Identity and reputation systems on ethereum

Abstract

In the context of ethereum, a *reputation* is that something by which a smart contract or a DAO recognizes a user. An *identity* here is defined as a collection of reputations, who are associated with each other just by virtue of being earned by that identity or (we assume) user. We explore the logistics of building identity and reputation systems on ethereum, with particular emphasis placed on avoiding and mitigating Sybil attacks with minimal impact on the dynamics of reputation systems.

Introduction

Reputation systems are useful for any smart contract, DO, or DAO who needs to keep track of their users on an individual basis, likely so that they can reward them for their contributions. Every system that cannot use cryptographic proof of contribution must use social proof. Social proof inherently requires a secure identity and reputation system in order to function without being exploited. To be clear, if no reputation is required, an attacker can introduce many identities (perhaps at some expense to herself) in order to get whatever benefit is made available to contributors without having actually made a contribution. As is convention, we will refer to this kind of an attack as a *Sybil attack*, and we will refer to the attacker as *Sybil*, and her identities as her *faces* or *personalities* (after the book movie inspired by a woman with multiple personality disorder). Moreover, we will refer to a user who is not Sybil as *Cyril*. We will begin by developing a scheme for the security of identity and reputation as identifiers, and then will turn our attention to the security of reputation systems.

Identity and reputation

We realize that inasfar as reputation is valuable and it is possible, reputation will be bought and sold. However, the intuition about reputations is that they should not be transferable, for after they have been transferred they will not be as reliable an indicator of their owner's integrity or ability. We could attach reputation to public keys in order to make their trade more difficult, but then a compromise of the corresponding private key would compromise the user's reputation. If we make reputation a transferrable token, this problem would be somewhat mitigated in the case where the private key *might* have been compromised, but it also facilitates the transaction and theft of reputation. We can do better by instead making *identity* a transferrable token, and having reputations be permanently attached to particular identities. The problem of reputation sale is mitigated with this approach, if there are more than one reputations attached to identities.

However, the issue of the compromise of a signing key leading to identity theft and loss of control is still a concern. We therefore will separate the signing key for the activity associated with an identity's reputations (the *usage key*), and the keys required to revoke and replace the usage key (the *security keys*), in case of a possible compromise. Here we will use an N of M multisig scheme of the user's choosing, so that users can choose their prefered tradeoff between identity security and convenience of usage key revocation and replacement. We go to these lengths because we predict that without them, identity theft may become as big of an issue as bitcoin theft. Note also that this scheme makes the sale of reputation and identity even more difficult, although it is still possible. We tentatively regard this as a satisfactory state of affairs, so we will shift our focus from identity and reputation as identifiers to the specification of reputation systems.

Asocial reputation systems

While we will mainly focus on reputation systems whose purpose is to produce social proofs, it is worth noting that asocial reputation systems are possible, for example when cryptographic proof can be used to measure contribution. These systems, too, can be subject to Sybil attacks under some circumstances, as we will soon see. This analysis will be trivial, but it provides useful intuitions that will carry to the analysis of social reputation systems. As we have mentioned before, so long as reputation is valuable, it will be bought and sold. Therefore, we will use *reputation price* (or *rep price*, for short) to refer to the price at which the particular reputation in question would be traded, perhaps inferred by the benefits reserved for the identity which holds that reputation. Additionally we introduce the notion of *reputation cost* (or *rep cost*), which is the cost to a particular user of obtaining a particular reputation. Note that rep cost may be user-dependent, while rep price is not. Finally, we might consider that in general any reputation system might require a *membership fee* of users before granting them a reputation, and the corresponding access that that entails.

In the figures that follow, we plot possible functional relationships between rep price and rep cost. The red curve, common to all plots, reflects the cost of membership incurred by a user upon entering the reputation system. The slope of the green curve, also common to all plots, reflects the baseline rep price to rep cost ratio. Finally, the blue curve reflects the relationship between rep price and rep cost, as implemented in the particular reputation system. The difference in slopes and the intersection point between the green and blue curves are of particular importance in the analysis.

If for any given user the relationship between rep price and rep cost is concave down (as in **Figure 1**), then there is a rep cost which maximizes the rep price to rep cost ratio. This incentivizes users to spread a multiple of the optimal cost evenly across multiple accounts, thereby making the reputation system vulnerable to Sybil attack. If, on

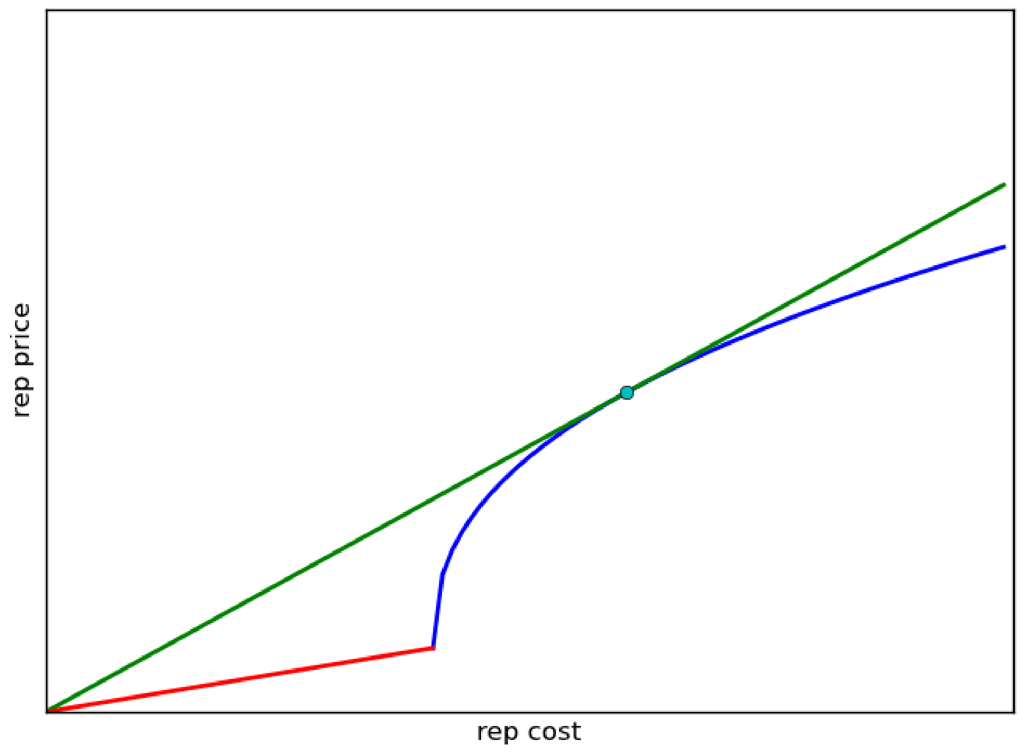
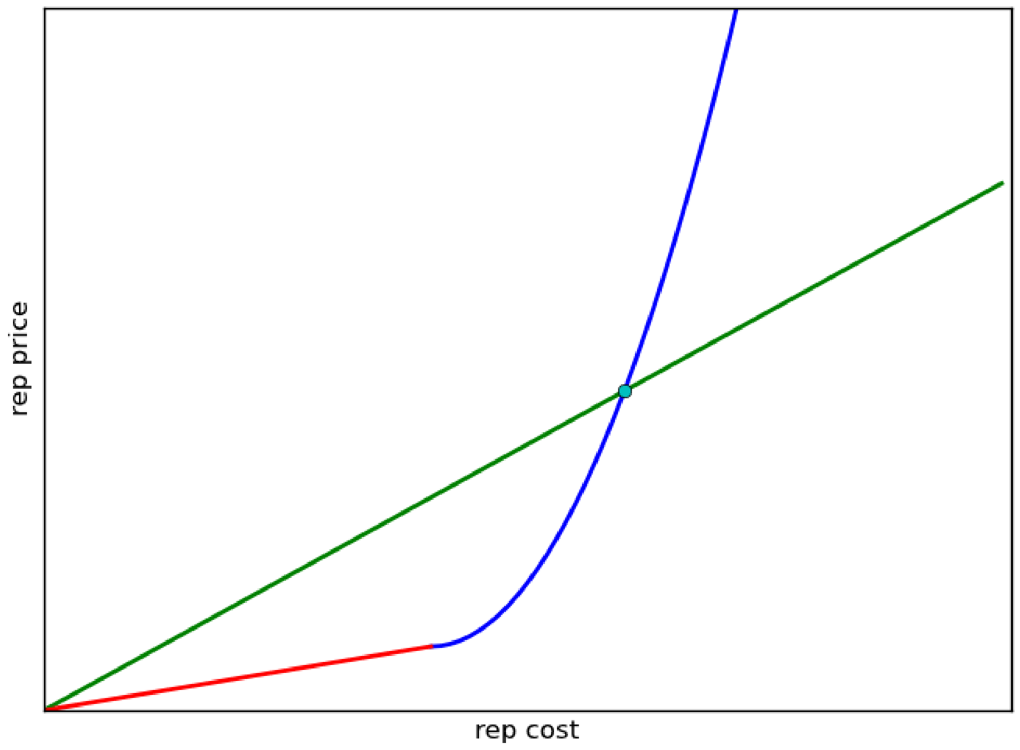


Figure 1: When the price of reputation is concave in the reputation's cost, Sybil has an optimal rep cost investment for each of her personalities

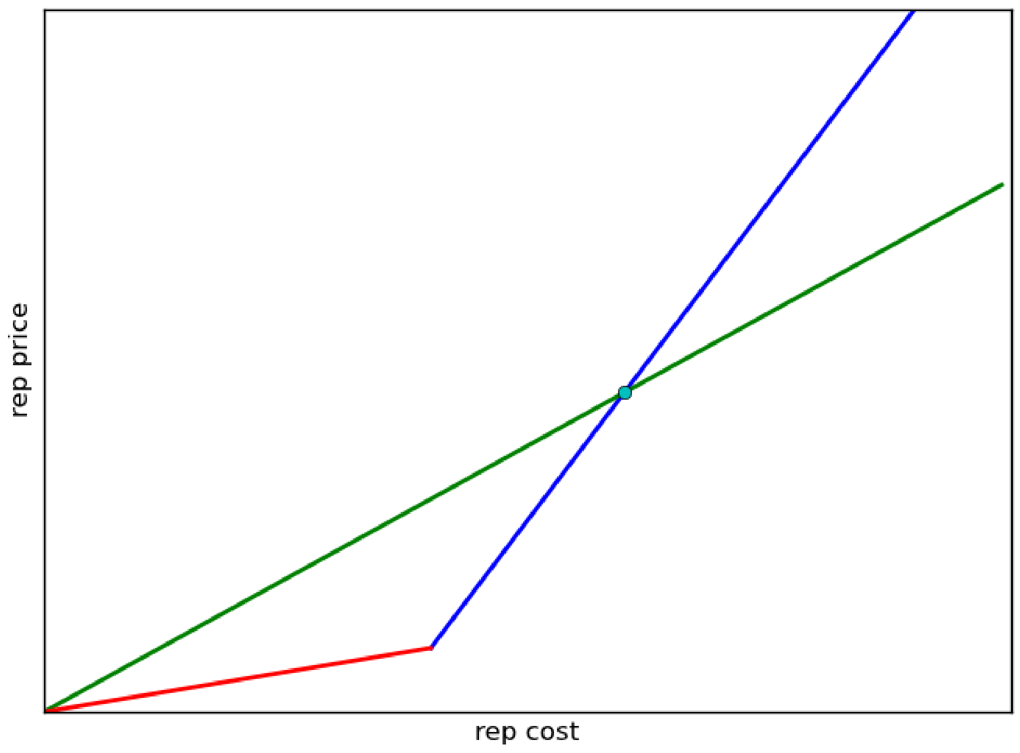
the other hand, the relationship between rep price and rep cost is concave up (as in **Figure 2**), then users have an incentive to have only one reputation, and Sybil attacks are prevented. The drawback in this case is that users who are able to invest more rep cost are going to devalue or far overshadow the rep price of users who are not able to invest as much cost, since the ratio of rep price to rep cost is unbounded. However, there is a third possibility (**Figure 3)**: let the relationship be (at least asymptotically) linear. This is appealing because it simultaneously bounds the rep price to rep cost ration and prevents Sybil attacks by incentivizing users to have only one reputation.

Figure 2: When the price of reputation is convex in the rep cost, Sybil has no incentive to introduce new nodes, but the ratio of rep price to rep cost is unbounded



We observe that Bitcoin is an asocial reputation of the concave up variety, with hashing power as reputation. Here the rep price can be regarded as a pie that is split between the users in proportion to their reputation, because rep price is determined by the share of newly minted bitcoin that it represents. The effect is that users who are able to invest more into mining get much higher rep price, and they thereby dilute the rep price of other users. If Bitcoin’s proof-of-work function was restricted to CPUs, the relationship between rep price and rep cost would be much closer to linear.

Figure 3: When the price of reputation is linear in the rep cost, the rep price to rep cost ratio is bounded and Sybil has no incentive to introduce new nodes



Cyril tokens

Ideally, mitigating Sybil attacks on social reputation systems discourages as much as possible Sybil's malicious use of the system without discouraging contributors. The first class of approaches that we will discuss aims to make it more expensive for Sybil to use the reputation system. There are two "good" ways we know of accomplishing this. One simply makes the user pay a membership fee in order to make being Sybil more expensive, either through proof-of-burn or a payment directly to the network. The other is to design *challenges* or *tasks* that are resistant to Sybil attacks because they are relatively easy for humans to do a small number of times, but are hard to automate. The first set of solutions is straight-forward and is guaranteed to be effective - to a point - but the obvious drawback is that all users have to pay. Solutions of the second type, on the other hand, are challenging to implement sustainably particularly because of advances in machine learning. If they can be done in a convenient way, however, it will save users coin and help keep Sybil off of the network. We will focus on solutions of the second type, by creating a system of tokens, called *Cyril tokens*, that are issued upon completion of such tasks.

A task-based Cyril token is said to be *broken* if it is easy for Sybil to earn these tokens. A broken Cyril token is therefore cheap, and this will be reflected in its price if it is traded on a market. On the other hand, a good Cyril token is expensive, providing evidence that Sybil has not yet succeeded in automating its production. There is a strong incentive for people to automate the creation of these tokens inasfar as they are expensive or required for access to reputation systems. If they succeed in automating the completion of these challenges, the price of the Cyril tokens will fall, revealing that the challenge has been broken.

If a reputation system requires Cyril tokens for use, instead of destroying them, it may sell them onto the market, thereby interfering with the price that is meant to signal whether or not the Cyril token is broken. Therefore, Cyril tokens should be aware of whether they are being transferred onto/from the exchange or not - if they are transferred outside of the exchange, they should automatically become void. While enforcing such a rule has its difficulties, it has the additional benefit of making the price signal stronger, by nearly eliminating the possibility of a dark market for Cyril token.

Social reputation systems

The main challenge when building social reputation systems, as we have alluded to, is avoiding the effective exploitation and subsequent co-opting of the system by Sybil and her many personalities. We have already mentioned that a reputation system may require the expenditure of Cryil tokens for its use, however this isn't sufficient mitigation against Sybil attacks for a reputation system whose design is inherently vulnerable. For example, a naive implementation might let any user increase any other user's reputation in proportion to their own reputation. While requiring Cyril tokens increases the cost to Sybil of using the system, this cost increases linearly in the number of nodes, while Sybil may increase her reputation much faster, in the number of reputable nodes introduced. In this case, we have an instance of **Figure 2**, where an increase in cost to Sybil leads to an accelerating increase in her rep price. We therefore must find methods by which social reputations can ensure that Sybil's rep price increases linearly in her rep cost. Preferably, this mitigation will not cripple the intended dynamics of the reputation system.

We have discovered one such solution, which happens to be both compelling and elegant. It requires that the reputation system give users a *reputation allowance*, an exhaustible budget for giving other users reputation. Reputation allowance must be earned, and it must cost Sybil more than the increase in rep price that she might receive from increasing her own reputation. Specifically, a reputation system should require the expenditure of Cyril tokens during the activity that earns rep allowance in an effort to make it possible for the reputation system to be profitable for Cyril, but unprofitable for Sybil. How precisely a social reputation system accomplishes this coordination will have to be implementation-specific, and therefore will not receive treatment here.

All other solutions that we've come up with have either been computationally difficult or have failed to make the increase in rep price for Sybil, gained by adding reputable nodes, linear in the rep cost invested. We will list some of these "solutions" here, for reference. Keeping track of pairs of users to insure that they do not increase their rep too much is an expensive mitigation, and it only reduces the benefit to Sybil to be quadratic in the number of nodes, in the best case. Having ethereum create instances where only chosen users can increase each other's reputation increases the cost or waiting time to Sybil, but does not change the fundamental dynamics of the reputation system; it merely makes the same instances of exploitation occur probabilistically. Using machine learning to identify abusive behaviours is expensive to the point where it is unfeasible to build into the ethereum contracts of the reputation systems, although it provides "fuzzy" promise. Finally, it is always possible to employ a web-of-trust model, in this case requiring a centralized initialization process to bootstrap the network. While a web-of-trust may be promising for smaller communities, it is cumbersome and inhibitive on a larger scale.

Discussion

Membership fees offer a simple mitigation of Sybil attacks. Additionally, to further prevent Sybil attacks and centralization of rep price, it is important for reputation systems to have an asymptotically-linear relationship between rep price and rep cost. This is true of both asocial and social reputation systems. In social reputation systems, requiring Cyril tokens for network activity that might earn a user reputation allowance can successfully mitigate Sybil attacks, if the amount of Cyril tokens requires is calibrated correctly. However, this calibration will become more difficult over time as it becomes harder for any given user to prove that they are not automated. Therefore, the sooner we establish a working identity and reputation system, the more likely we are to find success without excessive difficulty. Once we can cheaply automate the completion of all Turing tests, however, secure reputation systems will become unprofitable for Cyril.

Conclusion

We have developed a scheme that makes it possible to create working identity and reputation systems, which cannot be exploited by automated attackers, so long as working tests for automation can be designed. This does not then mean that they cannot be attacked by dedicated human attackers. The only mitigation against these more sophisticated attacks is to require that a reputation system's members have so many reputations attached to their identities that they could not possibly have many identities in the system. This will work, but is only possible in a mature identity and reputation system. Alternatively, users of a reputation system could be required to constantly broadcast everything they are doing, so that they can convince the network that they are not taking on multiple identities, but this is not an attractive solution inasfar as we can solve the same problem by more efficient means.