AnonBlock

Alex Tong

November 2017

1 Goals

The goal of this project is to build a blockchain based, tracking resistant, global reputation system utilizing previous work on untraceable cryptocurrencies. Ideally our system should provide a permissionless (anyone can create a new pseudonym at will) system with a generalizable reputation function (allowing an arbitrary feedback function) which is either a global or personal, where we are unable to associate posts or feedback messages to a specific user or between each other.

2 Introduction

In the past few years there has been an explosion of development in cryptocurrencies with a variety of applications. While Bitcoin, the oldest and largest by market cap, does not inherently supply any form of privacy, there are a number of other efforts to provide a secure untraceable form of electronic cash. Tokens such as Monero, ZCash, and DASH, are examples of these efforts. Monero is currently used to perform a large amount (How much?) of illegal transactions every day. To hide their money the agents behind the Wannacry ransomware attack transferred their Bitcoin tokens into Monero tokens. It comes to this, if the Monero protocol can be trusted to hide large financial value, then it should serve as a useful primitive for building an anonymous reputation system.

3 Assumptions

We assume the availability of an anonymous transaction service that provides functions of an anonymous cryptocurrency, specifically (1) wallets with amounts hidden (optionally) from all but the private key holder, (2) publicly verifiable transactions which optionally hide the sender address, receiver address, and amount, (3) an eventually consistent system preventing double spending of tokens.

We will also assume a global reputation system, where the reputation value calculated for a given agent is eventually consistent between all agents, and

whose feedback update function depends on the feedback giver's reputation to provide some sybil resistant properties. Let the feedback sender's reputation be R_s, the sender's feedback score be S, and the feedback reciever's reputation be R_r, then the global feedback update function should be of the form

$$Feedback(R_s, S, R_r) \rightarrow R'_r$$

This implies independence between feedback messages directed at the same receiver. We can fix this by generalizing the receivers reputation from a scalar form to an object containing the reputation score as well as useful metadata for the specific reputation algorithm, this would allow us to mitigate a slandering attack by a sybil adversary by tracking the number of feedback messages from low reputation agents

There are a number of assumptions that apply to more specific types of reputation systems.

4 Background

Monero uses BLANK to secure transactions, ZCash uses BLANK.

5 Our Approach

Since we assume a secure, private, and anonymous transaction system, our goal is to build a protocol leveraging such a system to create an anonymous reputation system. The high level plan is as follows. There are two types of wallets, agent wallets and message wallets. An agent holds any number of messages by retaining the message wallet's private key. To send a message an agent transfers an amount from his wallet to the message's wallet. To give feedback, an agent sends a transfer indicating his reputation and score for a specific message.

There are four possible transaction sender, receiver tuples:

1. Agent → Agent
2. Message → Agent
3. Agent → Message
4. Message→ Message

Types 1,2 are performed as normal, type 3 is performed as specified below, and type 4 is disallowed.

We implement the following additional functions for agent to message transactions.

- Post(reputation, data)
- GiveFeedback(score, reputation, messageId)

A post by an agent posts a publicly available message which is cannot be associated with the agent. This is implemented as a transaction with private sender, public receiver and amount. We implement this as a transaction to utilize the double spending protection inherent in they underlying system.

A $GiveFeedback$ request represents a feedback post to a message by an agent. In this request we hide the sender's identity, but make their score, reputation, and messageId publicly available. This is implemented as a transaction from the feedback giver to the message wallet with private sender, public receiver and amount. Implementing this as a transaction allows us to utilize the double spending protection of our underlying system to prevent an agent from posting feedback with more reputation than they own. In each block round an agent could potentially post many feedback messages, however the underlying double spending resistance means that the reputation of all $GiveFeedback$ messages in a given round for a given agent sum to less than the agents total reputation.

Since an agent owns the private keys to all of the messages it owns, then at any time an agent can withdraw the reputation stored in one of its messages to itself. This requires a transaction with a private receiver.

In a political blog example where the posters and feedback givers want to hide their identities, the following mapping could apply, the *data* of each post could be the blog post text, the *score* of a feedback request could be a thumbs up or thumbs down, and the $messageId$ could be the hash of the post text.

6 Model Assumptions

A major drawback to this approach is that when posting a message, an agent attaches an amount of their reputation which they then cannot use in another message. This property varies with the model assumptions specifically in the (1) Feedback window F, and (2) whether negative feedback is allowed. A bounded feedback window makes sense in a reputation system like a file delivery system (Gnutella) where a we can assume a client can fairly quickly give feedback as to the validity of the file. Such a bound makes much less sense in a public posting forum such as the blog example above where we would like to receive feedback for an indefinite period of time.

Allowing only non-negative feedback makes things easier, as the protocol does not have to actually complete the transfer from agent to message, the agent can retain all its reputation tokens and the system can verify that the poster had at least as many reputation tokens as it was posted with. If we allow negative feedback then the message must hold onto the reputation tokens we posted with this means that the agent cannot use these posted tokens until the feedback period for the message has ended.

The only case which presents a problem is then a model which allows negative feedback and has a long feedback window. Under these model assumptions an agent must distributed their reputation among all of their messages.

An identity with zero reputation is untrustable in this system as it cannot

be given negative feedback in any meaningful way. Negative reputation is not representable in general in a permissionless reputation system as an adversary can easily create a new identity with zero reputation at any time. Therefore when an agent posts a message with a given amount of reputation not greater than the agents total reputation.

7 Notes

Agent to agent transactions allows direct transfer of reputation (not sure if this a feature)

To calculate the reputation of a message we need the full chain (requiring a ton of space). It would be possible to personalize the reputation system if we wanted because all information necessary to calculate repuation is publically available.

Seems like Proof of stake algorithms for distributing reputation and maintaining the chain might be very reasonable in this system.

In the far distant future it would be really cool to generalize reputation between companies or systems. Since we can provide proof of a reputation on a certain system with our private key, then we can use this to gain an initial reputation on a new/different system.

It would be really nice to solve the long/negative feedback case, but it seems more likely we could prove something in the negative for it.

In Monero I'm pretty sure it is hard to make a viewable wallet. This means that there might be further limitations on the protocol if we used Monero as a base protocol. `https://www.reddit.com/r/Monero/comments/4ce5ui/what_is_the_use_of_view_only_wallet_when_its/`

References