What is Trust

Orfeas Stefanos Thyfronitis Litos

University of Edinburgh
o.thyfronitis@ed.ac.uk

Abstract. We will try to define all the abstract properties that we would like "Trust" to have.

1 Introduction

Consider the UC setting, with an environment $\mathcal{E}$, an adversary $\mathcal{A}$ and a set of ITIs that follow a given protocol Π.

Definition 1 (Player). *A player is an ITM that follows Π. Let $\mathcal{P}$ be the set of all players.*

Intuitively, players spontaneously feel different desires of varying intensities and seek to satisfy them, either on their own, consuming part of their input tokens in the process, or by delegating the process to other players and paying them for their help with part of their input tokens. The choice depends on the perceived difference in price. Each player plays rationally, always attempting to maximize her utility.

More precisely, let $\mathcal{D}$ be a (finite) set containing all possible desires. At arbitrary moments during execution, $\mathcal{E}$ can provide input to any player $Alice \in \mathcal{P}$ in the form (idx, d, u), where $idx \in \mathbb{N}, d \in \mathcal{D}, u \in \mathbb{R}^+$. idx represents an index number that is unique for each input $\mathcal{E}$ generates, d represents the desire, whereas u the additional utility $Alice$ will obtain if d is satisfied. d is satisfied when $Alice$ learns the string $s\left(idx, d, Alice\right)$, either by directly calculating it or by receiving it as subroutine output from another player. Some of the players, given as input the tuple $(idx, d, Alice)$, can calculate $s\left(idx, Alice, d\right)$ more efficiently than $Alice$, which means that they need to consume less input tokens than $Alice$ for this calculation. $Alice$ can choose to delegate this calculation to a more efficient player Bob and provide the necessary input tokens for his computation with a surplus to compensate Bob for his effort. Both players are better off, because $Alice$ spent less tokens than she would if she had calculated $s\left(idx, d, Alice\right)$ herself, whilst Bob obtained some tokens which can in turn be used to satisfy some of his future desires.

Definition 2 (Cost of desire). *The cost of Alice's indexed desire, say* $(idx, d) \in (\mathbb{N}, \mathcal{D})$*, when satisfied by Bob is equal to the input tokens that Alice is required by Bob to give to him in order for him to calculate* $s\,(idx, Alice, d)$ *and is represented by* $c\,(idx, d, Alice, Bob)$*. The cost of satisfying this desire herself is represented by* $c\,(idx, d, Alice)$ *and is equal to the number of computational steps Alice must make in order to calculate* $s\,(idx, d, Alice)$*.*

2 Motivation for our Trust model

Nevertheless, one can say that at first sight it is in *Bob*'s best interest to trick *Alice* into believing that he can efficiently calculate $s\,(idx, Alice, d)$ and skip the computation entirely after obtaining *Alice*'s input, thus keeping all the tokens of the defrauded player. Evidently *Alice* would avoid further interaction with *Bob*, but without any way to signal other players of this unfortunate encounter, *Bob* can keep defrauding others until the pool of players is depleted; if the players are numerous or their number is increasing, *Bob* may keep this enterprise very profitable for an indefinite amount of time. This being a rational strategy, every player would eventually follow it, which through a "tragedy of the commons" effect invariably leads to a world where each player must satisfy all her desires by herself, entirely missing out on the prospect of division of labor.

One answer to that undesirable turn of events is a method through which *Alice*, prior to interacting with an aspiring helper *Bob*, consults the collective knowledge of her neighborhood of the network regarding him. There are several methods to achieve this, such as star-based global ratings. This method however has several drawbacks:

- Very good ratings cost nothing, thus convey little valuable information.
- Different players may have different preferences, global ratings fail to capture this. Arrow's impossibility theorem is possibly relevant here.
- Susceptible to Sybil attacks; mitigation techniques are ad-hoc and require (partial) centralization and secrecy/obfuscation of methods to succeed, thus undermining the decentralized, transparent nature of the system, a property that we actively seek.

3 Definitions

We thus define two kinds of trust: direct and indirect. Direct trust from *Alice* to *Bob* is represented by input tokens (initially belonging to *Alice*) actively given by her to an escrow from which *Bob* can also take them. As long as *Bob* does not take these tokens, *Alice* directly trusts him equally to the amount of tokens held by the escrow.

This information can be used by another player *Charlie* that directly trusts *Alice* in order to derive information regarding *Bob*'s trustworthiness, even if *Charlie* does not directly trust *Bob*. This is called indirect trust.

Definition 3 (Direct Trust). *The direct trust from Alice to Bob, represented by $DTr_{Alice \to Bob}$, is equal to the available tokens count sent by $\mathcal{F}_{Trust}$ as a response to a message* `query_direct_trust(`*Alice*`, `*Bob*`)`*.*

Definition 4 (Indirect Trust). *The indirect trust from Alice to Bob, $Tr_{Alice \to Bob}$, is measured in tokens like direct trust and can be calculated deterministically given the existing direct trusts between all players. It is equal to the number sent by $\mathcal{F}_{Trust}$ as a response to a message* `query_indirect_trust(`*Alice*`, `*Bob*`)`*.*

We would like to provide players with an ideal functionality where they:

1. Directly trust coins to another player
2. Steal coins previously entrusted by another player
3. Retract coins previously entrusted to another player
4. Query trust towards another player

The following functionality provides such an interface:

$\mathcal{F}_{Trust}$

```
Initialization:
for all $Alice, Bob \in \mathcal{P}$
  $DTr_{Alice \to Bob}$ = 0
  $Tr_{Alice \to Bob}$ = 0

Upon receiving input entrust($Bob$, $1^{\text{tokens}}$) from $Alice$:
  $DTr_{Alice \to Bob}$ += tokens

Upon receiving message steal($Bob$, tokens) from $Alice$:
  If $DTr_{Bob \to Alice}$ >= tokens
    $DTr_{Bob \to Alice}$ -= tokens
    input $1^{\text{tokens}}$ to $Alice$
  Else discard request
```

```

Upon receiving message distrust($Bob$, tokens) from $Alice$:
  If $DTr_{Alice \to Bob}$ >= tokens
    $DTr_{Alice \to Bob}$ -= tokens
    input $1^{\texttt{tokens}}$ to $Alice$
  Else discard request

Upon receiving message query_direct_trust($Alice$, $Bob$) from
    $Charlie$:
  If $Charlie \in \{Alice, Bob\}$ # Privacy
    send message $DTr_{Alice \to Bob}$ to $Charlie$
  Else discard request

Upon receiving message query_indirect_trust($Alice$, $Bob$)
    from $Charlie$:
  If $Charlie$ == $Alice$ # Privacy
    $Tr_{Alice \to Bob}$ = calculate_indirect_trust($Alice$, $Bob$, all
        direct trusts)
    send message $Tr_{Alice \to Bob}$ to $Charlie$
  Else discard request
```

4 Desired Properties for Indirect Trust

1. $Tr_{Alice \to Bob} \geq DTr_{Alice \to Bob}$
2. If universe (1) and (2) are identical except for $DTr_{Alice \to Bob}$, then

$$Tr^2_{Alice \to Bob} = Tr^1_{Alice \to Bob} - DTr^1_{Alice \to Bob} + DTr^2_{Alice \to Bob} \ .$$

3. Consider an indexed desire (idx, d) $Alice$ has. If

$$\begin{aligned} &c\left(idx, d, Alice, Bob\right) < c\left(idx, d, Alice\right) \text{ and} \\ &c\left(idx, d, Alice, Bob\right) \leq Tr_{Alice \to Bob} \ , \end{aligned}$$

   then it should be rational for $Alice$ to prefer to delegate the calculation of $s\left(idx, d, Alice\right)$ to Bob than to calculate it herself.

References