Enabling Interactions Between Encointer Local Communities: Sketch of a Global Proof-of-Personhood Protocol

E. Delacour
Supervised by A. Brenzikofer, Y. Vonlanthen, R. Wattenhofer

2023

Abstract

Electronic voting, online social networks, and resource distribution all suffer from a common issue: individuals creating multiple accounts, commonly referred to as Sybil attacks. Within the realm of blockchain, various strategies have been developed to address this challenging problem, and can be coined under the term of *proof-of-personhood*. Among these approaches, the Encointer protocol distinguishes itself by offering a decentralized, privacy-centric, and secure solution that is based on physical meetings within local communities, each equipped with its own currency and economy. However, the Encointer protocol focuses on a local approach to proof-of-personhood, working within a bounded geographic area. In this work, we introduce an extension of the Encointer protocol for a global proof-of-personhood framework founded upon local communities' proof-of-personhood. This global protocol relies on a trust-graph connecting these communities. We also propose a design for a decentralized currency exchange, tailored to the specificity of Encointer economies and that serves as a foundation to derive the trust-graph, leveraging economic incentives.

Contents

1 Introduction

The question of identity and pseudonyms on the Internet has been a topic of concern since the advent of the digital era. If the possibility of expressing oneself under a pseudonym is sometimes seen as an opportunity for freedom of speech [14], it is also this ease of creating pseudonyms that became an obstacle to online voting: how to ensure that each person has one voice, and one voice only, when it's so easy to obtain a multitude of accounts. The illegitimate creation of multiple accounts is known as a Sybil attack and is associated with undesirable practices such as ballot stuffing, astroturfing, or sock puppetry.

If preventing Sybil attacks can be considered through identifying users, it is actually not required to do so, but just to make sure they do not own several accounts. Various solutions to these problems have been proposed, each with their specific strengths and weaknesses, and are generally coined under the term of proof-of-personhood. Among these solutions, the Encointer protocol [3] offers a proof-of-personhood protocol with strong guarantees for local communities. Each Encointer community organizes regular physical and local meetings, and delivers its certifications to participants.

The goal of this work is to develop mechanisms to allow interactions between local communities. The primary focus is on designing a decentralized exchange (DEX) between communities, which all have their own currency. The particular nature of Encointer economies requires some adaption from standard type of decentralized exchanges, such as Constant Product Market Makers (CPMM). The second is to derive a global proof-of-personhood protocol based on the local protocols. Our approach is based on establishing a web-of-trust among communities, leveraging economic incentives that rely on our DEX.

Our main contributions are a classification of proof-of-personhood approaches and their applications, an adaptation of CPMM to demurrage, with a demurrage reduction mechanism aimed at incentivizing liquidity providing, a mechanism to derive a web-of-trust between Encointer communities leveraging economic incentives, and a classification of communities between trusted and not, based on a PageRank [5] analysis of the web-of-trust and an initial trusted setup.

In the next section we provide a definition of proof-of-personhood, detail some promising applications, and provide an overview of some existing approaches.

2 Proof-of-Personhood: the Solution to Sybil Attack

2.1 Context

Blockchains are a type of distributed ledger technology (DLT). They basically are peer-to-peer decentralized (i.e., without central authority) and immutable records, where it is possible to store data in an append-only mode. Since the creation of Ethereum [6], some blockchains also allow the decentralized execution of code. Such programs are called smart contracts. A data stored on a blockchain is said to be *on-chain*.

We consider a set of persons who can own accounts. Accounts are typically a username/password couple, or more frequently in the context of blockchains, a public key/private key couple. A person owns an account if they know the associated password or private key. An account is publicly identified by its address, which is generally its public key or its username.

2.2 Proof-of-Personhood: Definitions

Definition 1. *A proof-of-personhood protocol is a series of rules that defines a set of accounts that are said to be certified. This set must be such that any certified account satisfies two properties: 1) it is owned by a person, and 2) the owner does not own more than one certified account of the protocol* ***at a time****.*

We call a certification the fact that an account is certified.

Definition 2. *A protocol that emits certifications that do not satisfy the two properties of Definition 1 is said to be a Sybil protocol. Such certifications are said to be Sybil certifications. A Sybil protocol is not a proof-of-personhood protocol.*

Given the two properties defined in Definition 1, that constitute the essence of proof-of-personhood, several choices are still to be made in the design.

Definition 3. *A certification is said to be* ***perishable*** *if it has a fixed expiration date, after which the certification is no longer valid. A protocol is said to be* ***perishable*** *(resp.* ***non-perishable****) if all its certifications are* ***perishable*** *(resp.* ***non-perishable****). Note that a* ***non-perishable*** *certification is not necessarily definitive, as it can be revoked for other reasons than an expiration date.*

Definition 4. *A protocol is said to be* ***t-renewable*** *if a user can have a new account certified every t units of time. Having a new account certified always requires the former certification to be no longer valid (expired or revoked). Within the t time frame, a user cannot change its certified account. A system where a user can never change its certified account is said to be ∞-renewable. We said that a protocol is* ***renewable*** *if it is t-renewable with $t < \infty$.*

As we will see, whether a protocol is renewable or not has implications on the possible applications. Some specific applications can require the protocol to be ∞-renewable. However, when this is not required, a t-renewable protocol with $t < \infty$ can provide higher privacy guarantees, as changing account prevents from linking past activities to present ones.

Definition 5. *Two protocols are said to be* ***disjoint*** *if it is impossible for a person to have accounts certified in both protocols at the same time.*

If two protocols are disjoint, then considering both protocols certifications as valid constitutes a proof-of-personhood protocol, i.e., only a human can have a certified account, and it is not possible to have two certified accounts at the same time.

Definition 6. *Given a sequence of* ***disjoint*** *proof-of-personhood protocols $\{P_i\}_i$, we define the union protocol of $\{P_i\}_i$ by the protocol in which the set of certifications is equal to the union of the sets of certifications of $\{P_i\}_i$. It is a proof-of-personhood protocol.*

As we will see, the range of applications is wide. In the next subsections, we provide a non-exhaustive classification of proof-of-personhood applications.

2.3 Applications

2.3.1 Voting

Voting generally implies three requirements:

1. a person can only vote once
2. only allowed persons can vote (for example, only resident of a region can vote for an election, or only members of the board for a decision within a company etc.)
3. **coercion resistance:** it is hard to buy votes, i.e., it is hard for a voter to prove to a vote buyer that they actually voted for the demanded outcome.

When votes take place in person, requirement 3 was traditionally achieved with voting booths. It is harder to satisfy in the case of electronic voting, but protocols such as [17] focus on offering layers to achieve coercion resistance.

Proof-of-personhood is naturally focused on satisfying requirement 1. In the case of Encointer, as we will see, a proof-of-personhood certification is associated with the physical presence in a specific area. Therefore, in the case of a local community vote, with the assumption that residents are the one allowed to vote, the Encointer proof-of-personhood protocol can also be used to satisfy requirement 2.

Democracy: Proof-of-personhood allows achieving 1-person-1-vote, and by extension, it becomes possible to run referendums, or elections, in a decentralized way. Generally, we do not need a ∞-renewable protocol for these applications. Renewable protocols are often even better, as they guarantee a higher level of privacy.

Rating on social networks: Social networks are often not resilient to Sybil attacks. When it comes to likes, followers, upvotes or downvotes, which are forms of votes, the creation of multiple accounts can allow giving credit to an idea, make it look more consensual. Proof-of-personhood could typically be a solution to this problem. The protocol can either be ∞-renewable or renewable as long as votes are taken into account only if their author is an address currently certified (in particular, it is not revoked or expired).

Consensus: Bitcoin consensus uses proof-of-work. In a nutshell, participants' voting power is proportional to their computing power. This gives a considerable edge to participants having money, living in a country where electricity is cheaper, or where cooling is easier. With proof-of-stake, participants' voting power directly depends on the amount of money they have locked, and therefore on their wealth. Proof-of-personhood could be used alternatively, to run a more democratic and fairer consensus protocol. It would also be more energy-efficient than proof-of-work.

Dispute Resolution: In dispute resolution protocols, such as Kleros [8], jurors are randomly selected to arbitrate a dispute. To prevent Sybil attacks (a big number of jurors is created in order to increase the chance of an attacker to decide on the outcome of a dispute), jurors are selected with a probability proportional to the amount of Kleros tokens staked. A proof-of-personhood-based selection system has the potential to enable a justice equally run by the people, regardless of individuals' wealth.

It is worth noting that the Kleros dispute resolution protocol is used to

arbitrate the Proof-of-Humanity protocol, which will be discussed in the next subsection.

Decentralized oracle: Bringing real world data on-chain in a decentralized way is a hard problem known as decentralized oracles. Proof-of-personhood could be a pillar for democratic decentralized oracles, where participants would vote for the outcome they consider to be true.

2.3.2 Resource Distribution:

Fair resource distribution may be achieved through proof-of-personhood. Universal basic income is a kind of resource distribution, and is already implemented by several proof-of-personhood protocols [3] [24].

Another example of resources distribution is fair air drop, which would work as 1 person 1 drop. In most cases, a renewable protocol works well. In the case of a resource that should be given only once in a lifetime, a ∞-renewable protocol could fit the application better. For example, a ∞-renewable protocol could be used to provide freemiums and trial versions. A free trial period could be distributed to each person, without fearing the Sybil attack that would consist in creating multiple accounts to benefit from trial periods again and again. A t-renewable protocol could also be used, with a limited use of the software per person and per t units of time.

2.3.3 Anti-Spam and Anti-Bot:

Proof-of-personhood can be used to limit the use of bots. Indeed, an online service can decide to answer only to certified addresses. Persons are still able to use bots, but only one per person, and a per-person rate limit can be enforced. It is also useful to prevent from spam, or DOS-like attacks, as soon as it is easier for a server to check if an address is certified, than actually answering the request.

2.3.4 Identity Verification and Self Sovereign Identity:

Proof-of-personhood and self sovereign identity are two solutions aimed at solving two different problems. Still, it is worth noting that one could help to build the other.

On the one hand, if identities are unique, providing them through a self-sovereign-identity protocol would be a form of proof-of-personhood. As we

have already mentioned, however, it could be desirable to not be forced to provide one's identity in order to prove one's personhood.

On the other hand, proof-of-personhood could be used as a building brick for a self sovereign identity system. Indeed, identity attributes are often data that by essence cannot change through time, or only under particular circumstances; names, date of birth etc.

In a ∞-renewable proof-of-personhood protocol, no one is able to change its certified address, which becomes a unique identifier for humans. Proof-of-personhood alone is not enough to run a self sovereign identity protocol. For example, in the context of a ∞-renewable protocol, we know that a person can claim only once their name, birthdate etc. So, we can be sure that no one can change their birthdate, for example. However, proof-of-personhood alone does not allow checking that the initially announced birthdate was true.

2.4 A Classification of Proof-of-Personhood Protocols

In this subsection, we provide a classification of known approaches to proof-of-personhood.

2.4.1 Centralized Identity Verification

Preventing Sybil attack can be considered through identifying user based on identity documents. However, this solution is currently not possible for a significant part of the population, which do not have access to identity documents [23]. Moreover, simply relying on images of such documents struggles to be a fraud-resistant and privacy-protective solution. If the privacy and security of an identity documents based proof-of-personhood protocol could be improved through cryptographic primitives, enabling a global and standard solution that would fit in the environment of the decentralized web still seems challenging.

2.4.2 Biometric Proof

Collecting biometric data can be used to check the uniqueness of a user before providing them a certification. Faces, or irises, are biometric attributes that are considered to have high enough entropy among the world's population so that no two humans have the same image of the attribute through a sufficiently precise sensor (e.g., a camera with a high enough resolution). In such protocol, in order to get a certification, a user must provide an image

of a biometric attribute. This image is then compared with the database of already certified users. If it is different enough from them, the new image is added to the database and the user gets a certification.

Such techniques have the advantage to be more inclusive than identity documents based solutions, as almost everyone has a face, or irises, etc. However, they can face security, privacy, and centralization issues.

The first option is that the user provides themself the image of their biometric attribute. For example, in the Proof-of-Humanity protocol, users are asked to upload a video of them speaking with their address visible [16]. It is hard to guarantee that the provided image is not fake. The advances in deepfake, and other image generation techniques make it hard to rely on self-provided biometric images in the long term, even if it is claimed that faking such videos is still difficult [16].

The other option is that users do not provide the image themselves. The Worldcoin protocol uses a specific hardware called the Orb to capture users' irises [24]. Only an image captured by a trusted Orb is accepted. Users must go to an operator who organizes the registration with an Orb, in order to have their irises captured. In this case, it is harder for a random user to provide a fake image of their irises. However, it results in a raise in the centralization of the protocol, where the operators and manufacturers of the Orb concentrate the power.

The issue of privacy can also be discussed. Despite potential improvements through encryption, biometric protocols inherently pose challenges to privacy.

2.4.3 Social Trust

Several approaches of Sybil-resistant and decentralized identities registry protocols are based on social trust. In 1992, the PGP creator Phil Zimmermann describes his so-called *web-of-trust*, which is a graph where nodes are public keys, and edges are a signature of a public key and the associated identity by another user who trusts the signed key/identity couple. In this context, if a key/identity couple is signed by a lot of other users, it is reasonable to assume that the key indeed belongs to the claimed identity. Even though the goal was not to create a proof-of-personhood protocol, but rather a public key infrastructure, this approach paved the way to social trust-graph based methods. In the context of online social networks, SybilRank [7] is aimed at detecting Sybil identities by analyzing the social graph

of friendships on online social networks. Based on PageRank [5] the algorithm is able to detect big regions of interconnected Sybil nodes. However, it does not offer a high enough accuracy (false positive and false negative) to be used in the context of proof-of-personhood.

Protocols such as brighId [4] or Upala [20] use social trust to achieve proof-of-personhood. A user is given a certification if they are trusted enough by other users. However, as stated in [9], it is hard to assume that even the closest relative to a person can be able to trust that they own only one account.

2.4.4 Simultaneous Tests

A wide range of protocols are based on **simultaneous tests**. A series of tests happen simultaneously, and is repeated on a regular basis. The tests are such that it is impossible to pass two at the same time, and only a person can pass a test. The users who passed their test get a certification, that is valid (i.e., not perished) until the next series of tests. As all tests happen simultaneously, one person can only get one certification.

We call a cycle the timeframe that starts at the beginning of a series of tests, and ends just before the next one, and we call the cycle period, the time between two series of tests (that is usually constant). The choice of the cycle period relies on the following tradeoff: on the one hand, the bigger it is, the more convenient it is for users as they need to take tests less frequently; on the other hand, new users risk to have to wait longer until the next series of tests to get a certification. Also, missing a series of tests is more serious, as the time the user will spend without a certification is bigger. We outline two main class of simultaneous tests.

Reverse Turing Tests Based

A reverse Turing Test is a problem that is designed to be hard to solve for a computer. CAPTCHAs (for Completely Automated Public Turing test to tell Computers and Humans Apart) are typical reverse Turing tests. The Idena protocol works with Flip tests, which are problems proposed by the community, to implement a proof-of-personhood protocol [15]. The user gets their perishable certification if they manage to solve enough problems within a limited time. This solution has the advantage to be easy, as it only requires an internet connection and a few minutes to get a certification. However, the gain in terms of convenience results in a loss in terms of security. Indeed, even if it is clear that one human is not able to solve

enough Flip tests to get hundreds or thousands of certifications, it is not clear that a well-trained human would not be quick enough to get 2 or a few more certifications. Moreover, even if the current public AI technologies seem incapable of solving flip test, it is not to exclude that a model able to solve flip tests could be developed in a more or less near future. Such a technology could be kept secret and result in a massive Sybil attack that could go unnoticed.

Physical presence Pseudonym parties are in-person meetings, where users must show their physical presence in order to pass the test. As it is impossible to be physically at two meetings at the same time, the presence at a meeting, when all meetings occur at the same time, constitutes a valid test for a simultaneous based proof-of-personhood protocol. The presence of a person is assessed by other persons, either users or organizers, depending on the type of meeting. The presence of a user is generally put on record by organizers or other users signing their public key.

Pseudonym Parties traditionally allow anyone around the globe to organize a party [9] [10]. Then, in order for a party to be trusted, they should allow anyone to witness that the party is well organized, and mainly that the number of certifications produced is not bigger than the number of persons present. For communities that are very far away, where witnessing is hard, this implies producing proofs, such as videos of the events to convince other members to trust this party. Indeed, it is easy for an attacker to claim that they organized a party in a place they know nobody will be able to come to witness the valid execution, and produce more certifications than the number of people actually present. The users of the protocol might expect a video of the party in that case, along with a proof that it was not filmed in advance. But using videos as a proof, such as in biometric-based protocols, represents serious flaws. The first one is a security weakness, that relies on the fact that a video can be faked, edited, AI-generated, etc. The progress in artificial intelligence shows more and more realistic deep fakes, and videos can less and less be accepted as proofs. Moreover, the diffusion of such videos might jeopardize the users' privacy. Indeed, they allow linking a group of addresses with a group of faces (or other recognizable traits). Even when these groups are big, after aggregating this data on several parties, it may become possible to identify the owners of the certified addresses.

3 The Encointer Local Protocol

3.1 Encointer Proof-of-Personhood

Similarly to the traditional Pseudonym Parties proposal [10], the Encointer protocol [3] is based on simultaneous test of physical presence. In the Encointer protocol, certifications are produced within local communities. Each community of the Encointer protocol is bounded to its own geographic area and has its own proof-of-personhood protocol. Users can choose to join their community according to where they live. Contrary to the traditional Pseudonym Parties proposal [10], users cannot choose to which parties they can go. Instead, they are randomly assigned to one of the meetings of their community. Each meeting involves a small enough number of people to not require organizers, and takes place within the community geographic area, which guarantees small travel distances for participants. They happen at a predefined frequency, and—as in the traditional Pseudonym Parties proposal—all at the same time. Certifications are produced during meetings, and are valid during one cycle (i.e., until the next series of meetings), which guarantees that a person can at most have one certification at a time. Encointer communities' protocols are therefore **perishable** and T-**renewable** protocols based on simultaneous physical presence tests, with T the (constant) period of cycles.

As users do not choose the place nor the people with whom they will meet, there is no need to convince the other members of the community that this particular meeting was executed correctly. Each member of the community, by participating, witnesses meetings and is able to assess the honesty of the other members. Even if there is a secret collusion (or several), it cannot control the fact that there will be honest participants assigned in their meetings, who will prevent them from faking a certification. There are witnesses in all parties by default.

The fact that all meetings take place in a defined community, bounded within a geographic area, in addition with small and randomized meetings, make it possible for a member of a community to have a great trust in the protocol of their community and its certifications, without requiring videos or other kind of proofs.

3.2 Encointer Economy: Universal Basic Income and Demurrage

Encointer protocol aims at allowing communities to run a local and equal opportunity economy where each community has its own currency. It implements a universal basic income (UBI), which members earn each time they participate in a meeting. These incomes come from freshly minted tokens.

Minting big amounts of tokens regularly without burning some would result in a constant increase in the supply of the currency, leading to high inflation. Encointer therefore includes a burning mechanism called demurrage. A percentage of the supply is constantly burned at a fixed rate. This traditionally affects all wallets and other form of holding equally. In Section 5, however, we introduce an exception to this rule for liquidity providing.

Demurrage along with UBI can be seen as a form of wealth redistribution. The amount of the UBI and the demurrage rate are decided by each community.

4 Problem Statement

As we have seen, the Encointer solution to proof-of-personhood has many advantages, including strong privacy. The protocol of a given community is secure when a big enough percentage of the population is honest (or when the biggest collusion represents a small enough percentage of the population). This assumption can be reasonably made for a participant of the community, who is able to assess the honesty of the other participants and the proper execution of its local community protocol.

However, assessing and trusting the proper execution of a foreign community may require more work. It is hard to expect from individuals to maintain an up-to-date list of the foreign communities that are trustworthy. Moreover, for certain applications such as protocol governance or global voting, communities of the protocol must reach a consensus on the set of communities that are trusted, and whose certifications can be taken into account in voting. Each community can of course run a vote to ask its member which foreign communities they trust. But firstly, we cannot expect members to do this demanding work and rely on it if they do not have skin in the game and incentives to do it well. Secondly, even if each community knows which they trust, it is not trivial to derive from this information the set of communities that can be trusted, as several dishonest communities can claim to

trust each other.

We consider a set of Encointer communities $\mathcal{C}$. Each community $A \in \mathcal{C}$ runs its proof-of-personhood protocol, and has its own set of certified addresses, its own currency whose symbol is denoted $¤_A$, and its demurrage rate denoted D_A. We note S_A, N_A and w_A respectively the supply of tokens $¤_A$, the population of community A (i.e., the number of certifications) and the amount of the UBI of community A in terms of $¤_A$.

A community is said to be honest or non Sybil if its protocol is not Sybil. On the contrary, a community is said to be Sybil or dishonest if it produces Sybil certifications.

We say that a person is honest if they behave accordingly to the protocol. In particular, they are not willing to own more than one certification, and they do not sign the keys of absent persons during meetings.

We assume that there can be a large number of Sybil communities, and they can create an arbitrarily large number of Sybil certifications.[1]

Moreover, an honest community can become Sybil at any time. In practice, this can happen if the number of participants become very low. In this case, a collusion can take control of the community and create fake certifications as there are no longer honest participants in the meetings.

The goal is to create and maintain, in a decentralized way, a set of trusted communities that we can reasonably assume to be honest. We call it the trusted set $\mathcal{T} \subset \mathcal{C}$. It is critical that the trusted set contains no Sybil communities. It is desirable that a large part of the honest communities is included in the trusted set.

If the communities' protocols of the trusted set are disjoint—which is the case if they agree on the same schedule[2] for the cycles so it is impossible to attend two meetings at the same time—the trusted set becomes itself a proof-of-personhood protocol, at a global scale.

Before building the trusted set layer, we address, in the next section, the

[1] In practice, there is a limit in the Encointer protocol on the growth per unit of time of the number of certifications a community produces during each cycle.

[2] In fact, the meeting schedules of different communities can be slightly different. The requirement that must hold is that it is impossible to attend two meetings of the same cycle. For any communities $A, B \in \mathcal{C}$ the delay between the meetings of A and the meetings of B must be strictly inferior to the minimum time needed to travel between the two communities' areas. Because of time zones, it may be useful to allow such delays.

problem of foreign exchange between Encointer communities, which will be a foundation for the former. The particularities of Encointer communities' economies, and especially demurrage, can raise liquidity issues.

5 Foreign Currencies Exchange for Encointer Communities

Currencies exchanges require the availability of liquidity and a price oracle mechanism, that is, defining the rate at which currencies are traded. In traditional order book exchanges, the price is defined by the meet between supply and demand. The liquidity of the exchange is very sensitive to the transactions volume.

In the decentralized context of web3 and cryptocurrencies, automated market makers (AMM) have become very popular for their high liquidity availability, that comes from the fact that an AMM hold liquidity that is constantly ready to be traded.

However, in the case of Encointer, the demurrage makes it difficult to expect pooling large amounts of liquidity, resulting in high slippage for users, which is a type of additional cost inherent to AMMs.

In the next subsection, we recall the specification and main mechanisms of a CPMM. More details can be found in specific publications [25] [1].

5.1 Recall on CPMM

An AMM is a smart contract that is able to hold assets, and to trade them with users.

Constant function market makers (CFMM) are a specific class of AMM where trades are allowed if and only if they keep invariant a particular function of the exchange variables. In particular, a constant product market maker (CPMM) between two base currencies $¤_A$ and $¤_B$ allows any trade that keeps the invariant $r_A r_B$ unchanged, with r_A (resp. r_B) the quantity of currency $¤_A$ (resp. $¤_B$) held by the CPMM, named the reserves. The reserves of the CPMM are provided by the so-called liquidity providers. They receive in exchange *share tokens* that represent their shares in the reserves and allow them to trade back their base tokens. We note $¤_{AB}$ the symbol of the share tokens.

Formally, a CPMM exports the following operations for users:

Operation: swap_for_B(x_A)
Requirement: $x_A > 0$ and user owns more than $x_A¤_A$
Outcomes: • $x_A¤_A$ are transferred from the user's wallet to the CPMM reserve • $x_A\frac{r_B}{r_A+x_A}¤_B$ are transferred from the CPMM reserves to the user's wallet

Operation: swap_for_A(x_B)
Requirement: $x_B > 0$ and user owns more than $x_B¤_B$
Outcomes: • $x_B¤_B$ are transferred from the user's wallet to the CPMM reserve • $x_B\frac{r_A}{r_B+x_B}¤_A$ are transferred from the CPMM reserves to the user's wallet

Operation: provide(x_A, x_B)
Requirement: $x_A > 0$ and $x_B > 0$ and $\frac{x_A}{x_B} = \frac{r_A}{r_B}$ and user owns more than $x_A¤_A$ and $x_B¤_B$
Outcomes: • $x_A¤_A$ and $x_B¤_B$ are transferred from the user's wallet to the CPMM reserves • $\frac{x_A}{r_A}S_{AB}¤_{AB}$ are minted and transferred to the user's wallet, with S_{AB} the supply of share tokens before the minting

Operation: withdraw(x_{AB})
Requirement: $x_{AB} > 0$ and user owns more than $x_{AB}¤_{AB}$
Outcomes: • $\frac{x_{AB}}{S_{AB}} r_A ¤_A$ are transferred from the CPMM reserves to the user's wallet • $\frac{x_{AB}}{S_{AB}} r_B ¤_B$ are transferred from the CPMM reserves to the user's wallet • $x_{AB}¤_{AB}$ are burned from the user's wallet

It can be checked that a swap operation always preserves the product $r_A r_B$, while a liquidity provision or withdrawal operation preserves the ratio r_A/r_B, and by extension the price.

Definition 7. *The **marginal price** when buying $x_B ¤_B$ (resp. $x_A ¤_A$) is defined by r_A/r_B (resp. r_B/r_A). It corresponds to the price in terms of asset $¤_A$ (resp. $¤_B$) paid when x_B (resp. x_A) is negligible compared to r_B (resp. r_A).*

The marginal price is also referred to as internal exchange rate, or spot price in the literature [1] [25] .

Definition 8. *The **swap price** when spending $x_A ¤_A$ to buy $¤_B$ (resp. spending $x_B ¤_B$ to buy $¤_A$) is defined by $\frac{r_B}{r_A + x_A}$ (resp. $\frac{r_A}{r_B + x_B}$). It corresponds to the price actually paid, in terms of $¤_A$ (resp. $¤_B$).*

The swap price is also referred to as swap rate or realized price in the literature [1] [25].

Each time an asset is bought from the CPMM, it becomes more expensive, as the amount of this asset in the reserves decreases while the other increases. Equivalently, each time an asset is sold to the CPMM, it becomes less expensive. This property is central.

An AMM is useful if it offers a swap price close to the market price, which is the price defined by supply and demand and that reflects the real value of the base assets. CPMMs exhibit two favorable properties, which make them a popular type of AMM: budget solvency and price discovery [1] [25].

Proposition 1. ***Price discovery:*** *In the presence of arbitrageurs, which are agents trading with the CPMM in order to maximize their net worth computed regarding the market price, the marginal price of the CPMM eventually*

aligns with the market price.

We assume the presence of arbitrageurs.

Proposition 2. ***Budget solvency:*** *It is impossible to empty the reserves of the CPMM, which means that it is always possible to trade.*

In practice, users pay the swap price, which is always higher than the marginal price, and by extension the market price. This discrepancy is called the **slippage**. It is associated with the fact that as one is buying an asset, its price increases. When the size of the trade is negligible compared to the reserve, the slippage is negligible, but if the size of the trade is too big compared to the reserves, the slippage becomes too high for the trade to be acceptable.

For the sake of simplicity, we have not included fees in the above specification. In practice, in order to reward the liquidity providers, each swap is charged with a certain percentage of fees. These fees are then spread between liquidity providers proportionally to the amount of share tokens they own. However, liquidity providers also face a loss in their net worth each time there is a change in the market price between the two assets that eventually results in a change in the marginal price. This loss is usually referred to as impermanent loss, or sometimes divergent loss [25].

Proposition 3. ***Impermanent Loss:*** *The impermanent loss is defined by* $1 - \frac{W_{provided}}{W_{hold}}$ *with* $W_{provided}$ *the net worth of the liquidity providers after a market price change and the associated arbitrage, and* W_{hold} *the net worth of the liquidity providers if they had kept their liquidity out of the AMM during the price change. For a price change equal to a multiplication by* ρ*, the impermanent loss is equal to* $1 - \frac{2\sqrt{\rho}}{1+\rho}$.

It is referred to as impermanent because if the market price returns to its original value without liquidity providers withdrawing their funds, the impermanent loss becomes null again. Liquidity providers do not fear price fluctuations around the mean, but rather long-term crashes or pumps.

In a nutshell, liquidity providers want to provide liquidity for a stable pair whose average price does not vary much in order to minimize their impermanent loss. They also want to provide liquidity for a pool that can offer high fees reward, i.e., a pool with a lot of transaction and/or with high fees.

In the next two subsections, we propose an adaptation of the CPMM approach to the specific case of Encointer. For each pair of Encointer commu-

nities, we consider a pool that exports the above operations of a CPMM. We note AB-pool the pool between two communities $A, B \in \mathcal{C}$, and we note r_A^{AB}, r_B^{AB} its reserves respectively in $¤_A$ and $¤_B$. When there is no ambiguity on the pool considered, we simply write r_A, r_B.

5.2 Pairing Contract to Balance Liquidity Providing Operations

AMMs typically implement liquidity providing operations that have no impact on the price. For CPMM, liquidity providing must be balanced, i.e., the ratio between the reserves is kept after the operation.

This implies that a liquidity provider provides both tokens. However, in the context of Encointer, participants of a community typically own only one type of token, which is their community's token.

We can either assume that liquidity providers will somehow acquire the tokens they need to provide liquidity (by trading with a peer for example, *over the counter*), or we can include in the design a way to provide liquidity with peers.

Note that some CFMM does allow unbalanced liquidity provision such as Balancer [19] for example. But in this case, the AMM performs a sequence of swaps with itself before providing liquidity, so the unbalanced liquidity operation can be decomposed in a sequence of swaps, and a pure liquidity (balanced) operation. In this case, *unpure* liquidity providing operations do have a price impact and a slippage cost.

We suggest another solution based on a smart contract aimed at balancing liquidity providing operation, by pairing funds from users owning one type of asset and users owning the other. A pairing contract is associated with each pool. When a user i wants to provide $x_A ¤_A$ in the AB-pool, they deposit their funds in the corresponding pairing contract. If there is already $y_B ¤_B$ waiting in the smart contract, then it automatically takes the maximum funds of both tokens that respects the ratio r_A/r_B of the AB-pool and provides these funds to the reserves.

Formally, with L_A, L_B the reserves of the pairing contract,[3] as soon as $L_A L_B \neq 0$, one of the following operations of the CPMM is called by the

[3]Since we focus on one pool, we simply write L_A, L_B and not L_A^{AB}, L_B^{AB}.

pairing contract:

$$\begin{cases} \texttt{provide}(L_A, L_A\frac{r_B}{r_A}) \text{ is called if } L_A r_B \leq L_B r_A, \\ \texttt{provide}(L_B\frac{r_A}{r_B}, L_B) \text{ is called else.} \end{cases}$$

The pool share token minted upon this operation are equally shared between the providers.

User i now owns shares in the pool, that represent ownership of tokens $¤_A$ **and** tokens $¤_B$. It is equivalently to i trading some of its tokens $¤_A$ for tokens $¤_B$ at the marginal price, to provide liquidity (without slippage but with possible wait time).

If there are no tokens $¤_B$ in the smart contract, then the funds of i wait until some tokens $¤_B$ are added to the contract.

This pairing contract could accept some additional requirements by the user, for example a certain price range in which the user is ready to provide liquidity. Outside this price range, the smart contract would not use the funds of that user.

5.3 Demurrage Reduction

Our pools export the same operations with the same specification as the standard CPMM defined above, with only one difference: the demurrage. Indeed, a demurrage can be applied to the reserves. We discuss the consequences on the exchange behavior, after discussing our demurrage reduction mechanism.

In typical CFMM, liquidity providers are incentivized to provide their funds by swap fees paid by the users, and that are distributed to the liquidity providers proportionally to their provision, i.e., the number of shares tokens they own.

However, in the case of Encointer, where a demurrage is constantly applied to liquidity, holding is expensive—and liquidity providing is a form of holding. To promote liquidity providing and keep it profitable, we suggest applying a smaller demurrage rate inside pools.

Each community applies its own demurrage rate to the supply. For the reason explained above, we suggest that a smaller rate is applied to pools. The choice of the `pool demurrage reduction` of a community has consequences on the evolution of its supply and its inflation.

A constant null demurrage rate removes the non-inflationary property of Encointer economies. A non-null, constant pool demurrage keeps the non-inflationary property, but is more or less equivalent to changing the demurrage rate of the economy, as everyone can use the pool to hold their fund (exception made of impermanent loss).

The idea is thus to limit the amount of funds that can benefit from a demurrage reduction, using a dynamic demurrage rate inside pools, that depends on the total amount of funds provided in all the pools, and that tends to the standard rate when it reaches a certain limit.

Formally, a community A decides at creation on a `pool demurrage rate` function $d_A \colon \mathbb{R}_+ \longrightarrow [0, D_A]$ with D_A the standard demurrage rate of the community. The demurrage applied on tokens $¤_A$ inside pools is then $d_A(R_A)$, with R_A the total amount of tokens $¤_A$ provided in pools, that is $R_A = \sum_{B \in \mathcal{C} \setminus \{A\}} r_A^{AB}$.

Null `pool demurrage rate`: $d_A = 0$

Constant `pool demurrage rate`: $d_A = d$, with $d \in [0, D_A]$

Dynamic `pool demurrage rate`: $d_A : \mathbb{R}_+ \longrightarrow [0, D_A]$

5.3.1 Targeted Amount of Liquidity

We suggest one of the following pool demurrage rate functions, which allows keeping the non-inflationary property of Encointer economies, and incentivizes liquidity providing up to a certain target of liquidity $T_A \in]0, S_A[$.

Step `pools demurrage rate`:

$$d_A(R_A) = \begin{cases} 0 & \text{if } R_A \leq T_A \\ D_A & \text{else} \end{cases}$$

with T_A a certain target of liquidity inside pools. It must be chosen wisely, as liquidity providers are not incentivized to provide more funds than this target. It can be chosen considering a slippage target given an average size of trade, and the number of commonly used pools.

Sigmoid `pools demurrage rate`:

One could want a continuous function, with the same idea, to prevent border effects. Sigmoid functions are typical continuous approximations of the step

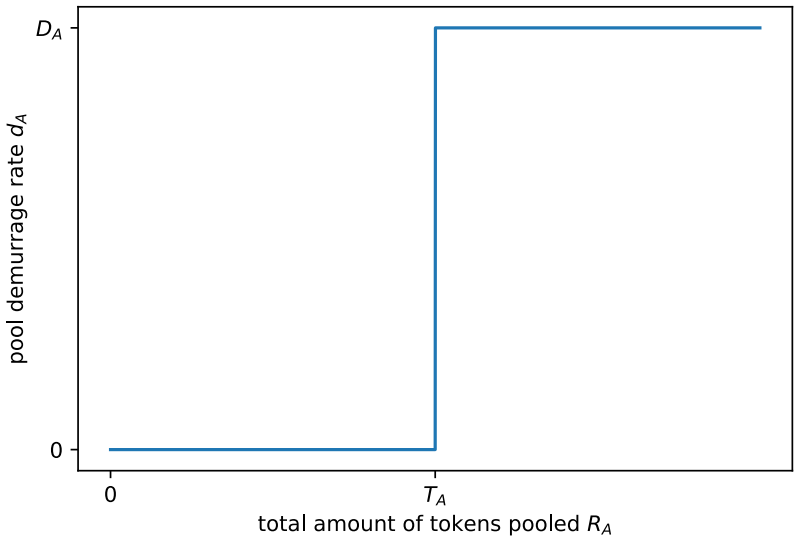


Figure 1: Step pool demurrage rate

function, with good regularity properties (the target being the inflection point).

For example, one could use a logistic function

$$d_A(R_A) = \frac{D_A}{1 + e^{-k(x-T_A)}}, \text{ with } k \in \mathbb{R}_+$$

or, an algebraic function

$$d_A(R_A) = \frac{D_A x}{\sqrt{1 + (x - T_A)^2}}.$$

Outside the case of a stationary economy (i.e., the population is constant) at equilibrium, the supply is not constant. In order to have a constant proportion of the supply committed to DEXs, we suggest using a target that depends on the supply. Formally, $T_A = \lambda_A S_A$ with $\lambda_A \in [0, 1[$.

Remark 1. *If community A has a stationary economy and a non-null demurrage, its final supply with a percentage λ_A of the supply that is not affected by demurrage is $S_A = \frac{N_A w_A}{1-(1-D_A)(1-\lambda_A)-\lambda_A} ¤_A$.*

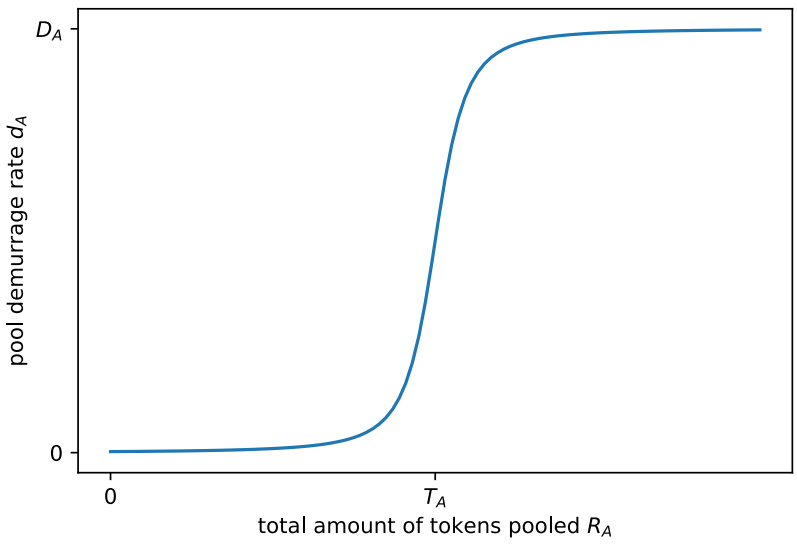


Figure 2: Algebraic pool demurrage rate

Proof. The supply in this case follows an arithmetico-geometric sequence: $S_A(t+1) = S_A(t)((1-\lambda_A)(1-D_A)+\lambda_A) + N_A w_A$ that converges to $\frac{N_A w_A}{1-(1-D_A)(1-\lambda_A)-\lambda_A}$. □

5.3.2 Consequences of the Demurrage Applied to Pools Reserves

For an AB-pool, if the pool demurrage applied to tokens ¤$_A$ is equal to the pool demurrage applied to tokens ¤$_B$, that is $d_A = d_B$, the demurrage is equivalent to a pure withdrawal operation. Indeed, as the same proportion of tokens ¤$_A$ and tokens ¤$_B$ are removed (and burned), the ratio r_A^{AB}/r_B^{AB} is preserved.

If $d_A \neq d_B$, it is equivalent to performing a swap followed by a liquidity withdrawal operation (the withdrawn funds are burned). Therefore, applying demurrage to the reserves is compatible with the standard CPMM specification and we keep their properties.

Proposition 4. *Applying demurrage $d_A, d_B \in [0,1[$ to a CPMM reserves is equivalent to performing a swap operation followed by a withdrawal operation if $d_A \neq d_B$. Only a withdrawal operation is needed if $d_A = d_B$.*

Proof. We note $r_A(0), r_B(0)$ the initial reserves, and we show that it is possible to arrive at $r_A(2) = r_A(0)(1 - d_A)$ and $r_B(2) = r_B(0)(1 - d_B)$ in a future state, with standard CPMM operations.

The first operation is a swap operation aimed at reaching the final reserves ratio, that is:

$$\frac{r_A(1)}{r_B(1)} = \frac{(1 - d_A)r_A(0)}{(1 - d_B)r_B(0)} = c\frac{r_A(0)}{r_B(0)}, \text{ with } c = \frac{1 - d_A}{1 - d_B}.$$

Moreover, a swap operation is possible if and only if it preserves the product invariant, that is

$$r_A(1)r_B(1) = r_A(0)r_B(0).$$

If $d_A = d_B$, the ratio is already reached and no swap operation is needed. Else, we have $c \neq 0$ and we can find a possible swap operation that reaches the right ratio by solving for $r_A(1), r_B(1)$ with the two precedent equations. We find $r_A(1) = \sqrt{c}r_A(0)$ and $r_B(1) = \sqrt{\frac{1}{c}}r_B(0)$.

We can now perform any withdrawal/deposit operation that preserve the reserves ratio, that is

$$\frac{r_A(2)}{r_B(2)} = \frac{r_A(1)}{r_B(1)}.$$

Moreover, we want to reach $r_A(2) = r_A(0)(1-d_A)$ and $r_B(2) = r_B(0)(1-d_B)$. This is possible through a withdrawal/deposit operation because it preserves the ratio, that is

$$\frac{(1 - d_A)r_A(0)}{(1 - d_B)r_B(0)} = \frac{r_A(1)}{r_B(1)}.$$

Finally, this corresponds to a withdrawal operation because $r_A(2) \leq r_A(1)$ since $d_A \in [0, 1[$.

□

However, if $d_A \neq d_B$, it creates a constant unbalancing of the marginal price r_A^{AB}/r_B^{AB} as the two reserves decrease at a different rate. The presence of arbitrageurs guarantees that this marginal price will be realigned with the market price, but this will result in a specific loss for liquidity providers that

is equivalent to an impermanent loss triggered by a change in the market price. We call this loss the unbalancing loss.

Theorem 1. *We consider an AB-pool that is affected by a pool demurrage d_A (resp. d_B) for tokens $¤_A$ (resp. $¤_B$). Liquidity outside the pool is affected by the same demurrage. The application of demurrage results in an unbalancing loss of* $1 - \frac{2\sqrt{\frac{1-d_B}{1-d_A}}}{1+\frac{1-d_B}{1-d_A}}$. *It is equivalent to an impermanent loss triggered by a market price multiplication by* $\rho = \frac{1-d_B}{1-d_A}$.

Proof. Token $¤_A$ is the numéraire, i.e, the token in terms of which we express worth and prices. This choice can be done without loss of generality. We assume a constant market price π expressed as the price of one token $¤_B$ in terms of the numéraire $¤_A$.

We first compute the net worth of liquidity providers in the case where they invest their liquidity in the pool, and then in the case where they hold their liquidity.

We note $r_A(t), r_B(t)$ the reserves of the pool at state t.

First case: liquidity provided State 0: the initial reserves are $r_A(0), r_B(0)$. This liquidity has been provided at the market price, i.e., $r_A(0)/r_B(0) = \pi$. The state is at equilibrium.

The net worth of liquidity providers is $W(0) = r_A(0) + \pi r_B(0) = 2r_A(0)$.

State 1: The demurrage is applied. The new reserves are $r_A(1) = (1 - d_A)r_A(0)$ and $r_B(1) = (1 - d_B)r_B(0)$.

State 2: Trades are issued by arbitrageurs until the marginal price of the pool realigns with the market price. Therefore, we have $r_A(2)/r_B(2) = \pi$. Because state 0 was at equilibrium, this gives

$$r_B(2) = r_B(0)\frac{r_A(2)}{r_A(0)} \tag{1}$$

Between state 1 and state 2, only swap operations are performed. The product invariant is kept: $r_A(2)r_B(2) = r_A(1)r_A(1)$. Substituting with equation from state 1 wet get

$$r_A(2)r_B(2) = (1 - d_A)(1 - d_B)r_A(0)r_B(0) \tag{2}$$

Solving for $r_A(2)$ with equations 1 and 2 we get $r_A(2) = \frac{r_A(1)r_B(1)}{r_B(2)} = \frac{(r_A(0))^2(1-d_A)(1-d_B)}{r_A(2)}$ and thus $r_A(2) = r_A(0)\sqrt{(1-d_A)(1-d_B)}$.

We can now compute the net worth of liquidity providers after the demurrage was applied:

$$W(2) = r_A(2) + \pi r_B(2) = 2r_A(2) = 2\sqrt{(1-d_A)(1-d_B)}r_A(0)$$

Second case: liquidity hold In this case, the liquidity providers keep their liquidity outside the pool. Liquidity still undergoes demurrage, but not arbitrage. We keep the notation r_A and r_B for the tokens owned by liquidity providers, but these tokens are now outside the pool and are not the reserves of the pool.

State 0: Liquidity providers hold $r_A(0)¤_A$ and $r_B(0)¤_B$ outside the pool.

State 1: Liquidity undergoes demurrage: $r_A(1) = (1-d_A)r_A(0)$ and $r_B(1) = (1-d_B)r_B(0)$.

State 2: There is no arbitrage, which would have had an effect on the liquidity: $r_A(2) = r_A(1)$ and $r_B(2) = r_B(1)$.

We compute the net worth

$$\begin{aligned} W(2) &= r_A(1) + \pi r_B(1) \\ &= (1-d_A)r_A(0) + (1-d_B)r_B(0)\frac{r_A(0)}{r_B(0)} \\ &= (1-d_A+1-d_B)r_A(0) \end{aligned}$$

Loss With the ratio of these two net worth, we get the loss:

$$L(d_A, d_B) = 1 - \frac{2\sqrt{(1-d_A)(1-d_B)}r_A(0)}{(1-d_A+1-d_B)r_A(0)} = 1 - \frac{2\sqrt{\frac{1-d_B}{1-d_A}}}{1+\frac{1-d_B}{1-d_A}}$$

Note that $L(d_A, d_B) = L(d_B, d_A)$, which justifies that the choice of the numéraire token has no importance.

□

This unbalancing loss could be small enough to be acceptable for liquidity providers. For example, in the conditions of Theorem 1, with $d_A = 0\%$

and $d_B = 10\%$ expressed as monthly rate, the unbalancing loss is approximately 0.14% per month. Moreover, in practice, the demurrage applied to liquidity outside pools will always be bigger, which minimizes the actual loss associated with the unbalancing of reserves triggered by different demurrage rates.

Still, if this loss is considered too big, it can make sense to apply either the max or the mean of the two demurrage rates on the pool's reserves to prevent unbalancing.

5.3.3 Encouraging Useful Liquidity Providing

With this design, there are two behaviors that could be desirable.

First, we want to encourage people to spread their liquidity between all trustworthy communities. Indeed, we do not want all the liquidity being concentrated on the most famous community, neglecting other communities that could be less canonical but still trustworthy to a certain point.

Second, we want to avoid people creating fake communities, to benefit from demurrage reduction without actually being useful to any real DEX.The problem is mainly utilitarian: demurrage reduction should be a reward for the service of providing liquidity to a useful pool.

Swap fees incentivize liquidity providers to provide liquidity for communities where there is little liquidity in the pool, as they will get a greater percentage of the swap fees in these pools. Therefore, liquidity providers are incentivized to provide for any pair that is trustworthy enough and for which there is a demand for trades. Swap fees are therefore the most natural answer to the two matters introduced above, as they incentivize liquidity providers to spread their funds wherever it is useful.

If it is observed that swap fees do not constitute a high enough incentive to foster these behaviors, the following options could be explored.

Liquidity Distribution Factor An additional factor is applied to compute the demurrage rate and act as a bonus/malus on demurrage reduction. This factor can be computed at different levels: per community, per pool, or per person.

The factor could measure the spread of funds provided across pools by a community A. It would be used to tune the demurrage reduction applied on any tokens $¤_A$ in pools: the reduction tends to zero when the funds are too

concentrated. This way, if liquidity providers do not spread enough their funds, they will face a higher demurrage rate. The same mechanism could be considered at a person level. In this case, there is one factor per person, that measures the spread of their own funds. This would have the advantage that each person faces the consequences of its own behavior. However, it adds complexity as there would be several demurrage rates inside a same pool for the same token.

At the pool level, the factor could depend on the amount of liquidity in a particular pool. The more there is liquidity in a given pool, the smaller the demurrage reduction would be. The demurrage reduction thus not only depends on the total amount of tokens provided in any pools, but also on the amount in the particular pool. This way, the demurrage reduction would be higher in underfunded pools, which would incentivize liquidity providers to spread their funds between trustworthy communities.

6 Extending Local Proof-of-Personhood

We utilize liquidity providing as an economically incentivized choice of which communities can be trusted as non Sybil. We derive a trust-graph between communities from this information. Then, we outline a mechanism based on PageRank [5] to build our trusted set $\mathcal{T}$.

6.1 A Web-of-Trust Based on Liquidity Providing

We want to derive a graph representing the trust relationship between communities that could be used to decide on which communities can be trusted. The intuitive idea is that if community A trusts community B, and community B trusts community C, then, to a certain extent, community A can trust community C. A trust-graph thus allows communities to get information on the trustworthiness of foreign communities, and possibly extend the set of communities they trust.

Definition 9. *A trust-graph is a directed graph whose nodes are entities of the system—the communities in our case—where an edge from a node A to a node B represents the trust of A in B.*

In our solution, we consider a weighted graph, in order to represent continuous levels of trust. The nodes being the communities around the world, we need a way to derive the trust edges between nodes to build the graph.

A first idea could be to build these edges through voting. Indeed, each

community runs its proof-of-personhood protocol and is thus able to run voting on any issue. One must keep in mind that the trust in a community should not be fixed and constant. Rather, it evolves, in function of the state of the trusted community, because an honest community can become dishonest at any time. In this context, the vote should be repeated regularly, or be continuous (i.e., voters can change their vote at any time, and the trust edges are constantly updated accordingly).[4] However, voting presents a major weakness. Voters have little skin in the game. Their incentive to invest time and energy to keep their vote up-to-date and well-informed is weak. Even worse, they could give false information on purpose, without risking or loosing anything. A claim that a foreign community is trustworthy can be considered more relevant if it is backed by a collateral staked by the claimer.

In our context, there is a particular activity when people risk their own money, by trusting foreign communities' proof-of-personhood protocols: liquidity providing. Indeed, if a foreign community is able to create Sybil certifications, it is also able to print unlimited money, since the amount of money printed for universal basic income is proportional to the number of certifications in the community. It becomes then possible to deplete the reserve of the honest token of the pool, which is owned by liquidity providers, or equivalently, it triggers a severe price change resulting in impermanent loss for liquidity providers. Thus, when a liquidity provider decides to deposit money in a pool between their community and a foreign community, they must trust the economic stability of that foreign community, and by extension, they must trust their proof-of-personhood protocol. Based on this argument, we extract the trust information from liquidity providing.

Definition 10. *We define the trust from community A to community B by*

$$\tau_{AB} = \frac{1}{\sqrt{U_A}\sqrt{S_A}} \sum_{i \in \mathcal{U}_A} \sqrt{p_{AB}(i)}$$

with $\mathcal{U}_A$ is the set of certified accounts in community A, $U_A = |\mathcal{U}_A|$, and $p_{AB}(i)$ the amount of tokens $¤_A$ provided by certified account i in an AB-pool.

This trust will be the weight of the edge from A to B in the trust-graph.

[4]This could be done with on chain smart contracts, by giving a vote token to each member of a community, and for each foreign community. The voters then deposit their tokens in either the yes, the no, or the neutral digital urn.

The square root introduces a sort of quadratic voting.[5] We show that the effect is that the influence of a single person, even with a lot of money, is reduced. Money is not enough to have an impact on trust, proof-of-personhood certifications are also required.

Theorem 2. *The trust from A to B can be expressed as the product of a term that represents the spread of liquidity provided by users, and a term representing the percentage of the supply committed to the particular pool:*

$$\tau_{AB} = \frac{\sum_{i \in \mathcal{U}_A} \sqrt{\alpha_{AB}(i)}}{\sqrt{U_A}} \frac{\sqrt{\tilde{R_{AB}}}}{\sqrt{S_A}}$$

Proof. Let $\tilde{R_{AB}} = \sum_{i \in \mathcal{U}_A} p_{AB}(i)$ the amount of tokens $¤_A$ provided in a AB-pool, by users with a proof-of-personhood certification in A. For all $i \in \mathcal{U}_A$, let $\alpha_{AB}(i) = \frac{p_{AB}(i)}{\tilde{R_{AB}}}$, which implies $\sum_{i \in \mathcal{U}_A} \alpha_{AB}(i) = 1$.

With this substitution, we get the result. □

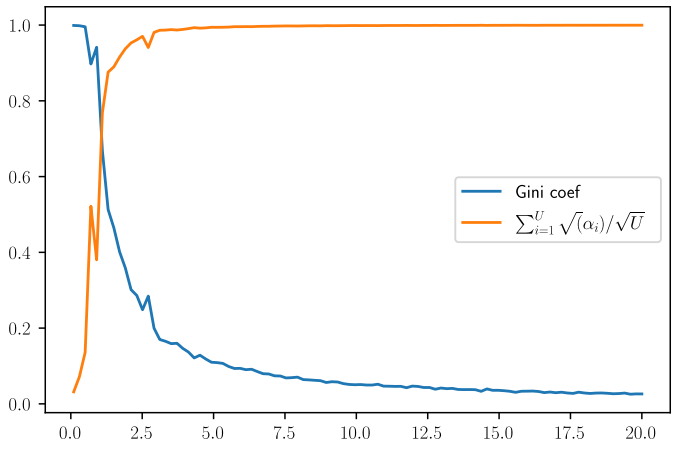


Figure 3: Gini coefficient and spread metric as defined above, for Pareto samples of shape between 0.1 and 20 on the x-axis. The sample size is 1000 and the scale parameter is 1.

[5] Though the choice of the exponent can be discussed, 1/2 is common and was proved to have optimality properties [18].

The first term can be viewed as a spread metric. Indeed, as we will see, the case that maximizes this metric is when funds are equally provided by liquidity providers, and in this case the term is equal to the square root of the percentage of users who provide liquidity.

Proposition 5. *We have* $0 \leq \tau_{AB} \leq 1$*, and more generally*

$$0 \leq \tau_{AB} \leq \sqrt{\frac{L_A}{U_A}}\sqrt{\frac{\tilde{R_A}}{S_A}} \leq 1$$

with L_A *the number of certified users of community A who provide liquidity in the AB-pool.*

Moreover, the first upper bound is reached when the amount of funds provided by users of community A, in the AB-pool, is equally spread among these users. The second upper bound, which is 1, is reached when, additionally, every user from community A is involved in liquidity providing with B, and all the supply of A is provided by these users.

Proof. The Cauchy-Schwartz inequality applied with the Euclidean scalar product to vectors $(\sqrt{\alpha_{AB}(1)}, \ldots, \sqrt{\alpha_{AB}(n)}), (1, \ldots, 1) \in \mathbb{R}^n$ gives

$$(\sum_{i \in \mathcal{L}_A} \sqrt{\alpha_{AB}(i)})^2 \leq n \sum_{i \in \mathcal{L}_A} \alpha_{AB}(i) = n$$

with $n = L_A$ and $\mathcal{L}_A = \{1, \ldots, n\}$ the subset of certified users that provide liquidity.

Moreover, the inequality is an equality if and only if the two vectors are linearly dependent, *i.e.,* $(\alpha_{AB}(i))_{i \in \mathcal{L}_A} \propto (1, \ldots, 1)$, which is equivalent to the funds being equally spread.

In addition, since by definition, $i \in \mathcal{U}_A \setminus \mathcal{L}_A \implies \alpha_{AB}(i) = 0$, we have

$$\sum_{i \in \mathcal{U}_A} \sqrt{\alpha_{AB}(i)} = \sum_{i \in \mathcal{L}_A} \sqrt{\alpha_{AB}(i)} \leq \sqrt{L_A}.$$

This inequality along with **Theorem 2** gives the result.

□

These interpretations of the trust are crucial. In order to increase the trust from A to B, two resources are needed: 1) certified addresses with the proof-of-personhood of community A, and 2) a quantity of tokens $¤_A$.

Proposition 6. *The total trust outgoing from a community is bounded when the number of communities is bounded. More precisely, with $A \in \mathcal{C}$, we have*

$$\sum_{B \in \mathcal{C} \setminus \{A\}} \tau_{AB} \leq \sqrt{|\mathcal{C}| - 1}.$$

*Moreover, the bound is reached when all the population and the supply of community A is involved in liquidity providing, and the funds are equally spread between liquidity providers **and** between foreign communities.*

Proof.

$$\begin{aligned}
\sum_{B \in \mathcal{C} \setminus \{A\}} \tau_{AB} &= \sum_{B \in \mathcal{C} \setminus \{A\}} \frac{\sum_{i \in \mathcal{U}_A} \sqrt{\alpha_{AB}(i)}}{\sqrt{U_A}} \frac{\sqrt{\tilde{R_{AB}}}}{\sqrt{S_A}} \\
&\leq \sum_{B \in \mathcal{C} \setminus \{A\}} \frac{\sqrt{L_A}}{\sqrt{U_A}} \frac{\sqrt{\tilde{R_{AB}}}}{\sqrt{S_A}} = \sqrt{\frac{L_A}{U_A S_A}} \sum_{B \in \mathcal{C} \setminus \{A\}} \sqrt{\tilde{R_{AB}}} \\
&\leq \sqrt{\frac{L_A}{U_A S_A}} \sqrt{|\mathcal{C} \setminus \{A\}|} \sqrt{\sum_{B \in \mathcal{C} \setminus \{A\}} \tilde{R_{AB}}}
\end{aligned}$$

The last inequality is obtained with the Cauchy-Schwartz inequality, applied similarly as **Proposition 5.**

The last term corresponds to the square root of the total amount of tokens A committed to pools by users with certifications from A. As each token can be committed to only one pool at a time, the last term is bounded by $\sqrt{S_A}$.

We thus get

$$\sum_{B \in \mathcal{C} \setminus \{A\}} \tau_{AB} \leq \sqrt{|\mathcal{C}| - 1}.$$

□

This proposition expresses the fact that the total trust that a community can give is limited.

As we have seen, the two resources required to create trust are certified accounts and money. Any dishonest organization is able to buy tokens from a community, as well as to get a certified address for each person of the organization who can attend meetings. They are thus able to buy trust from this community toward a potential dishonest community.

Proposition 7. *If a dishonest organization owns a fraction $\gamma \in [0,1]$ of the certifications of a community A, and a fraction $\sigma \in [0,1]$ of the supply of community A, it is able to buy an increase in the trust edge weight of $\sqrt{\gamma\sigma}$ to any other community B.*

Proof. We divide the set of certified addresses between the ones controlled by the dishonest organization and the others: $\mathcal{U}_A = \mathcal{D}_A \dot{\cup} \mathcal{H}_A$. We have

$$\tau_{AB} = \frac{1}{\sqrt{U_A}\sqrt{S_A}} \sum_{i \in \mathcal{H}_A} \sqrt{p_{AB}(i)} + \frac{1}{\sqrt{U_A}\sqrt{S_A}} \sum_{i \in \mathcal{D}_A} \sqrt{p_{AB}(i)}$$

The second term corresponds to the trust bought by the dishonest organization. We note $\tau^{\mathcal{D}}_{AB}$ this term. According to **Theorem 2**, it can be rewritten as $\tau^{\mathcal{D}}_{AB} = \frac{\sum_{i \in \mathcal{D}_A} \sqrt{\alpha_{AB}(i)}}{\sqrt{U_A}} \frac{\sqrt{r_{\tilde{AB}}}}{\sqrt{S_A}}$, with $r_{\tilde{AB}}$ the amount of tokens provided by addresses in $\mathcal{D}_A$ in a pool between A and B.

According to **Proposition 5**, the best strategy for the dishonest organization is to spread equally the funds between its member, and in this case we have

$$\tau^{\mathcal{D}}_{AB} = \frac{\sqrt{|\mathcal{D}_A|}}{\sqrt{U_A}} \frac{\sqrt{r_{\tilde{AB}}}}{\sqrt{S_A}}$$

If, moreover, the dishonest organization commits all of its tokens to pool (which is clearly the best strategy to maximize the trust), we get the result

$$\tau^{\mathcal{D}}_{AB} = \sqrt{\gamma\sigma}.$$

□

6.2 Graph Analysis and Trusted Set

Anyone can use the trust-graph as they want, and in particular use any trust-graph analysis algorithm to decide on a set of community they will trust. For example, a website from community A can decide to allow only requests made by a user with a certification from a community connected enough to A.

However, for some applications such as protocol governance or global voting, a standard analysis method is needed, as honest communities around the globe must reach a consensus on who is allowed to vote, i.e., they must derive a trusted set $\mathcal{T}$.

We suggest an analysis based on PageRank [5]. Indeed, we want to evaluate the communities' trustworthiness based on their position in the trust-graph, and it is natural that communities that are highly trusted by trustworthy communities are themselves trustworthy. PageRank allows exactly this: a node pointed to by well ranked nodes is itself well ranked. This recursive mechanism, the efficient computation by the power method [12], and the fact that it is only based on graph topology analysis and no other data makes PageRank a good choice for our problem.

6.2.1 Recall on PageRank

PageRank is a graph algorithm developed in 1998 and that was the foundation of Google's search engine [5]. It was designed to assign scores of importance to web pages, based on their incoming hypertext links. The algorithm relies only on the graph of the web (the pages as nodes and the links as directed edges) and not on the content of the pages. The idea is to base the scoring on the probability of presence of a random walker across the graph, where the probability of going from a node A to a node B corresponds to the ratio between the weight of the edge from A to B (0 if no edge) and the outdegree of A. This constitutes a Markov chain where the transition matrix is defined by the hypertext links ratios. However, general Markov chains do not necessarily admit a stationary distribution, which would be this probability of presence. Indeed, in general, graphs can admit sinks (i.e., nodes with a null outdegree) or regions that imprison the random walker. The approach of PageRank is to add a probability of *random restart* before each step that teleports the walker to a random node, and to perform a *forced restart* when the walker is on a sink.

Formally, PageRank is an algorithm that takes a graph (that in our case

is directed and weighted) as input and derives a score for each node of the graph, that represents their importance.

We represent a weighted directed graph by a set of nodes and a weight function: $F = (V, \omega)$, with $V = \{1, \dots, n\}$ and where $\forall v_1, v_2 \in V, \omega(v_1, v_2)$ is the sum of the weights of the edges from v_1 to v_2 with the convention that $\omega(v_1, v_2) = 0$ is equivalent to no edges from v_1 to v_2. We do not consider the number of edges from one node to another, but just the total weight from one node to another. We note $\deg^+{}_F(v) = \sum_{u \in V} \omega(v, u)$ the outdegree of node v.

For a graph $F = (V, \omega)$ and a subset of its nodes $U \subset V$, we note $F[U]$ the subgraph of F induced by U, defined by $F[U] = (U, \omega_{|U^2})$.

Definition 11. *For a graph $F = (V, \omega)$, we note $\mathrm{PR}_{F,\delta} \in \mathbb{R}^{|V|}$ the PageRank scores vector, which is defined by*

$$\mathrm{PR}_{F,\delta} = \delta \, \mathrm{PR}_{F,\delta} \, T + (1 - \delta) J$$

where $1 - \delta$ is the probability of random restart chosen as a parameter, and J and T are defined by $J = (1/n, \dots, 1/n)$ and

$$\forall i, j \in V, T_{ij} = \begin{cases} \frac{\omega(i,j)}{\deg^+{}_F(i)}, & \text{if } \deg^+{}_F(i) > 0 \\ \frac{1}{n}, & \text{otherwise,} \end{cases}$$

with $n = |V|$. We note $\mathrm{PR}_{F,\delta}(i)$ the i-th element of $\mathrm{PR}_{F,\delta}$ that corresponds to the score of node i.

There are several possible normalizations of the PageRank scores. We consider the normalization such that $\sum_{v \in V} \mathrm{PR}_{F,\delta}(v) = 1$. It can be shown that this PageRank vector corresponds to the stationary distribution of the Markov chain described above.

As the parameter δ is often fixed in our work, we only write PR_F when there is no ambiguity.

6.2.2 Trusted Set

We note G our trust-graph defined by $G = (\mathcal{C}, \omega)$ with $\forall A, B \in \mathcal{C}, \omega(A, B) = \tau_{AB}$.

We cannot simply rely on PR_G to derive the trusted set $\mathcal{T}$. Indeed, $\mathcal{C}$ can contain an arbitrarily big number of Sybil communities, which allow them

to have high scores. Considering a trust-graph consisting of two separate clusters of nodes, one honest and one dishonest, if there are no edges from the dishonest cluster to the honest cluster, then it can be shown that the total PageRank score of the dishonest cluster is greater than $\frac{n_s}{n_h+n_s}$ with n_s the number of dishonest nodes and $n_h = |\mathcal{C}| - n_s$ the number of honest nodes. Since n_s can be arbitrarily bigger than n_h, the total PageRank of the dishonest cluster tends to 1 while the total PageRank of the honest cluster tends to 0.

This phenomenon is related to a kind of attack traditionally called spamdexing in the context of search engine: creating a big number of dummy nodes pointing to a target node to inflates its score.

A common response to this problem is personalized PageRank. When a random restart occurs, instead of choosing the new node with a uniform distribution over all the nodes, certain predefined nodes have higher chance of being selected (a custom distribution is chosen for the vector J of Definition 11). These nodes are called the seeds. As honest nodes are poorly connected to Sybil regions, the fact that the walker starts from an honest node makes its probability of presence higher in the honest regions.

For example, TrustRank [13] suggests a seed-based approach in the context of search engine to prevent spamdexing. A set of recognized web pages are used as seeds. In the context of detecting Sybil attacks on online social network, SybilRank [7] proposes a seed-based approach where the seeds are a set of hand-selected trustworthy accounts. Though this solution is useful to detect broad regions of Sybil nodes, it fails to detect a significant amount of Sybil nodes, making it hardly applicable to proof-of-personhood.

Moreover, in our context, we can hardly decide on a set of seeds in a decentralized way. And even if we could, this set of seeds would require to be constantly updated, as we assume that an honest community can become Sybil at any time.

In addition, since anyone is able to create Sybil communities, and perfectly control their outgoing trust edges, it is particularly hard to distinguish Sybil from honest nodes. Most likely, honest communities will be connected, but Sybil ones can be too. This would result in a graph with clusters of honest nodes, and cluster of Sybil nodes that are indistinguishable only looking at the graph at a given moment. Whatever the honest cluster will look like, a Sybil one can look the same. Therefore, we assume the first communities of the protocol to be initially honest, and base our trusted set on this initial

trusted setup. Note that initially honest does not mean that they cannot become Sybil later (however, we do assume that a majority of honest community in the trusted set cannot all become Sybil at the same time).

For these reasons, we define the trusted set with a recurrence relation, i.e., the new state of the trusted set depends on its previous state.

We define an *acceptance criterion* $a_{G,\mathcal{T}} : \mathcal{C} \to \{0,1\}$ that depends on the current trusted set $\mathcal{T}$ and that determines whether a community can be included in the next trusted set. A community $A \in \mathcal{C}$ that is candidate is included in the trusted set if and only if $a_{G,\mathcal{T}}(A) = 1$. We use the exact same criterion to determine whether a community already included in the trusted set $\mathcal{T}$ can stay. If $A \in \mathcal{T}$ and $a_{G,\mathcal{T}}(A) = 0$, then A is excluded from the trusted set. Thus, the criterion to enter the trusted set is the same as the criterion to stay in the trusted set.

Definition 12. *We define the acceptance criterion by*

$$\forall A \in \mathcal{C}, a_{G,\mathcal{T}}(A) = \begin{cases} 1, & \textit{if } \mathrm{PR}_{G[\mathcal{T}\cup\{A\}]}(A) \geq \dfrac{\beta}{|\mathcal{T}\cup\{A\}|} \\ 0, & \textit{otherwise} \end{cases}$$

where A is the candidate community, and $\beta \in [0,1]$ is a tolerance parameter.

This formula simply expresses the idea that a community is accepted, if and only if its score is not too much below the average PageRank score, which is $1/n$ with n the number of nodes.

In practice, there can be several candidates at the same time. In this case, the candidate with the highest PageRank is examined first. Formally, with $\Gamma \subset \mathcal{C} \setminus \mathcal{T}$ the set of candidates, the first candidate to be examined is $\arg\max_{A\in\Gamma} \mathrm{PR}_{G[\mathcal{T}\cup\{A\}]}(A)$.

To know if a community in the trusted set is to be excluded, the PageRank scores $\mathrm{PR}_{G[\mathcal{T}]}$ are regularly computed, and any community that would not satisfy the acceptance criterion is immediately excluded.

If a community in the trusted set becomes Sybil, liquidity providers will withdraw their funds from its pools, resulting in making its PageRank score low and the community to be excluded. We can expect a high reactivity from the financial agents, and the community to be excluded before its Sybil certifications have made damages.

Parameter β must be chosen considering a tradeoff between inclusivity and security: the bigger β, the more honest communities will be included in the trusted set, but the bigger the risk is to include a Sybil community.

6.3 Potential Threat and Sketch of Defense

6.3.1 Spam

Though PageRank algorithm is well suited for big graphs, a spam attack could consist in filling the set of candidate communities Γ with a big number of dummy communities. This would make it difficult to compute the best candidate $\arg\max_{A\in\Gamma} \mathrm{PR}_{G[\mathcal{T}\cup\{A\}]}(A)$. Though there could be possible computational optimizations, it can make sense to charge fees or to require an endorsement from a trusted community in order to be considered as a candidate. This would help to limit the size of Γ and spam attack on the protocol.

6.3.2 Buying Trust

To provide trust in our model, two resources are needed: money and proof-of-personhood certifications. Therefore, a dishonest organization that controls a certain number of legit certifications in trusted communities, and that can buy a certain amount of tokens, is able to have an influence on the trust-graph. However, based on the way trust is computed, such an attack costs a significant percentage of the supplies of a community, and a significant percentage of the certifications of a community, as shown in Proposition 7. What is taken into account in the PageRank for the probability to go from node A to node B is the proportion of the edge from A to B over the total outdegree of B. This means that if a community has a very low outdegree, then buying trust from this community has more impact on the PageRank scores. Thus, the presence of a community with a high PageRank score but a low outdegree could be dangerous. A solution could be to increase the probability of random restart for communities with low outdegrees, which is equivalent to adding edges of small weights to all the other communities in the PageRank computation. This way, every community would have an outdegree of similar magnitude.

6.3.3 Twin Sybil Communities

When a community is not trustworthy anymore, incoming trust links vanish, and the community is excluded from $\mathcal{T}$.

However, if a dishonest organization takes control of several communities in $\mathcal{T}$ at the same time, as these communities can provide trust to each other, they can keep a significant PageRank score and not be excluded.

If there are no edges from honest nodes to these controlled communities, and there are no sinks among the honest nodes, the only way for the random walker to go from honest nodes to the controlled cluster is through random restart. However, even in this case, the total PageRank score of the controlled communities can reach $\frac{k}{n}$, with k the number of controlled communities and $n = |\mathcal{T}|$. In particular, if there are no edges from the controlled communities to honest nodes, and if the graph of controlled communities is strongly connected with edges of equal weights, then each controlled community has a PageRank score of $\frac{1}{n}$, which is enough for each of them to stay in the trusted set.

To perform such an attack, the k communities must be taken at the same time, so they continue providing trust to each other and not be excluded from $\mathcal{T}$. A number of controlled communities $k \geq 2$ is enough for an attacker to maintain two Sybil communities in $\mathcal{T}$.

It is required that two communities become dishonest **at the same time**. Indeed, if the first community is known to be no longer trustworthy before the second community becomes dishonest, then the latter trust to the former will quickly decrease, as well as the trust from other communities in the trusted set, resulting in the first community being excluded. Taking control of two communities at the exact same time seems hard, and assuming that it will not happen could be reasonable.

Still, it is also possible to allow communities of the trusted set to vote to exclude other communities, based on a 1-community-1-vote. This would allow honest communities to exclude two dishonest communities maintaining a high PageRank score by providing trust to each other, as explained above.

6.4 Security Models and Future Work

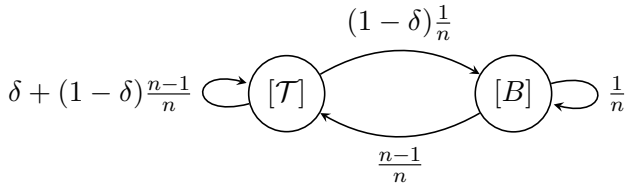
Formally studying the security of this approach is hard. It implies optimization problems and requires economic models. We introduce four models to study our solutions, from the simplest one to the most complete one. Several problems stay open for future work.

6.4.1 Dishonest communities only receive trust from dishonest communities

In this model, we assume that dishonest communities get trust only from dishonest communities. In particular, if an initially honest community becomes dishonest, all trust edges from honest communities are assumed to disappear as soon as the community becomes dishonest. This also means that liquidity providers from honest communities only provide liquidity for communities they are sure to be honest.

In this case, we can show that the trusted set contains only honest communities. Consider a trusted set $\mathcal{T}$ initially honest. We assume that $|\mathcal{T}| \geq 2$ and that there are no sinks in $G[\mathcal{T}]$ (this is reasonable, as only one token provided by one user from each community is enough).

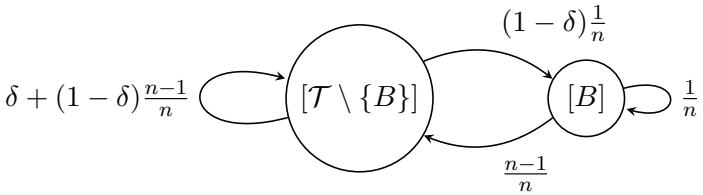
No dishonest communities can be included Let B a dishonest community that is candidate to be included in $\mathcal{T}$. To compute its PageRank score $\mathrm{PR}_{G[\mathcal{T}\cup\{B\}]}(B)$, we can consider a Markov chain with two states: a state $[B]$ corresponding to the random walker being on node B, and a state $[\mathcal{T}]$ corresponding to the random walker being on one of the nodes already in $\mathcal{T}$. We note $n = |\mathcal{T}| + 1$. The probability for a walker in state $[\mathcal{T}]$ to stay in the same state is equal $\delta + (1-\delta)\frac{n-1}{n}$. Indeed, since there are no edges from honest communities to B, and all communities of $\mathcal{T}$ are currently honest, and there are no sinks in $G[\mathcal{T}]$, exception made of the random restart, a walker in state $[\mathcal{T}]$ stays there. In case of random restart, the probability to stay there is $\frac{n-1}{n}$. The best strategy for B is to be a sink, i.e., there are no edges from B to any community in $\mathcal{T}$. In this case, a walker in $[B]$ will be forced restarted and its probability to stay in $[B]$ is equal to $1/n$. We thus have the following Markov chain.



The probability of presence in state $[B]$ obtained from the stationary distribution of the Markov chain above is equal to $\mathrm{PR}_{G[\mathcal{T}\cup\{B\}]}(B)$ and can be computed as $\frac{1-\delta}{n-\delta}$. The condition for B to not be included is $a_{G,\mathcal{T}}(B) = 0$, which thus translates into $\frac{1-\delta}{n-\delta} < \frac{\beta}{n}$, or equivalently $\beta > \frac{1-\delta}{1-\delta/n}$. With the

classic choice of $\delta = 0.85$ and $n = 3$ (worst case), we need $\beta > 0.209302$ to guarantee that no dishonest communities can be included.

A community that becomes dishonest is excluded We now consider the case of a community B initially honest and included in $\mathcal{T}$ that becomes dishonest. By assumption, as soon as B becomes dishonest, all incoming links from the other communities of $\mathcal{T}$ are removed, i.e., honest liquidity providers withdraw their funds. We obtain the same Markov chain (with this time $n = |\mathcal{T}|$), where the probability for a walker to arrive in state $[B]$ is due to random restart and forced restart.



With a similar reasoning than previously, we have that if $\beta > \frac{1-\delta}{1-\delta/n}$, then community B is excluded from the trusted set.

The condition on β are the same in the two cases. The criterion is so that it is as hard to stay as it is to be included.

6.4.2 Dishonest communities can buy trust from any community at the same price

In practice, as we have already discussed, it is possible to buy a trust edge from any honest community, with two resources: money and a number of persons. Our solution is designed so that it is very expensive in terms of these two resources to get as much incoming trust as a well-recognized, honest community. Parameter β must be large enough so that it is impossible to buy an inclusion in the trusted set, given the maximum resources that we assume a dishonest organization can have. Computing the cost of the attack would be useful to choose such parameter.

In this model, we consider that all trusted communities have equal market capitalization, and equal population size. With this assumption, the resources of the attacker in terms of money represents the same percentage of the supply regardless of the community considered. Likewise, the resources of the attacker in terms of personhood represents the same percentage of the population regardless of the community considered. Let $\gamma \in \mathbb{R}_+$ be the

number of supplies the attacker can afford. For example, $\gamma = 0.1$ means that the attacker is rich enough to buy 10% of the supply of a community (recall that all supplies have the same value). Similarly, let $\sigma \in \mathbb{R}_+$, the number of populations the attacker controls.

With these resources, the attacker wants to optimize the PageRank of a dishonest community D. We note τ^0_{AB} the trust edge from A to B before the attacker has bought trust, and τ_{AB} after the attacker has bought trust. The optimization problem related to the best buying strategy for the attacker is

$$\begin{aligned}
\text{Maximize} \quad & \mathrm{PR}_{G[\mathcal{T}\cup\{D\}]}(D) \\
\text{Subject to} \quad & G = (\mathcal{C}, (A, B \mapsto \tau_{AB})) \\
& \forall A, B \in \mathcal{T} \cup \{D\}, \tau_{AB} = \tau^0_{AB} + \sqrt{\gamma_{AB}\sigma_{AB}} \\
& \sum_{A,B\in\mathcal{T}\cup\{D\}} \gamma_{AB} \leq \gamma \\
& \sum_{A,B\in\mathcal{T}\cup\{D\}} \sigma_{AB} \leq \sigma.
\end{aligned}$$

The question of the cost of the attack is, given parameter β, what would be the resources needed for a dishonest community to be included in $\mathcal{T}$ by buying trust. Or formulated differently, how large should be β, so that a dishonest community with the optimal buying strategy cannot be included, given the maximum resources we assume it can have.

6.4.3 Dishonest communities can buy trust from community with specific prices

In reality, communities will have different market capitalization and different population size. As a result, some communities edges are cheaper to buy. However, we can expect that communities with the largest market capitalization and the largest population are also communities that attract liquidity providers the most, because their economy could be stronger and they could generate plenty of transactions. Consequently, we can expect that communities that have the highest PageRank scores are also communities that are the most expensive to buy trust from. Studying the system in this context therefore requires economic considerations.

6.4.4 Honest liquidity providers can make mistakes

Finally, we can relax the assumption that honest liquidity providers only provide liquidity for honest community. They are economically incentivized to not provide liquidity to dishonest communities, but in practice, they may make mistakes. Thus, the model must account for the possibility that a certain percentage of the liquidity provision from honest communities are *mistakes*, i.e., they are provided to a pool with a dishonest community. In addition, liquidity providers may also not react instantaneously to withdraw their funds in the event of a community becoming dishonest unpredictably. This could give time for a second community to become dishonest and lead to the scenario introduced in Paragraph 6.3.3.

7 Conclusion

We addressed two problems related to enabling interactions between Encointer communities: foreign currency exchange with demurrage, and global proof-of-personhood protocol. Concerning the design of a DEX, we showed that demurrage is compatible with the standard specification of CPMMs, but can raise liquidity issues. We suggested a demurrage reduction mechanism to incentivize liquidity providing. The second part of the work, focused on proof-of-personhood, suggests an approach that would allow benefiting, at a global scale, from the numerous advantages of the Encointer local proof-of-personhood protocol: decentralization, security and privacy. We based this extension on a trust-graph that is derived from the behavior of liquidity providers, as we argue that they have economic incentives to assess the trustworthiness of foreign communities' proof-of-personhood protocols. Moreover, contrary to standard social trust and web-of-trust approaches, our graph exhibits trusts between communities instead of individuals. Finally, we derived a trusted set using a criterion based on a PageRank analysis of the trust-graph, to constitute a global proof-of-personhood protocol.

References

[1] Massimo Bartoletti, James Hsin-yu Chiang, and Alberto Lluch-Lafuente. "A theory of automated market makers in defi". In: *Logical Methods in Computer Science* 18 (2022).

[2] Monica Bianchini, Marco Gori, and Franco Scarselli. "Inside pagerank". In: *ACM Transactions on Internet Technology (TOIT)* 5.1 (2005), pp. 92–128.

[3] Alain Brenzikofer. "encointer–Local Community Cryptocurrencies with Universal Basic Income". In: *arXiv preprint arXiv:1912.12141* (2019).

[4] *brightID Universal Proof of Uniqueness*. `https://www.brightid.org/whitepaper`. Whitepaper. Last Printed: January 20, 2022.

[5] Sergey Brin and Lawrence Page. "The anatomy of a large-scale hypertextual web search engine". In: *Computer networks and ISDN systems* 30.1-7 (1998), pp. 107–117.

[6] Vitalik Buterin et al. "A next-generation smart contract and decentralized application platform". In: *white paper* 3.37 (2014), pp. 2–1.

[7] Qiang Cao et al. "Aiding the detection of fake accounts in large scale social online services". In: *9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*. 2012, pp. 197–210.

[8] Federico Ast Clément Lesaege and William George. *Kleros Short Paper v1.0.7*. Tech. rep. 2019. URL: `https://kleros.io/whitepaper.pdf`.

[9] Bryan Ford. *Identity and Personhood in Digital Democracy: Evaluating Inclusion, Equality, Security, and Privacy in Pseudonym Parties and Other Proofs of Personhood*. 2020. arXiv: `2011.02412 [cs.CY]`.

[10] Bryan Ford and Jacob Strauss. "An offline foundation for online accountable pseudonyms". In: *Proceedings of the 1st workshop on Social network systems*. 2008, pp. 31–36.

[11] A Shaji George, AS Hovan George, and T Baskar. "Worldcoin: A Decentralized Currency for a Unified Global Economy". In: *Partners Universal International Research Journal* 2.2 (2023), pp. 136–155.

[12] David F Gleich. "PageRank beyond the web". In: *siam REVIEW* 57.3 (2015), pp. 321–363.

[13] Zoltán Gyöngyi, Hector Garcia-Molina, and Jan Pedersen. "Combating web spam with trustrank". In: *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*. 2004, pp. 576–587.

[14] Eric Hughes. "A Cypherpunk's Manifesto". In: *The Electronic Privacy Papers: Documents on the Battle for Privacy in the Age of Surveillance*. USA: John Wiley & Sons, Inc., 1997, pp. 285–287. ISBN: 0471122971.

[15] *Idena Whitepaper*. Accessed on November 2023. URL: `https://docs.idena.io/docs/wp/summary/`.

[16] Stuart James. "Proof of Humanity - an explainer". In: *Kleros* (Mar. 2021). URL: `https://blog.kleros.io/proof-of-humanity-an-explainer/`.

[17] Ari Juels, Dario Catalano, and Markus Jakobsson. "Coercion-resistant electronic elections". In: *Proceedings of the 2005 ACM workshop on Privacy in the electronic society*. 2005, pp. 61–70.

[18] Steven P Lalley and E Glen Weyl. "Quadratic voting: How mechanism design can radicalize democracy". In: *AEA Papers and Proceedings*. Vol. 108. American Economic Association 2014 Broadway, Suite 305, Nashville, TN 37203. 2018, pp. 33–37.

[19] Fernando Martinelli and Nikolai Mushegian. "A non-custodial portfolio manager, liquidity provider, and price sensor". In: *URl: https://balancer.finance/whitepaper* (2019).

[20] Peter Porobov. *upala Documentation*. `https://docs.upala.id/_/downloads/en/latest/pdf/`. 2022.

[21] Sasha Shilina. "Revolutionizing identity verification: An introduction to Proof of Personhood (PoP) protocols". In: ().

[22] Divya Siddarth et al. "Who watches the watchmen? a review of subjective approaches for sybil-resistance in proof of personhood protocols". In: *Frontiers in Blockchain* 3 (2020), p. 46.

[23] Olivia White et al. "Digital identification: A key to inclusive growth". In: (2019).

[24] *Worldcoin Whitepaper*. Accessed on November 2023. URL: `https://whitepaper.worldcoin.org/`.

[25] Jiahua Xu et al. "Sok: Decentralized exchanges (dex) with automated market maker (amm) protocols". In: *ACM Computing Surveys* 55.11 (2023), pp. 1–50.