\}\right) = 0 \ ,$$

thus $\delta^* \in \left(0, \max_{i\in[n]}\{c_i\}\right)$. From the proof of correctness for `BinSearch`, we know that $maxFlow\left(\delta^*\right) \in [F - V - \epsilon_1, F - V + \epsilon_2]$, given that ϵ_1, ϵ_2 are chosen so that $F - V - \epsilon_1 \geq 0, F - V + \epsilon_2 \leq F$, so as to satisfy the condition $[F' - \epsilon_1, F' + \epsilon_2] \subset [maxFlow\left(top\right), maxFlow\left(bot\right)]$. The last condition is always met due to lines 6 - 8. □

Complexity of algorithm `dinfmin`
The complexity of lines 4 - 5 and 6 - 8 is $O\left(1\right)$, the complexity of lines 2, 3, 9, 11 - 12 and 13 is $O\left(n\right)$ and the complexity of line 10 is $O\left(BinSearch\right) = O\left(\left(maxFlow + n\right)\log_2\left(\frac{\delta_{max}}{\epsilon_1+\epsilon_2}\right)\right)$, thus the total complexity of dinfmin is $O\left(\left(maxFlow + n\right)\log_2\left(\frac{\delta_{max}}{\epsilon_1+\epsilon_2}\right)\right)$. □

3 Algorithms

This algorithm calls the necessary functions to prepare the new graph.

```
Execute Turn
Input : old graph G_{j-1}, player A ∈ V_{j-1}, old capital
    Cap_{A,j-1}, TentativeTurn
Output : new graph G_j, new capital Cap_{A,j}, new history H_j
1  executeTurn(G_{j-1}, A, Cap_{A,j-1}, TentativeTurn) :
2    (Turn_j, NewCap) = validateTurn(G_{j-1}, A, Cap_{A,j-1},
       TentativeTurn)
3    return(commitTurn(G_{j-1}, A, Turn_j, NewCap))
```

The following algorithm validates that the tentative turn produced by the strategy respects the rules imposed on turns. If the turn is invalid, an empty turn is returned.

```
Validate Turn
Input : old G_{j-1}, player A ∈ V_{j-1}, old Cap_{A,j-1}, Turn
Output : Turn_j, new Cap_{A,j}
1  validateTurn(G_{j-1}, A, Cap_{A,j-1}, Turn) :
2    Y_st = Y_add = 0
3    Stolen = Added = ∅
4    for (action ∈ Turn)
5      action match do
6        case Steal(y, w) do
7          if (y > DTr_{w→A,j-1} or y < 0 or w ∈ Stolen)
```

```
8           return(∅, Cap_{A,j−1})
9         else
10          Y_st += y
11          Stolen = Stolen ∪ {w}
12      case Add(y, w) do
13        if (y < -DTr_{A→w,j−1} or w ∈ Added)
14          return(∅, Cap_{A,j−1})
15        else
16          Y_add += y
17          Added = Added ∪ {w}
18  if (Y_add - Y_st > Cap_{A,j−1})
19    return(∅, Cap_{A,j−1})
20  else
21    return(Turn, Cap_{A,j−1} + Y_st − Y_add)
```

Finally, this algorithm applies the turn to the old graph and returns the new graph, along with the updated capital and history.

```
Commit Turn
Input : old G_{j−1}, player A ∈ V_{j−1}, NewCap, Turn_j
Output : new G_j, new Cap_{A,j}, new H_j
commitTurn(G_{j−1}, A, NewCap, Turn_j) :
  for ((v, w) ∈ E_j)
    DTr_{v→w,j} = DTr_{v→w,j−1}
  for (action ∈ Turn_j)
    action match do
      case Steal(y, w) do
        DTr_{w→A,j} = DTr_{w→A,j−1} − y
      case Add(y, w) do
        DTr_{A→w,j} = DTr_{A→w,j−1} + y
  Cap_{A,j} = NewCap; H_j = (A, Turn_j)
  return(G_j, Cap_{A,j}, H_j)
```

It is straightforward to verify the compatibility of the previous algorithms with the corresponding definitions.

References

1. Sanchez W.: Lines of Credit. https://gist.github.com/drwasho/2c40b91e169f55988618#part-3-web-of-credit (2016)
2. Nakamoto S.: Bitcoin: A Peer-to-Peer Electronic Cash System (2008)
3. Antonopoulos A. M.: Mastering Bitcoin: Unlocking Digital Cryptocurrencies. O'Reilly Media, Inc. (2014)
4. Karlan D., Mobius M., Rosenblat T., Szeidl A.: Trust and social collateral. The Quarterly Journal of Economics, pp. 1307-1361 (2009)
5. Cormen T. H., Leiserson C. E., Rivest R. L., Stein C.: Introduction to Algorithms (3rd ed.). MIT Press and McGraw-Hill (2009)
6. Orlin J. B.: Max Flows in O(nm) Time, or Better. STOC '13 Proceedings of the forty-fifth annual ACM symposium on Theory of computing, pp.765-774, ACM, New York, doi:10.1145/2488608.2488705 (2013)
7. Douceur J. R.: The Sybil Attack. International workshop on Peer-To-Peer Systems (2002)
8. Illés T., Terlaky T.: Pivot versus interior point methods: Pros and cons. European Journal of Operational Research, 140(2), 170-190 (2002)
9. Zimmermann P.: PGP Source Code and Internals. The MIT Press (1995)
10. Clarke I., Sandberg O., Wiley B., Hong T. W.: Freenet: A Distributed Anonymous Information Storage and Retrieval System. H. Federrath, Designing Privacy Enhancing Technologies pp. 46-66, Berkeley, USA: Springer-Verlag Berlin Heidelberg (2001)
11. What is a Public Key Infrastructure? http://www.net-security-training.co.uk/what-is-a-public-key-infrastructure/
12. Post A., Shah V., Mislove A.: Bazaar: Strengthening User Reputations in Online Marketplaces. Proceedings of NSDI'11: 8th USENIX Symposium on Networked Systems Design and Implementation, p. 183 (2011)
13. Lamport L., Shostak R., Pease M.: The Byzantine Generals Problem. ACM Transactions on Programming Languages and Systems (TOPLAS) 4.3, pp. 382-401 (1982)
14. Mui L., Mohtashemi M., Halberstadt A.: A Computational Model of Trust and Reputation. System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference, pp. 2431-2439 IEEE (2002)
15. Huynh T. D., Jennings N. R., Shadbolt N. R.: An Integrated Trust and Reputation Model for Open Multi-Agent Systems. Autonomous Agents and Multi-Agent Systems, 13(2), pp. 119-154 (2006)
16. Michiardi P., Molva R.: Core: a Collaborative Reputation Mechanism to Enforce Node Cooperation in Mobile Ad-hoc Networks. Advanced Communications and Multimedia Security, pp. 107-121, Springer US (2002)
17. Cannon L.: Open Reputation: the Decentralized Reputation Platform (2015) https://openreputation.net/open-reputation-high-level-whitepaper.pdf
18. Grünert A., Hudert S., König S., Kaffille S., Wirtz G.: Decentralized Reputation Management for Cooperating Software Agents in Open Multi-Agent Systems. ITSSA, 1(4), pp. 363-368 (2006)
19. Repantis T., Kalogeraki V.: Decentralized Trust Management for Ad-hoc Peer-to-Peer Networks. Proceedings of the 4th International Workshop on Middleware for Pervasive and Ad-hoc Computing, MPAC 2006, p. 6, ACM (2006)
20. Commerce B. E., Jøsang A., Ismail R.: The Beta Reputation System. Proceedings of the 15th Bled Electronic Commerce Conference (2002)

21. Suryanarayana G., Erenkrantz J. R., Taylor R. N.: An Architectural Approach for Decentralized Trust Management. IEEE Internet Computing, 9(6), pp. 16-23 (2005)
22. Visan A., Pop F., Cristea V.: Decentralized Trust Management in Peer-to-Peer Systems. 10th International Symposium on Parallel and Distributed Computing, pp. 232-239, IEEE (2011)
23. Suryanarayana G., Diallo M., Taylor R. N.: A Generic Framework for Modeling Decentralized Reputation-Based Trust Models. 14th ACM SigSoft Symposium on Foundations of Software Engineering (2006)
24. Caronni G.: Walking the web of trust. Enabling Technologies: Infrastructure for Collaborative Enterprises, WET ICE 2000, Proceedings, IEEE 9th International Workshops, pp. 153-158 (2000)
25. Penning H.P.: PGP pathfinder pgp.cs.uu.nl
26. Soska K., Kwon A., Christin N., Devadas S.: Beaver: A Decentralized Anonymous Marketplace with Secure Reputation (2016)
27. Zindros D. S.: Trust in Decentralized Anonymous Marketplaces (2015)
28. DeFigueiredo D. D. B., Barr E. T.: TrustDavis: A Non-Exploitable Online Reputation System. CEC, Vol. 5, pp. 274-283 (2005)
29. Fugger R.: Money as IOUs in Social Trust Networks & A Proposal for a Decentralized Currency Network Protocol. http://archive.ripple-project.org/decentralizedcurrency.pdf (2004)
30. Narayanan A., Shmatikov V.: De-anonymizing Social Networks. SP '09 Proceedings of the 2009 30th IEEE Symposium on Security and Privacy, pp. 173-187, 10.1109/SP.2009.22 (2009)
31. Konforty D., Adam Y., Estrada D., Meredith L. G.: Synereo: The Decentralized and Distributed Social Network (2015)
32. Ahuja R. K., Magnanti T. L., Orlin J. B.: Network Flows: Theory, Algorithms, and Applications. Prentice-Hall (1993) https://ocw.mit.edu. License: Creative Commons BY-NC-SA. (Fall 2010)
33. Jøsang A., Ismail R., Boyd C.: A Survey of Trust and Reputation Systems for Online Service Provision. Decision Support Systems, 43(2), pp. 618-644 (2007)
34. Johnson-George C., Swap W. C.: Measurement of specific interpersonal trust: Construction and validation of a scale to assess trust in a specific other. Journal of personality and social psychology, 43(6), 1306 (1982)
35. Mayer R. C., Davis J. H., Schoorman, F. D.: An integrative model of organizational trust. Academy of management review, 20(3), 709-734 (1995)
36. Meulpolder M., Pouwelse J., Epema D., Sips, H.: Bartercast: Fully distributed sharing-ratio enforcement in bittorrent. Delft University of Technology-Parallel and Distributed Systems Report Series (2008)
37. Yu H., Kaminsky M., Gibbons P. B., Flaxman A.: Sybilguard: Defending Against Sybil Attacks via Social Networks. ACM SIGCOMM Computer Communication Review (Vol. 36, No. 4, pp. 267-278), ACM (2006, September)