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
orfeas.litos@hotmail.com, 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 in which 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.

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

⋆ We thank Kyriakos Axiotis for his insights on flows.
⋆⋆ Research supported by ERC project CODAMODA, project #259152

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]. 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].

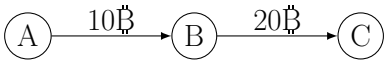

Fig.1: A trusts C 10฿

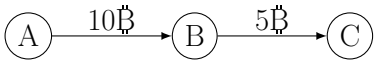

Fig.2: A trusts C 5฿

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 [4] 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 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* $\mathcal{V}_j = V(\mathcal{G}_j)$ *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 capital of A at the end of turn j, $Cap_{A,j}$, is defined as the total value that exists in P2PKH in the UTXO and can be spent by A at the end of turn j.*

A rational player would like to maximize her capital in the long term. The formal definition of direct trust follows:

Definition 4 (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.*

$$DTr_{A\to B,j} = \begin{cases} c_j\left(A,B\right), & if\ (A,B) \in \mathcal{E}_j \\ 0, & else \end{cases} \tag{1}$$

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 Appendix. An example graph with its corresponding transactions in the UTXO can be seen below.

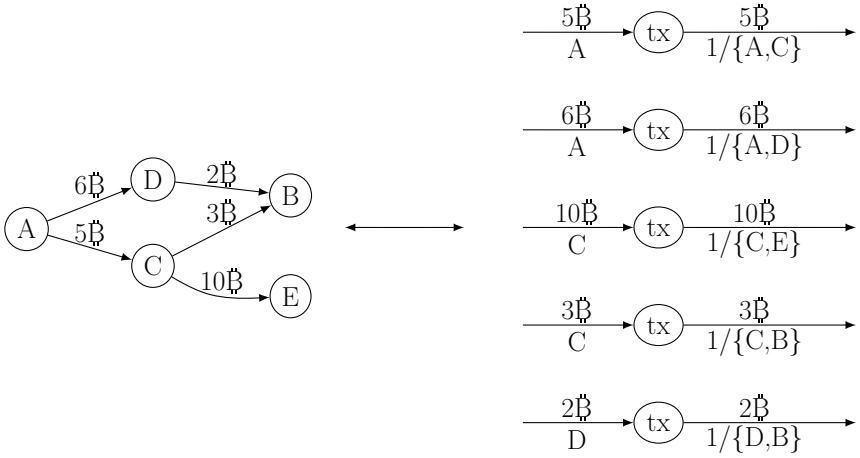

Fig.3: TrustIsRisk Game Graph and Equivalent Bitcoin UTXO

3 Evolution of Trust

Definition 5 (Turns). *The game we are describing is turn-based. In each turn j exactly one player $A \in \mathcal{V}, A = Player\,(j)$, chooses an action (according to a certain strategy) that can be one of the following, or a finite combination thereof:*

1. *Steal value y_B from $B \in N^-\,(A)_{j-1}$, where $0 \leq y_B \leq DTr_{B\to A,j-1}$. Then it is:*

$$DTr_{B\to A,j} = DTr_{B\to A,j-1} - y_B \;\; (Steal\,(y_B, B)) \tag{2}$$

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

$$DTr_{A\to B,j} = DTr_{A\to B,j-1} + y_B \;\; (Add\,(y_B, B)) \tag{3}$$

   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

$$Y_{add} - Y_{st} \leq Cap_{A,j-1} \;\; . \tag{4}$$

Capital is updated in every turn:

$$Cap_{A,j} = Cap_{A,j-1} + Y_{st} - Y_{add} \;\; . \tag{5}$$

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

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

Definition 6 (Damage). *Let j be a turn such that $Player\,(j) = A$.*

$$Damage_{A,j} = out_{A,prev(j)} - out_{A,j-1} \tag{6}$$

We say that A has been stolen value $Damage_{A,j}$ between $prev\,(j)$ and j if $Damage_{A,j} > 0$. If turns are not specified, we implicitly refer to the current and the previous turns.

Definition 7 (History). *We define History, $\mathcal{H} = (\mathcal{H}_j)$, as the sequence of all the tuples containing the sets of actions and the corresponding player.*

$$\mathcal{H}_j = (Player\,(j)\,, Turn_j) \tag{7}$$

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

$$Turn_1 = \{Steal\,(1, B)\,, Add\,(4, C)\}\ . \tag{8}$$

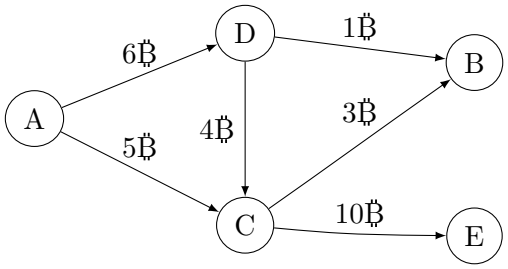

Fig.4: Game Graph after $Turn_1$ (8) passes on the Graph of Figure 3

In its initial form 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 = j + 1
  v ←$ V_j
  ProvisionalTurn = vOracle(G_0, v, (H)_{1...j-1})
  (G_j, Cap_{v,j}, H_j) = executeTurn(G_{j-1}, v, Cap_{v,j-1},
      ProvisionalTurn)
```

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}, ProvisionalTurn
Output : new graph G_j, new capital Cap_{A,j}, new history H_j
```

```
executeTurn(G_{j-1}, A, Cap_{A,j-1}, ProvisionalTurn) :
  (Turn_j, NewCap) = validateTurn(G_{j-1}, A, Cap_{A,j-1},
     ProvisionalTurn)
  return(commitTurn(G_{j-1}, A, Turn_j, NewCap))
```

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

```
Validate Turn
Input : old graph G_{j-1}, player A ∈ V_{j-1}, old capital
    Cap_{A,j-1}, ProvisionalTurn
Output : Turn_j, new capital Cap_{A,j}
validateTurn(G_{j-1}, A, Cap_{A,j-1}, ProvisionalTurn) :
  Y_st = 0
  Y_add = 0
  for (action ∈ ProvisionalTurn)
    action match do
      case Steal(y, w) do
        if (y > DTr_{w→A,j-1} || y < 0)
          return(∅, Cap_{A,j-1})
        else
          Y_st = Y_st + y
      case Add(y, w) do
        if (y < -DTr_{A→w,j-1})
          return(∅, Cap_{A,j-1})
        else
          Y_add = Y_add + y
  if (Y_add - Y_st > Cap_{A,j-1})
    return(∅, Cap_{A,j-1})
  else
    return(ProvisionalTurn, 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 graph G_{j-1}, player A ∈ V_{j-1}, Turn_j, NewCap
Output : new graph G_j, new capital Cap_{A,j}, new history H_j
commitTurn(G_{j-1}, A, Turn_j, NewCap) :
  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.

4 Trust Transitivity

In this section we define some strategies, along with their oracles. 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 8 (Idle Strategy). *A player A is said to follow the idle strategy if she passes in her turn.*

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

Definition 9 (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.*

```
Evil Oracle
Input : initial graph G_0, player A, history (H)_{1...j−1}
Output : Turn_j
evilOracle(G_0, A, H) :
  Steals = ⋃_{v∈N^−(A)_{j−1}} {Steal(DTr_{v→A,j−1}, v)}
  Adds = ⋃_{v∈N^+(A)_{j−1}} {Add(−DTr_{A→v,j−1}, v)}
  Turn_j = Steals ∪ Adds
  return(Turn_j)
```

Definition 10 (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.*

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

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

$$\sum_{v \in N^-(A)_{j-1}} y_v = Damage_{A,j} \wedge \forall v \in N^-(A)_{j-1}, y_v \leq DTr_{v \to A, j-1} \ . \quad (9)$$

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

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 $\mathcal{V}$. These conventions will hold whenever we use the Transitive Game.

```
Transitive Game
Input : graph G_0, A ∈ V idle player, E ∈ V evil player
Output : history H  #can remove output, this runs forever.
Angry = Sad = ∅
Happy = V \ {A, E}
for (v ∈ V \ {E})
  Loss_v = 0
j = 0
while (True)
  j = j + 1
  v <-$ V \ {A}
  Turn_j = vOracle(G_0, v, (H)_{1...j-1})
  executeTurn(G_{j-1}, Cap_{v,j-1}, Turn_j)
  for (action ∈ Turn_j)
    action match do
      case Steal(y, w) do
        exchange = y
        Loss_w = Loss_w + exchange
        if (v != E)
          Loss_v = Loss_v - exchange
        if (w != A)
          Happy = Happy \ {w}
          if (in_{w,j} == 0)
            Sad = Sad ∪ {w}
          else
            Angry = Angry ∪ {w}
```

```
24   if (v != E)
25     Angry = Angry \ {v}
26     if (Loss_v > 0)
27       Sad = Sad ∪ {v}                    #in_{v,j} should be zero
28     if (Loss_v == 0)
29       Happy = Happy ∪ {v}
```

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 Appendix (Theorem 6) for a formal proof of this simple fact). Moreover, let $v = Player(j)$ and $j' = prev\,(j)$. Given that

$$Damage_{v,j} = out_{v,j'} - out_{v,j-1} \ , \tag{10}$$

the algorithm generates turns:

$$Turn_j = \begin{cases} \emptyset, & Damage_{v,j} = 0 \\ \bigcup\limits_{w \in N^-(v)_{j-1}} \{Steal\,(y_w, w)\}, & Damage_{v,j} > 0 \end{cases} \ . \tag{11}$$

In the second case, it is

$$\sum_{w \in N^-(v)_{j-1}} y_w = \min\left(in_{v,j-1}, Damage_{v,j}\right) \ . \tag{12}$$

From the definition of $Damage_{v,j}$ and knowing that no strategy in this case can increase any direct trust, it is obvious 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. An example follows:

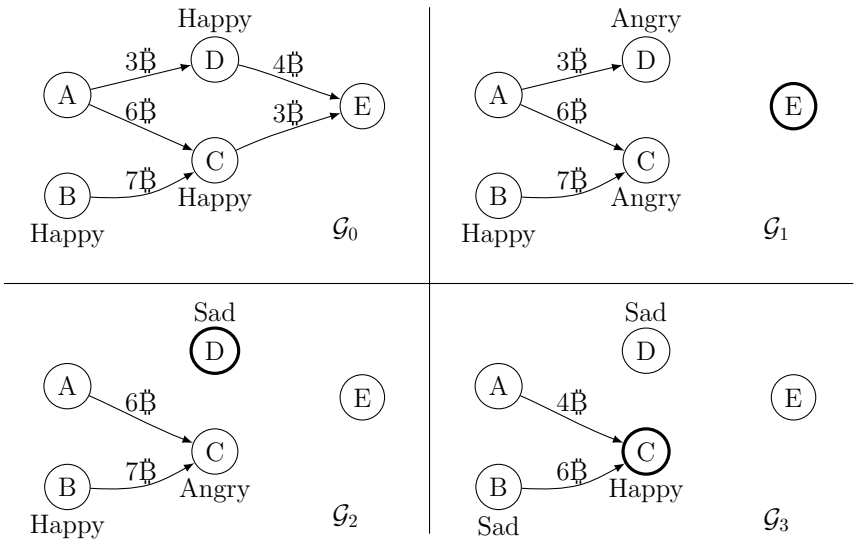

Fig.5: Turns of a `TransitiveGame(`$\mathcal{G}_0$`,`A`,`E`)`

5 Trust Flow

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

Definition 11 (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 obvious that $Tr_{A\to B} \geq DTr_{A\to B}$. The following theorem establishes that $Tr_{A\to B}$ is always finite.

Theorem 1 (Trust Convergence Theorem).
Consider a Transitive Game. There exists a turn j' such that

$$\forall j \geq j', Turn_j = \emptyset \ . \tag{13}$$

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. For the complete proof, see Appendix (Proof 2). □

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 $X = [x_{vw}]_{V \times V}$ with a source A and a sink B is valid when:

$$\forall (v, w) \in E, x_{vw} \leq c_{vw} \tag{14}$$

and

$$\forall v \in V \setminus \{A, B\}, \sum_{w \in N^+(v)} x_{wv} = \sum_{w \in N^-(v)} x_{vw} \ . \tag{15}$$

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 $O\,(|V||E|)$ 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

$$maxFlow\,(A, E) \leq Loss_A \ . \tag{16}$$

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. For the complete proof, see Appendix (Proof 3). □

Lemma 2 (Transitive Games Are Flows).
Let $\mathcal{H} = \texttt{TransitiveGame}(\mathcal{G}, A, E)$ *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*

$$\sum_{v\in\mathcal{V}} x_{Av} = Loss_A \ . \tag{17}$$

Proof Sketch. If we exclude the sad players from the game, the *Steal* () actions that remain constitute a valid flow from A to E. For the complete proof, see Appendix (Proof 4). □

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

$$Tr_{A\to B} = maxFlow\,(A, B) \ . \tag{18}$$

Proof.
From lemma 1 we see that there exists an execution of the Transitive Game such that

$$Loss_A = maxFlow\,(A, B) \ . \tag{19}$$

Since $Tr_{A\to B}$ is the maximum loss that A can suffer after the convergence of the Transitive Game, we see that

$$Tr_{A\to B} \geq maxFlow\,(A, B) \ . \tag{20}$$

Moreover, there exists an execution of the Transitive Game such that

$$Tr_{A\to B} = Loss_A \ . \tag{21}$$

From lemma 2, this execution corresponds to a flow. Thus

$$Tr_{A\to B} \leq maxFlow\,(A, B) \ . \tag{22}$$

The theorem follows from (20) and (22). □

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). *Let $\mathcal{G}$ game graph, $A, B \in \mathcal{V}$ and V the desired value to be transferred from A to B, with $V \leq Tr_{A\to B}$. Let also $\mathcal{G}'$ such that*

$$\mathcal{V}' = \mathcal{V} \tag{23}$$

$$\forall v \in \mathcal{V}' \setminus \{A\}, \forall w \in \mathcal{V}', DTr'_{v\to w} = DTr_{v\to w} \ . \tag{24}$$

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

$$Tr'_{A\to B} = Tr_{A\to B} - V \ . \tag{25}$$

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

$$DTr''_{A\to B} = DTr'_{A\to B} + V \ . \tag{26}$$

It then holds that

$$Tr''_{A\to B} = Tr_{A\to B} \ . \tag{27}$$

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

It is intuitively obvious that it is possible for A to reduce her outgoing direct trust in a manner that achieves (25), since $maxFlow\,(A, B)$ is continuous with respect to A's outgoing direct trusts. We leave this calculation as part of further research.

Proof 1. (Risk Invariance Theorem (3)) Let

$$\forall v, w \in \mathcal{V}', c'_{vw} = DTr'_{v\to w} \text{ and} \tag{28}$$

$$\forall v, w \in \mathcal{V}'', c''_{vw} = DTr''_{v\to w} \ . \tag{29}$$

Then

$$\forall v, w \in \mathcal{V}, c'_{vw} \leq c''_{vw} \tag{30}$$

and any valid flow on $\mathcal{G}'$ is a valid flow on $\mathcal{G}''$ as well. Furthermore, any $MaxFlow\,(A, B)$ chooses $x_{AB} = c_{AB}$, thus

$$x''_{AB} = x'_{AB} + V \ . \tag{31}$$

From (30) and (31) we see that

$$maxFlow_{\mathcal{G}''}\,(A, B) \geq maxFlow_{\mathcal{G}'}\,(A, B) + V \ . \tag{32}$$

Now suppose that

$$maxFlow_{\mathcal{G}''}(A,B) > maxFlow_{\mathcal{G}'}(A,B) + V \ . \tag{33}$$

Then

$$\sum_{v \in N^-(B)'' \setminus \{A\}} x''_{vB} > \sum_{v \in N^-(B)' \setminus \{A\}} x'_{vB} \ . \tag{34}$$

However, it holds that

$$\forall e \in \mathcal{V} \setminus \{(A,B)\}, c'_e = c''_e \ , \tag{35}$$

and x_{AB} flows directly from A to B without adding to the incoming or outgoing flow of any intermediate node, thus $MaxFlow_{\mathcal{G}''}$ can choose

$$\forall e \in \mathcal{V} \setminus \{(A,B)\}, x''_e = x'_e \tag{36}$$

and thus, by contradiction with (33), it holds that

$$maxFlow_{\mathcal{G}''}(A,B) \leq maxFlow_{\mathcal{G}'}(A,B) + V \ . \tag{37}$$

From (32) and (37) we get

$$maxFlow_{\mathcal{G}''}(A,B) = maxFlow_{\mathcal{G}'}(A,B) + V \ . \tag{38}$$

Finally, it holds that

$$\begin{gathered} Tr''_{A\to B} = maxFlow_{\mathcal{G}''}(A,B) \overset{(38)}{=} \\ = maxFlow_{\mathcal{G}'}(A,B) + V = Tr'_{A\to B} + V \overset{(25)}{=} Tr_{A\to B} \ . \end{gathered} \tag{39}$$

The proposition is proved. □

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 12 (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, if $S \subset \mathcal{V}$,*

$$\begin{gathered} Strategy(A) = Idle \wedge \forall E \in S, Strategy(E) = Evil \wedge \\ \wedge \forall v \in \mathcal{V} \setminus (S \cup \{A\}), Strategy(v) = Conservative \end{gathered} \tag{40}$$

and choices are the different actions between which the conservative players can choose, then

$$Tr_{A\to S,j} = \max_{j':j'>j,configurations} \left[out_{A,j} - out_{A,j'}\right] \tag{41}$$

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

$$\forall B \in S, DTr_{B\to T} = \infty \ . \tag{42}$$

It holds that

$$\forall A \in \mathcal{V} \setminus S, Tr_{A\to S} = maxFlow\left(A, T\right) \ . \tag{43}$$

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 13 (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 14 (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 15 (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.*

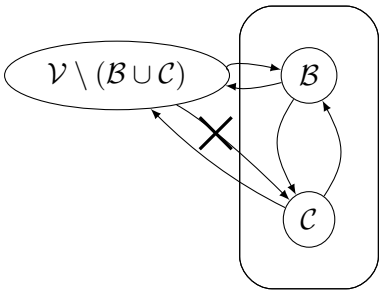

Fig.6: Collusion

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*

$$Tr_{A \to \mathcal{B} \cup \mathcal{C}} = Tr_{A \to \mathcal{B}} \ . \tag{44}$$

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. For the complete proof, see Appendix (Proof 5). □

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

With this we have successfully delivered our promise for a Sybil-resilient decentralized financial trust system with invariant risk when making a purchase.

7 Related Work

8 Further Research

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

First of all, concrete trust manipulation algorithms are needed to make use of Risk Invariance Theorem. Secondly, an extension of this work to a

dynamic setting where players can enter, leave and execute turns simultaneously and where there is no need for an algorithm like TrustIsRisk Game. Furthermore, the fact that $MaxFlow$ needs full network knowledge may be undesirable for some parties, consequently there should be further research on zero knowledge methods. Moreover, extended game theoretic analysis for cases more complex than the Transitive Game is needed to expand our comprehension on the proposed system. Obviously an implementation of the wallet is necessary to make the system available and related experimental results can give more insight on its dynamics. Finally, alternative multisig types, such as 2-of-3 can be explored.

Appendix

8.1 Common Notation

Definition 16 (Assets). *Sum of A's capital and outgoing trust.*

$$As_{A,j} = Cap_{A,j} + out_{A,j} \tag{45}$$

Definition 17 (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,*

$$N^+(A)_j = \{B \in \mathcal{V}_j : DTr_{A\to B,j} > 0\} \ . \tag{46}$$

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

$$N^+(S)_j = \bigcup_{A\in S} N^+(A)_j \tag{47}$$

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,*

$$N^-(A)_j = \{B \in \mathcal{V}_j : DTr_{B\to A,j} > 0\} \ . \tag{48}$$

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

$$N^-(S)_j = \bigcup_{A\in S} N^-(A)_j \tag{49}$$

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,*

$$N(A)_j = N^+(A)_j \cup N^-(A)_j \ . \tag{50}$$

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

$$N(S)_j = N^+(S)_j \cup N^-(S)_j \tag{51}$$

Definition 18 (Total Incoming/Outgoing Trust).

$$in_{A,j} = \sum_{v \in N^-(A)_j} DTr_{v \to A,j} \tag{52}$$

$$out_{A,j} = \sum_{v \in N^+(A)_j} DTr_{A \to v,j} \tag{53}$$

Let $A = Player(j)$. $Turn_j$ Examples:

1.
$$Turn_j = \emptyset \tag{54}$$

2.
$$Turn_j = \{Steal(y, B), Add(w, B)\} \ , \tag{55}$$

given that

$$DTr_{B \to A, j_2 - 1} \leq y \wedge -DTr_{A \to B, j_2 - 1} \leq w \wedge y - w \leq Cap_{A, j_2 - 1} \ . \tag{56}$$

3.
$$Turn_j = \{Steal(x, B), Add(y, C), Add(w, D)\} \ , \tag{57}$$

given that

$$\begin{gathered} DTr_{B \to A, j_3 - 1} \leq x \wedge -DTr_{A \to C, j_3 - 1} \leq y \wedge \\ \wedge -DTr_{A \to D, j_3 - 1} \leq w \wedge x - y - w \leq Cap_{A, j_3 - 1} \ . \end{gathered} \tag{58}$$

4.
$$Turn_j = \{Steal(x, B), Steal(y, B)\} \tag{59}$$

is not a valid turn because it contains two $Steal()$ actions against the same player. If

$$x + y \leq DTr_{B \to A} \ , \tag{60}$$

the correct alternative would be

$$Turn_j = \{Steal(x + y, B)\} \ . \tag{61}$$

Definition 19 (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*

$$P = \{k \in \mathbb{N} : k < j \wedge Player(k) = A\} \text{ and} \tag{62}$$

$$N = \{k \in \mathbb{N} : k > j \wedge Player(k) = A\} \ , \tag{63}$$

then we define $prev(j), next(j)$ as follows:

$$prev(j) = \begin{cases} \max P, & P \neq \emptyset \\ 0, & P = \emptyset \end{cases} \tag{64}$$

$$next(j) = \min N \tag{65}$$

$next(j)$ is always well defined with the assumption that eventually everybody plays.

8.2 Proofs, Lemmas and Theorems

Lemma 3 (*Loss* Equivalent to *Damage*).
Let $j \in \mathbb{N}, v \in \mathcal{V}_j \setminus \{A, E\}, v = Player(j)$. It holds that

$$\min(in_{v,j}, Loss_{v,j}) = \min(in_{v,j}, Damage_{v,j}) \ . \tag{66}$$

Proof. – 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 15), which cannot be decreased again but during an Angry player's turn (line 17).
4. $in_{v,j} \geq 0$

Thus

$$v \in Happy_{j-1} \Rightarrow \min(in_{v,j}, Damage_{v,j}) = \min(in_{v,j}, Loss_{v,j}) = 0 \ . \tag{67}$$

– Let $v \in Sad_{j-1}$. Then
1. $v \in Sad_j$ because $Turn_j = \emptyset$,
2. $in_{v,j} = 0$ (lines 26 - 27),
3. $Damage_{v,j} \geq 0 \wedge Loss_{v,j} \geq 0$.

Thus

$$v \in Sad_{j-1} \Rightarrow \min(in_{v,j}, Damage_{v,j}) = \min(in_{v,j}, Loss_{v,j}) = 0 \ . \tag{68}$$

– Let $v \in Angry_{j-1} \wedge v \in Happy_j$. Then the same argument as in the first case holds, if we ignore the argument (1).
– Let $v \in Angry_{j-1} \wedge v \in Sad_j$. Then the same argument as in the second case holds, if we ignore the argument (1).
Thus the theorem holds in every case. □

Proof 2 (Trust Convergence Theorem (1))**.**
First of all,

$$\forall j > j_0 : Player\,(j) = E \Rightarrow Turn_j = \emptyset \tag{69}$$

because E has already nullified his 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, thus player E can do nothing $\forall j > j_0$. We also see that

$$\forall j > j_0 : Player(j) = A \Rightarrow Turn_j = \emptyset \tag{70}$$

because of the idle strategy that A follows. As far as the rest of the players are concerned, consider the Transitive Game. As we can see from lines 4 and 15 - 17, it is

$$\forall j, \sum_{v \in \mathcal{V}_j} Loss_v = in_{E,j_0-1} \ . \tag{71}$$

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 27, 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

$$\exists j' : \forall j \geq j', |Angry_j \cup Happy_j| = c > 0 \wedge Angry_j \neq \emptyset \ . \tag{72}$$

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 27), 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

$$\forall j \in \mathbb{N}, DTr_j = \sum_{w,w' \in \mathcal{V}} DTr_{w \to w',j} \ . \tag{73}$$

Also, without loss of generality, suppose that

$$\forall j \geq j_1, out_{A,j} = out_{A,j_1} \ . \tag{74}$$

In $Turn_{j_1}$, v steals

$$St = \sum_{i=1}^{m} y_i \ . \tag{75}$$

We will show using induction that

$$\forall n \in \mathbb{N}, \exists j_n \in \mathbb{N} : DTr_{j_n} \leq DTr_{j_1 - 1} - nSt \ . \tag{76}$$

Base case: It holds that

$$DTr_{j_1} = DTr_{j_1 - 1} - St \ . \tag{77}$$

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

$$DTr_{j_2} \leq DTr_{j_1} - St = DTr_{j_1 - 1} - 2St \ , \tag{78}$$

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

$$\exists k > 1 : j_k > j_{k-1} > j_1 \Rightarrow DTr_{j_k} \leq DTr_{j_{k-1}} - St \ . \tag{79}$$

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

$$DTr_{j_{k+1}} \leq DTr_{j_k} - St \ . \tag{80}$$

We have proven by induction that

$$\forall n \in \mathbb{N}, \exists j_n \in \mathbb{N} : DTr_{j_n} \leq DTr_{j_1-1} - nSt \ . \tag{81}$$

However

$$DTr_{j_1-1} \geq 0 \wedge St > 0 \ , \tag{82}$$

thus

$$\exists n' \in \mathbb{N} : n'St > DTr_{j_1-1} \Rightarrow DTr_{j_{n'}} < 0 \ . \tag{83}$$

We have a contradiction because

$$\forall w, w' \in \mathcal{V}, \forall j \in \mathbb{N}, DTr_{w \to w', j} \geq 0 \ , \tag{84}$$

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

Proof 3 (MaxFlows Are Transitive Games Lemma (1))**.**
Without loss of generality, 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 the execution of the $MaxFlow\,(A, B)$ algorithm on $\mathcal{G}_0$. For any directed weighted graph G there exists a $MaxFlow$ over G that is a DAG, according to the Flow Decomposition Theorem [9]. We also know that we can apply the topological sort algorithm to any DAG and obtain a total ordering of its nodes with the following property: $\forall$ nodes v, w, it holds that $v < w \Rightarrow x_{wv} = 0$ [10]. Put differently, there is no flow from larger to smaller nodes. We execute the topological sort on X and obtain a total order of the nodes, such that B is the maximum and A is the minimum node. B 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 algorithm 4 will choose players following the total order, starting from player B. We observe that $\forall v \in \mathcal{V} \setminus \{A, B\}$, $\sum\limits_{w\in\mathcal{V}} x_{wv} = \sum\limits_{w\in\mathcal{V}} x_{vw} \leq maxFlow\,(A, B) \leq in_{B,0}$. Player B 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 $v = Player\,(j)$ will have stolen value x_{wv} from each in neighbour w.

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

$$Turn_1 = \bigcup_{v\in N^-(B)_0} \{Steal\,(x_{vB}, v)\} \tag{85}$$

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

$$\forall j \in [k], Turn_j = \bigcup_{w \in N^-(v)_{j-1}} \{Steal(x_{wv}, w)\} \tag{86}$$

Induction step: Let $j = k + 1, v = Player(j)$. 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 $\sum_{w \in N^+(v)_{j-1}} x_{vw}$. Since it is the first time v plays, $\forall w \in N^-(v)_{j-1}, DTr_{w \to v, j-1} = DTr_{w \to v, 0} \geq x_{wv}$, thus v is able to choose the following turn:

$$Turn_j = \bigcup_{w \in N^-(v)_{j-1}} \{Steal(x_{wv}, w)\} \tag{87}$$

Moreover, this turn satisfies the conservative strategy since

$$\sum_{w \in N^-(v)_{j-1}} x_{wv} = \sum_{w \in N^+(v)_{j-1}} x_{vw} \ . \tag{88}$$

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 B 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 $maxFlow(A, B)$. Since there remains no Angry player, it is obvious that j_2 is a turn that Transitive Game has converged thus $Loss_{A,j_2} = Loss_A$. It is also obvious that if B 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 4 (Transitive Games Are Flows Lemma (2))**.**
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:

$$\mathcal{V}' = \mathcal{V} \cup \{T\} \tag{89}$$

$$\mathcal{E}' = \mathcal{E} \cup \{(T, A)\} \cup \{(T, v) : v \in Sad_{j_1}\} \tag{90}$$

$$\forall (v, w) \in \mathcal{E}, c'_{vw} = DTr_{v \to w, 0} - DTr_{v \to w, j_1} \tag{91}$$

$$\forall v \in Sad_{j_1}, c'_{Tv} = c'_{TA} = \infty \tag{92}$$

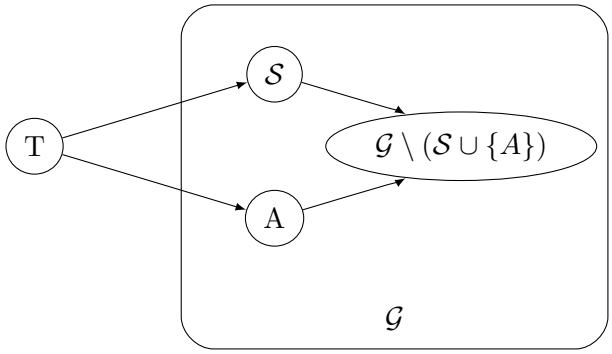

Fig.7: Graph $\mathcal{G}'$, derived from $\mathcal{G}$ with Auxiliary Source T.

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

$$\begin{aligned} &\sum_{w \in N^-(v)' \setminus \{T\}} c'_{wv} = \\ &= \sum_{w \in N^-(v)' \setminus \{T\}} (DTr_{w \to v,0} - DTr_{w \to v,j_1}) = \\ &= \sum_{w \in N^-(v)' \setminus \{T\}} DTr_{w \to v,0} - \sum_{w \in N^-(v)' \setminus \{T\}} DTr_{w \to v,j-1} = \\ &= in_{v,0} - in_{v,j_1} \end{aligned} \tag{93}$$

and

$$\begin{aligned} &\sum_{w \in N^+(v)' \setminus \{T\}} c'_{vw} = \\ &= \sum_{w \in N^+(v)' \setminus \{T\}} (DTr_{v \to w,0} - DTr_{v \to w,j_1}) = \\ &= \sum_{w \in N^+(v)' \setminus \{T\}} DTr_{v \to w,0} - \sum_{w \in N^+(v)' \setminus \{T\}} DTr_{v \to w,j-1} = \\ &= out_{v,0} - out_{v,j_1} \ . \end{aligned} \tag{94}$$

We can suppose that

$$\forall j \in \mathbb{N}, in_{A,j} = 0 \ , \tag{95}$$

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 $MaxFlow\,(T, B) = X'$ 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 (14) $\forall e \in \mathcal{E}$)
– Since $\forall v \in Sad_{j_1} \cup \{A\}, c'_{Tv} = \infty$, requirement (14) holds for any flow $x'_{Tv} \geq 0$.
– Let $v \in \mathcal{V}' \setminus (Sad_{j_1} \cup \{T, A, B\})$. According to the conservative strategy and since $v \notin Sad_{j_1}$, it holds that

$$out_{v,0} - out_{v,j_1} = in_{v,0} - in_{v,j_1} \ . \tag{96}$$

Combining this observation with (93) and (94), we have that

$$\sum_{w \in \mathcal{V}'} c'_{vw} = \sum_{w \in \mathcal{V}'} c'_{wv} \ . \tag{97}$$

(Flow Conservation requirement (15) $\forall v \in \mathcal{V}' \setminus (Sad_{j_1} \cup \{T, A, B\})$)
– Let $v \in Sad_{j_1}$. Since v is sad, we know that

$$out_{v,0} - out_{v,j_1} > in_{v,0} - in_{v,j_1} \ . \tag{98}$$

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

$$x'_{Tv} = (out_{v,0} - out_{v,j_1}) - (in_{v,0} - in_{v,j_1}) \ . \tag{99}$$

In this way, we have

$$\sum_{w \in \mathcal{V}'} x'_{vw} = out_{v,0} - out_{v,j_1} \text{ and} \tag{100}$$

$$\begin{aligned} \sum_{w \in \mathcal{V}'} x'_{wv} = \sum_{w \in \mathcal{V}' \setminus \{T\}} c'_{wv} + x'_{Tv} = in_{v,0} - in_{v,j_1} + \\ +(out_{v,0} - out_{v,j_1}) - (in_{v,0} - in_{v,j_1}) = out_{v,0} - out_{v,j_1} \ . \end{aligned} \tag{101}$$

thus

$$\sum_{w \in \mathcal{V}'} x'_{vw} = \sum_{w \in \mathcal{V}'} x'_{wv} \ . \tag{102}$$

(Requirement 15 $\forall v \in Sad_{j_1}$)
– We set

$$x'_{TA} = \sum_{v \in \mathcal{V}'} x'_{Av} \ , \tag{103}$$

thus from (95) we have

$$\sum_{v \in \mathcal{V}'} x'_{vA} = \sum_{v \in \mathcal{V}'} x'_{Av} \ . \tag{104}$$

(Requirement 15 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 $maxFlow(T, B)$ because all incoming flows to B are saturated. Also we observe that

$$\sum_{v \in \mathcal{V}'} x'_{Av} = \sum_{v \in \mathcal{V}'} c'_{Av} = out_{A,0} - out_{A,j_1} = Loss_A \ . \tag{105}$$

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

$$\mathcal{V}'' = \mathcal{V}' \tag{106}$$

$$E(\mathcal{G}'') = E(\mathcal{G}') \setminus \{(T, v) : v \in Sad_j\} \tag{107}$$

$$\forall e \in E(\mathcal{G}''), c''_e = c'_e \tag{108}$$

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

$$\sum_{v \in \mathcal{V}''} x''_{Tv} = x''_{TA} = \sum_{v \in \mathcal{V}''} x''_{Av} \ . \tag{109}$$

The outgoing flow from A in X'' will remain the same as in X' for two reasons: Firstly, using the Flow Decomposition Theorem [9] 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, thus

$$\sum_{v \in \mathcal{V}''} x''_{Av} \geq \sum_{v \in \mathcal{V}'} x'_{Av} \ . \tag{110}$$

Secondly, we have

$$\left.\begin{array}{l} \sum\limits_{v \in \mathcal{V}''} c''_{Av} = \sum\limits_{v \in \mathcal{V}'} c'_{Av} = \sum\limits_{v \in \mathcal{V}'} x'_{Av} \\ \sum\limits_{v \in \mathcal{V}''} c''_{Av} \geq \sum\limits_{v \in \mathcal{V}''} x''_{Av} \end{array}\right\} \Rightarrow \sum_{v \in \mathcal{V}''} x''_{Av} \leq \sum_{v \in \mathcal{V}'} x'_{Av} \ . \tag{111}$$

Thus we conclude that

$$\sum_{v \in \mathcal{V}''} x''_{Av} = \sum_{v \in \mathcal{V}'} x'_{Av} \ . \tag{112}$$

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

$$\sum_{v \in \mathcal{V}''} x''_{Av} = \sum_{v \in \mathcal{V}} x_{Av} \ . \tag{113}$$

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

$$\forall e \in \mathcal{E}, c_e \geq c''_e \ . \tag{114}$$

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

$$\sum_{v \in \mathcal{V}} x_{Av} = \sum_{v \in \mathcal{V}''} x''_{Av} \overset{(112)}{=} \sum_{v \in \mathcal{V}'} x'_{Av} \overset{(105)}{=} Loss_{A,j_1} \ , \tag{115}$$

which is the flow X. □

Theorem 6 (Conservative World Theorem).
If everybody follows the conservative strategy, nobody steals any amount from anybody.

Proof Sketch. If everybody is conservative, nobody can initiate the chain of steals. □

Proof.
Suppose that we are interested in graphs $\mathcal{G}_j$. Let (j_k) an increasing sequence of positive integers,

$$\begin{gathered} \text{let } S_{j_k} \subseteq N^- \left(Player\left(j_k\right) \right)_{j_k - 1} \text{ and} \\ \text{let } \forall v \in S_{j_k}, y_{v,j_k} > 0 \ . \end{gathered} \tag{116}$$

Suppose that there exists a subseries of History, $(Turn_{j_k})$, where

$$Turn_{j_k} = \bigcup_{v \in S_{j_k}} \{Steal(y_{v,j_k}, v)\} \ , \tag{117}$$

This subseries must have an initial element, $Turn_{j_1}$. However, $Player(j_1)$ follows the conservative strategy, thus somebody must have stolen from her as well, so $Player(j_1)$ cannot be the initial element. We have a contradiction, thus the theorem holds. □

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

$$\mathcal{V}_1 = \mathcal{V} \cup \{T_1\} \ , \tag{118}$$

$$\mathcal{E}_1 = \mathcal{E} \cup \{(v, T_1) : v \in \mathcal{B} \cup \mathcal{C}\} \ , \tag{119}$$

$$\forall v, w \in \mathcal{V}_1 \setminus \{T_1\}, DTr^1_{v\to w} = DTr_{v\to w} \ , \tag{120}$$

$$\forall v \in \mathcal{B} \cup \mathcal{C}, DTr^1_{v\to T_1} = \infty \ , \tag{121}$$

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 to facilitate comprehension. (Image)
According to Theorem (4),

$$Tr_{A\to\mathcal{B}\cup\mathcal{C}} = maxFlow_1(A, T_1) \wedge Tr_{A\to\mathcal{B}} = maxFlow_2(A, T_2) \ . \tag{122}$$

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 $X_1 = MaxFlow(A, T_1)$ can be used to construct a valid flow of equal value for the second graph if we set

$$\forall v \in \mathcal{V}_2 \setminus \mathcal{B}, \forall w \in \mathcal{V}_2, x_{vw,2} = x_{vw,1} \ , \tag{123}$$

$$\forall v \in \mathcal{B}, x_{vT_2,2} = \sum_{w\in N_1^+(v)} x_{vw,1} \ , \tag{124}$$

$$\forall v, w \in \mathcal{B}, x_{vw,2} = 0 \ . \tag{125}$$

Therefore

$$maxFlow_1(A, T_1) \leq maxFlow_2(A, T_2) \tag{126}$$

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

$$maxFlow_1(A, T_1) \geq maxFlow_2(A, T_2) \tag{127}$$

We conclude that

$$maxFlow(A, T_1) = maxFlow(A, T_2) \ , \tag{128}$$

thus from (122) and (128) the theorem holds. □

References

1. Sanchez W.: Lines of Credit (2016) https://gist.github.com/drwasho/2c40b91e169f55988618#part-3-web-of-credit
2. Nakamoto S.: Bitcoin: A Peer-to-Peer Electronic Cash System (2008)
3. Buterin V.: Bitcoin Multisig Wallet: The Future of Bitcoin. Bitcoin Magazine (2014),
4. Bitcoin Developer Guide, https://bitcoin.org/en/developer-guide

5. Cormen T. H., Leiserson C. E., Rivest R. L., Stein C.: Introduction to Algorithms (3rd ed.). MIT Press and McGraw-Hill (2009) [1990]
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.765âĂŞ774, ACM, New York (2013). doi:10.1145/2488608.2488705
7. Douceur J. R.: The Sybil Attack (PDF). International workshop on Peer-To-Peer Systems (2002)
8. 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)
9. Eulerian Walks, Flow Decomposition and Transformations https://ocw.mit.edu/courses/sloan-school-of-management/15-082j-network-optimization-fall-2010/lecture-notes/MIT15_082JF10_lec04.pdf
10. Kahn Arthur B.: Topological sorting of large networks. Communications of the ACM Vol. 5, Issue 11, pp. 558-562, ACM, New York (1962)
11. Zindros D. S.: Trust in decentralized anonymous marketplaces (2015) https://bitcoinmagazine.com/articles/multisig-future-bitcoin-1394686504