Backstop Layer 2: A forkable L2 to secure oracles and arbitration systems

Draft 0.2

Edmund Edgar[1], Alexander Herrmann[2], Silke Noa Elrifai[3]

Abstract

We describe a design for an L2 ledger with an enshrined dispute resolution oracle. Contracts deployed on the ledger benefit from oracles and governance arbitration backstopped by the ability, in extremis, to fork the entire L2 ledger on which they are running. Decisions about assets issued on the forkable ledger benefit from "subjectivocratic" security. Tokens bridged from other ledgers can also be managed, albeit with an economic security bound. Tokens representing Real-World Assets can be bridged in the same way, or by extending the trust users already place in their issuer.

[1]Founder of https://reality.eth.link/
[2]Crypto Researcher / https://www.linkedin.com/in/alexander-herrmann-a98573a5/
[3]Crypto Lawyer and Researcher

November 24, 2023

Contents

1. Introduction

A famously thorny problem for applications running on smart contract platforms is the Oracle Problem: How do programs running on blockchains get information about the outside world, in such a way that it cannot be profitably manipulated?

Decentralized Finance (Defi) protocols need external market data. Prediction markets need data about political or sports events for resolution. Insurance protocols need information about events that trigger claims. More broadly, smart contract systems often need access to information that is not purely algorithmic in nature. Escrow contracts sometimes result in disputes that need to be resolved. Decentralized content moderation systems need to determine if a user broke the rules. Moreover, decentralized systems often need to be upgraded, so they need to be able to distinguish a legitimate upgrade from a malicious one. The building block required can be broadly termed decentralized adjudication.

Attempts to tackle the Oracle Problem, or more broadly perform the function of adjudication in decentralized systems, have often rested ultimately on token-voting and token-staking mechanics. Such mechanics can typically be defeated by controlling a majority of the tokens, either through bribery or by simply purchasing tokens on the open market. The risk of such attacks in turn creates an *economic security bound* where the system is only secure against a rational, self-interested adversary if the cost of controlling the majority of the tokens is greater than the amount of value that can be stolen by making the oracle misreport.

We propose a solution to the Oracle Problem, and the related problem of decentralized Adjudication, using Subjectivocracy.

Our design uses a forkable L2 to create a ledger with a built-in adjudication system for decentralized applications. Contracts deployed on the L2 can take advantage of a layer of a built-in adjudication system. If a dispute arises about the functioning of the adjudication system, and it is sufficiently escalated, the entire L2 ledger can be forked, whereby the users choose whichever fork they consider the legitimate one. Forks duplicate the entire interconnected ecosystem of contracts, leaving relationships between contracts on each version of the ledger unchanged.

With “subjectivocratic” adjudication enshrined as a final backstop in the ledger, systems that are currently reliant on multisigs or insecure token mechanics can be redeployed with enhanced security with minimal modifications. The system enables more secure oracle infrastructure, more solid dispute resolution, less vulnerable management for upgradeable contracts, and a more sustainable base for governing Decentralized Autonomous Organizations (DAOs).

2. Background

2.1. Subjectivocracy

Vitalik Buterin coined the term "subjectivocracy" to describe how a ledger can be split into multiple forks, with users using the one they prefer. Subjectivocracy

represents the ultimate opt-in, exit-based governance system: Users are always free to use the fork that they consider represents the system working as it should. Additionally, since we may expect that the chain that faithfully embodies the users' expectations about the system's rules will be more useful and assets on it more valuable, the users' preferences will be reflected in price signals, providing objective information that can be observed about the subjective choice of users.

Subjectivocracy differs from other mechanisms commonly used to secure blockchain protocols such as token voting and Schelling Games which are fundamentally *economic* in nature: they can be defeated by spending some amount of money to buy tokens or bribe participants. This in turn limits the amount of value that can be secured before a rational actor will spend this amount of money attacking the system. Systems relying on *economic security* are thus subject to an *economic security bound*.

In contrast, subjectivocracy simply consists of users making independent judgments. A well-resourced attacker may be able to deploy their wealth and influence to *encourage* people to adopt their preferred fork in the hope of building a network effect around it, and also to manipulate objective measures like price signals. However, no amount of money can force a determined user to use a fork they consider corrupt over one they consider to be functioning correctly.

Subjectivocracy has often been used in the governance of blockchain base layers. In Ethereum, protocol upgrades are effectively subjectivocratic: Developers propose a change to the protocol rules, and users are free to use it or reject it as they see fit. Such upgrades are usually uncontroversial, but may on occasion result in enduring economic forks like the split between Ethereum and Ethereum Classic. A subjectivocratic solution is also available in the event that participants attempt to capture the system in ways that the users do not intend: If a majority of stakers were to coordinate to prevent certain transactions from getting into blocks, it is proposed that a new fork of the protocol would be created implementing "social slashing", with the effect of punishing the misbehaving stakers and removing their power to corrupt the system.

2.2. Forking and composability

We have seen that subjectivocracy is a key building block available to Layer 1 systems which underpins their security. However, oracle and arbitration features are usually implemented not as part of the Layer 1 system on which they run, but instead by applications running on top. In Ethereum, there have been occasional proposals that the base layer should commit to, or *enshrine*, some additional real-world data. In 2015 Martin Köppelmann suggested that Ethereum could serve as a generalized Ultimate Oracle, and the entire Ethereum ledger forked in case of a sufficiently escalated dispute. Justin Drake made a proposal for enshrined price feeds. Most recently, Alex Gluchowski of Matter Labs advocated for the establishment of an Ethereum Supreme Court with Forking as the Court of Final Appeal.

However, neither proposal has gained traction as designers of Layer 1 systems usually prefer to limit the universe of things that the blockchain has to come to consensus about.

Forking individual applications without forking the entire base layer is complicated because of its effect on *composability*: To usefully fork an Ethereum contract, it is not enough to make a new copy of the contract in question and let users use the new version. You also need other contracts to know that they should talk to the new version, which would imply forking other contracts as well.

2.3. Escalation games and backstops

Systems with a need to establish some kind of external truth have often converged on the use of *escalation games* [4] to incentivize participants to report what they consider to be true.

These systems typically use an escalating system of competing bonds or stake: A user reports an initial result and puts up a bond. If this is unchallenged by other users, the system treats this as the final answer. However, other users can challenge their result by putting up a higher bond. Once the escalation has reached a predefined level, the system resolves the dispute using some other mechanism, such as a vote of token holders.

Such systems incentivize rational participants to act honestly as long as participants not participating in a particular attack have access to enough capital to escalate where necessary and, crucially, as long as the method of ultimate resolution can be expected to act honestly. Any participant putting up a bond behind a malicious response can expect to lose it, and any participant who sees a malicious response can expect to make a profit by challenging it.

Since the method of ultimate resolution does not normally need to be invoked, it can often have characteristics that would be unsuitable for everyday resolution; In particular it can be slow and socially expensive, requiring the attention of many participants. The ultimate resolution may be invoked occasionally, or it may act as a game-theoretical backstop which never needs to be invoked at all; The mere possibility of invoking it is enough to dissuade rational attackers.

3. Prior work

3.1. SchellingCoin

Vitalik Buterin discusses a design for a minimal-trust universal datafeed using a *schelling game*. Participants make reports in two stages, the first committing to a value and the second revealing it. They are rewarded or punished to the extent that their answers are in line with the answers provided by other participants. It is assumed that the most likely choice by other participants is the truth, so each individual participant will be motivated to report truthfully. This depends on the assumption that a majority of participants are not coordinating.

[4] A pure escalation game, with the method of ultimate resolution left open, is implemented by reality.eth.

3.2. Augur

Building on the Truthcoin design by Paul Sztorc, Augur created a prediction market on the Ethereum blockchain using a combined *schelling game* and *escalation game* backstopped by the ability to fork into two versions. The ability to fork protects holders of their native token, REP. This protection in turn protects the system against certain game-theoretical attacks on the schelling game mechanism such as the p-epsilon attack originally described by Andrew Miller and discussed by Vitalik Buterin.

However, since the tokens Augur users use to bet are tokens like ETH and DAI that it cannot cause to fork, it must make use of an *economic security* mechanism to decide which of the two forks is "correct". The mechanism the design chooses is to require holders of its governance token to move the tokens into one branch or the other, and measure which fork attracts the most tokens. The extent of its "subjectivocratic" protection is thus limited to holders of their native token: A REP holder is always free to choose the "honest" branch, and continue using the version of the system that reports honestly, even if they are in the minority, but ETH and DAI holders are at the mercy of the majority. This in turn means that their system cannot safely handle more value than a fraction of the total market cap of their governance token, and only economic, not subjectivocratic security, holds.

To ensure that the system does not exceed its *economic security bound*, the system keeps track of the value of outstanding bets, or *open interest*. If the value of the open interest becomes dangerously high in relation to its token, it automatically raises fees in the hope of increasing the token value, reducing the open interest, or both.

Augur is intended to be used as a self-contained system, with its oracle serving only Augur markets. The authors note other contracts using the system as a general-purpose oracle would undermine even its economic security as it would increase the amount secured without increasing revenue to token-holders; These are considered "parasite contracts", and the possibility that they will be deployed is left as an unsolved problem.

3.3. Kleros

Kleros uses a similar mechanism, but where the escalation game is represented as a hierarchy of courts, and appeal courts with the Kleros "General Court" at its apex. To become a juror in any of Kleros' courts, one needs to stake Kleros' native token PNK in that chosen court. When an arbitration is requested, the wallet addresses are randomly chosen as jurors, the more PNK tokens staked the higher the probability to be chosen. The jurors then participate in schelling games casting their votes. Unlike Augur, Kleros does not have the ability to fork built into its smart contracts. However, the ability to fork the system in case of an attack exists implicitly, and its developers have expressed the intention that future versions will facilitate forking the system if necessary as a defence against attacks on its governance process.

But as with Augur, forking would only be enough to save the honest tokenholders, not the smart contracts relying on their data, which would still

be pointing at the old version of the system in the event of a fork. As such, the forking mechanism is useful in deterring game-theoretical attacks within its economic security bound, but not in escaping the economic security bound.

3.4. UMA

UMA follows Augur in attempting to manage the value it controls and ensure it remains below its economic security bound, which they term the "Cost of Corruption". Unlike Augur, it is designed with the goal of serving external contracts. Instead of managing its open interest internally, it requests that contracts using it report their own open interest. It then adjusts its fees to attempt to keep open interest and token value in balance.

The fees involved in maintaining sufficient economic security may not be trivial; The authors of the UMA whitepaper surmise that in a steady state, sustaining $100 of open interest may incur $5 fees per year, as investors typically expect a 5% return on invested capital. They anticipate their system to be used for financial instruments where the notional value of a position is often much greater than the amount of actual open interest, in which case these numbers may seem less daunting; However, there are many use-cases for oracles and adjudication systems where this will not be the case. For example, if we imagine using such a system as a backstop to DAO governance, such fees would devour nearly 20% of the DAO's assets in just 4 years. As the authors note, the viability of such systems is at its highest early on, when the valuation of the token mainly comprises future revenue (after expected growth). This is highly advantageous when attempting to bootstrap the system, as users can be onboarded with minimal fees. However, this dynamic acts in the opposite direction as the system matures, particularly if we expect that at some point it will transition beyond the steady state considered by the paper and into a winding-down phase.

The UMA whitepaper also discusses approaches to handle what Augur called Parasite Contracts, and UMA terms "tax evaders". As the authors note, in a composable system like Ethereum, it is not possible to prevent users from reading data from a contract which claims to secure a small amount of open interest and using it to serve a contract which handles a large amount of open interest. An alternative the UMA whitepaper discusses is to "fuzz" its outputs, so that instead of simply reporting results, its reporters are instead tasked with making the ultimate payouts to users, and these payouts are obscured using interactive zk-proofs to prevent their use by parasites.

3.5. Amoveo

Amoveo, a blockchain for financial derivatives, implements a proof-or-work blockchain with an escalation game and an enshrined oracle: Facts about the world can be settled with an on-chain betting game, and it is proposed that in the event of a malicious resolution, the blockchain would be forked. To the extent that it represents assets managed on the blockchain, and users can be expected to follow even a chain with a minority of proof-of-work, Amoveo can be considered to have escaped the economic security bound.

In practice, implemented as a standalone blockchain, Amoveo has suffered from a lack of compatibility with other parts of the smart contract ecosystem.

3.6. Chainlink

Chainlink Whitepaper version 2 specifies a staking-based "first-tier incentive model" to incentivize the publishing of accurate data feeds, but the security of this model relies on a further "second-tier adjudication model" that can give reliable judgments to backstop their first-tier system. Several suggestions are made for possible “second-tier” systems, but they tend to rely on reputation rather than cryptoeconomics; For some situations, participants with strong reputations could participate in a multisig arrangement. For others, trusted hardware could be used, leveraging trust placed on the hardware manufacturer. We propose that a similar design could leverage our adjudication layer as the “second-tier” system, removing the need for reliance on reputation.

4. Applying a subjectivocratic mechanism to a layer 2 ledger

We saw in the discussion of subjectivocracy that the problem of making an adjudication model that is not subject to an economic security bound is already implicitly addressed by the base layers, which benefit from subjectivocracy for their own security and governance. However, the way we have layered applications on top of the base layers has prevented the same mechanism from being practical on the application layer. A single contract can be forked either by its own code or purely in social space, i.e. by creating a new contract and using that instead of the original one. But individual contracts are not used in isolation: when a contract forks, any other contract that expects to be able to call it also needs to be forked, or have some other way to determine which fork to use. It is therefore necessary to be able to fork not just a single contract, but an entire interacting ecosystem of contracts.

We propose to enable such an interacting ecosystem of contracts by creating a Layer 2 system which can be forked. A forkable L2 ledger can be anchored into a L1 system via rollup contracts. The forking process can then be initiated on L1 by forking the anchor rollup contract of L2 into two. This allows an entire ledger to be efficiently cloned into two parallel versions with the same initial state. Since the different chains can communicate trustlessly with each other and the base ledger, it is also possible to objectively measure the relative prices of tokens on each ledger.

This mechanism allows us to build an L2 with a robust, enshrined backstop adjudication system. In extreme cases, arbitration or oracle inputs can be challenged and escalated to such a point where the forking is initiated, thereby resolving the dispute. The mere existence of this backstop mechanism makes it easier to design and secure oracle and adjudication protocols.

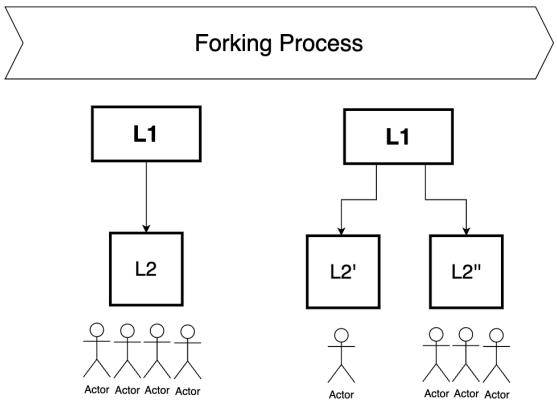

Figure 1: **Forking Process:** Forks of applications are enabled by forking the whole L2 on which they are deployed. Long-term the users will then use their preferred fork: The one that - in their opinion - has the truthful oracle input or correction conflict resolution process/outcome

4.1. Forkable L2 specification

Our proposed system, which we will refer to here as the "Forkonomic System", is based on a zk-rollup Layer 2 system, such as Polygon zkEVM or zkSync. A contract on a Layer 1 ledger, such as Ethereum, manages a state root, representing the state of a separate Layer 2 ledger. Changes to the state of the Layer 2 ledger are submitted in batches to the Layer 1 contract, and a zk-proof is submitted for each batch allowing the Layer 1 contract to verify that the resulting state root correctly reflects a valid set of changes to the state of the Layer 2 ledger. Messages can be sent from contracts on Layer 1 and relayed trustlessly to Layer 2, and vice versa.

The Forkonomic System makes the following modifications to a stock L2 system to allow it to serve as an ultimate backstop for adjudication that can serve oracles and governance processes:

1. The Forkonomic System has its own governance token, represented by a native token of the Layer 2 ledger, the "Forkonomic Token". This token can also be used for gas payments on the Layer 2.
2. The Forkonomic System enshrines an approved list of Adjudicators, such as oracles or dispute resolution services, as part of a contract we call an Adjudication Framework.
3. In the event of a dispute about the operation and performance of an Adjudicator its Adjudication Framework uses an escalation game such as reality.eth to consider removing an Adjudicator from the approved list.
4. Where the dispute is not resolved, at the cost of a certain proportion of governance tokens, anyone can cause the system to fork into two parallel ledgers, one enshrining the original list of approved Adjudicators for its

requesting Adjudication Framework, the other one without the disputed Adjudicator.

5. The governance tokens spent to trigger the fork are used to subsidize an auction or similar price-discovery mechanism on Layer 1 trading governance tokens on one of the new forks against each other.
6. Contracts involving sending messages to and from the Layer 1 ledger are modified to handle the possible existence of multiple forks, and where necessary choose between them. The decision process on which fork to choose may resort to the result of the price-discovery process mentioned in the previous step

4.2. Adjudicators and Adjudication Frameworks

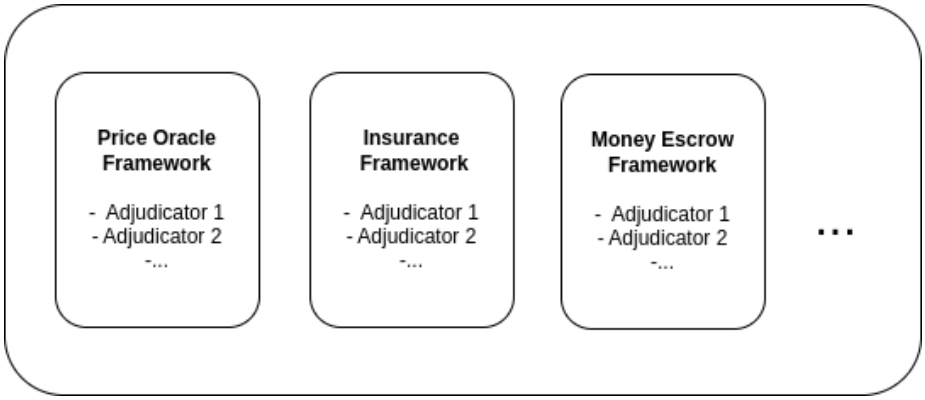

Figure 2: **Adjudication Frameworks and Adjudicators:** Are embedded into Adjudication Frameworks that define rules for them

An *Adjudicator* is a contract on the Layer 2 ledger that can adjudicate a dispute. The dispute may for example concern the addition of a price feed to a price oracle service, a specific data point reported by a price oracle service, a report of factual data, or whether a governance decision was in accordance with a predefined constitution.

Adjudicators are managed as part of an *Adjudication Framework*. An Adjudication Framework specifies a set of properties that its member Adjudicators must have or abide by. These may include specific processes and requirements, which can be tailored to different use-cases. The Adjudication Framework may also define additional requirements, such as confidentiality of information used in proceedings.

See *Designing adjudication processes* for more discussion of the role and principles underpinning the design of Adjudicators and Adjudication Frameworks.

Adjudication Frameworks define the rules for adding and removing Adjudicators. The Adjudication Frameworks implement a permissionless removal: Anyone can challenge an Adjudicator's membership of any given Adjudication Framework at any time, at the cost of a minimum bond which they will lose if

the challenge is unsuccessful. During the challenge, the Adjudication process continues as usual unless the bonds have reached a sufficiently high threshold as specified by the Adjudication Framework.

We envisage a challenge to be decided by an escalation game, conducted on Layer 1. Where the procedure does not resolve the dispute, its ultimate adjudication consists of a fork of the ledger, resulting in one version of the ledger for which the disputed Adjudicator is still part of the Adjudication Framework in question, and another in which it has been removed.

4.3. The dispute process with forking as a backstop

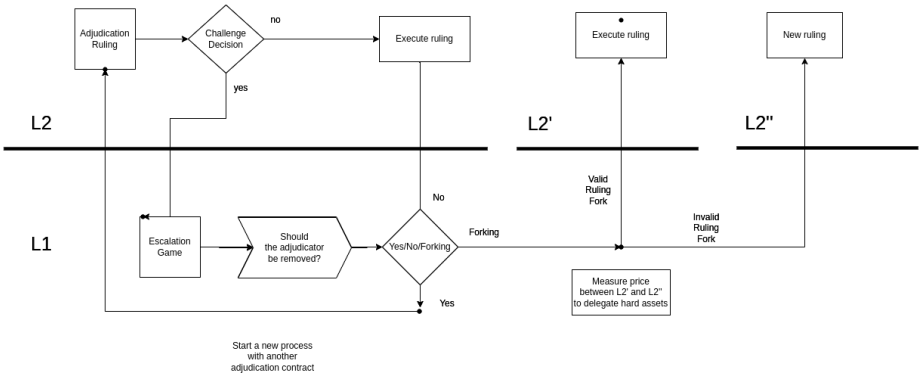

Figure 3: **The adjudication process with forking as a backstop** An Adjudicator's ruling on L2 can be challenged in L1. If there is no unanimous decision in the challenge process, then the chain splits into two branches L2' and L2"

4.3.1. An initial question or query

Each claim starts with some initial query, usually by a contract on the Layer 2 system.

Although it is technically possible for a question to be settled directly by an Adjudication Framework, it is anticipated that there will be an additional process prior to this, with the Adjudication Framework being used only for dispute resolution.

For example, a contract needing to settle an insurance claim on an office burning down might create a question using reality.eth, with an Adjudication Framework specializing in insurance acting as the arbitrator. A data feed protocol may have its own system for allowing data providers to send data to its oracle contract where it can be read by consumers, along with a process to challenge bad data providers, and use one of our Adjudication Frameworks for sufficiently escalated challenges.

4.3.2. Escalation to an Adjudicator

In the case of a dispute - in reality.eth this usually involves two parties putting up increasingly high bonds until one pays a fee for arbitration - the issue is sent to an Adjudication Framework.

The Adjudication Framework delegates it in turn to one of its Adjudicators. The method by which an Adjudicator is chosen is defined by the Adjudication Framework. For instance, it may allow anyone to assign the dispute to an Adjudicator on a first-come, first-served basis, or prioritize Adjudicators according to an order specified by the user.

4.3.3. Ruling by an Adjudicator

Using its own processes, the Adjudicator makes a ruling and sends it to the Adjudication Framework. It is then locked for a certain period (again defined by the Adjudication Framework) to allow for a challenge. If there is no challenge, the decision is sent back to the querying contract and the query is complete.

The nature of possible challenges should be specified by the Adjudication Framework. To illustrate, a challenge may be procedural - meaning that the challenge claims that the procedures of the output framework were not correctly followed. It may be substantive/material - meaning that the adjudicator's decision is itself not correct. Or it may be a combination of both.

The specific procedure for the challenge is defined by the Adjudication Framework. One implementation would be to do a further escalation game using reality.eth, this time on whether the Adjudicator should be removed. To raise this challenge, all information that was available to the Adjudicator needs to be published on a challenge platform. In the case of an Adjudication Framework or Adjudicator using confidential proceedings, this requirement may reveal previously private information. Accordingly, raising the challenge would have to come with a significant cost that compensates for costs associated with a disclosure.

4.3.4. Handling the result of a challenge

A challenge can be resolved either by its own escalation game (for example, if an answer to the reality.eth game is left unchallenged), or by ending up on one side or the other of a fork (see below).

If the challenge fails, the answer given by the Adjudicator will stand and can be sent to the querying contract.

If the challenge succeeds, the Adjudicator is removed from the Adjudication Framework and the original question is instead sent to a different Adjudicator, using the procedure specified by the Adjudication Framework. If the Adjudication Framework was assigning disputes on a first-come, first-served basis, any other Adjudicator on the list would then be able to accept it.

4.3.5. Escalating a challenge to an Adjudicator

If the stakes in the escalation game continue to rise, one side may opt to pay a fee for the Forkonomic System to resolve the case by forking. The Forkonomic

System charges x% [5] Part of this fee will be burned, and part of it will be used to provide a subsidy for the price-finding auction (see below).

This escalation payment gives notice that the system will fork in a defined period, for example 1 week. The fee is sent to the *Price-Finding Auction*, which remains open until the last block before the fork, establishing the *Fork-Weight Price*.

Once the price-finding period ends, the anchor contract representing the L2 system will be duplicated into two separate contracts, each representing a new chain with a new Chain ID. Each new L2 system's initial state root will be identical to the old one. The bridge contracts on L1 will also fork into two bridges for each system. Users must update the Chain IDs in their wallets to continue interacting with their preferred chain.

The root contract of each ledger on L1 will notify the Adjudication Framework of the result of its dispute on that particular ledger: On one fork the Adjudicator will be removed, and on the other, it will be considered to be preserved.

During the entire outlined process, the bridges will remain active and should operate normally.

The last-resort forking mechanism ensures that the system is not easily attacked by a malicious party. If the system was used only by rational self-interested participants, we would expect that it would never escalate to a forking point: The mere existence of the forking mechanism should be enough to incentivize users not to escalate propositions that would put them on the losing side of a fork; it acts like a backstop. In practice, forks are not expected to occur frequently.

4.4. Designing adjudication processes

The forking process is socially expensive, as all its users need to make a conscious decision about choosing a fork they prefer. Hence, one cannot use subjectivocracy regularly, but rather only as a backstop mechanism. Hence, before considering subjectivocracy, all disputes need to go through an adjudication process. The Forkonomic System enforces this by only accepting forking requests from approved Adjudication Frameworks.

As different types of disputes require different approaches to dispute resolution and many paths can achieve the same outcome, the Forkonomic System adopts a modular and semi-permissionless approach to adjudication. Any EVM-compatible oracle or dispute resolution system can be integrated as an Adjudicator within one of the Adjudication Frameworks. Any dispute resolution or oracle system can use Adjudicators as an appeal authority at their choosing. Allow-listed Adjudication Frameworks can be added through stakeholder governance.[6]

[5] The authors expect the adjudication fee to be between 0.2% and 5% of the total token supply.

[6] e.g. Were Ethereum to establish an Ethereum Supreme Court, or similar dispute resolution framework, stakeholders in the Forkonomic System could decide to add it as an Adjudication Framework.

The structure of any Adjudication Framework is to ensure that cases can be dealt with fairness, efficiency, and finality, while the forking process ensures error correction.

We envisage that the protocol will accept Adjudication Frameworks addressing different categories of disputes, e.g. one focusing on oracles for on-chain price data, and another addressing insurance sector disputes. Accordingly, depending on the nature of claims adjudicated under that particular framework, the frameworks may enforce differing criteria (e.g. decision formats, escalation time periods).

Depending on the subject matter, Adjudicators may use a range of mechanisms including crypto-economic games, AI systems, and traditional panels of domain experts and dispute resolution practitioners more akin to arbitration tribunals or courts in a traditional legal system.

4.4.1. Principles and values of the adjudication process

Although the system is premised on permissionlessness, to preserve the functioning of the Forkonomic System, we propose that all Adjudication Frameworks require adherence by Adjudicators to the following minimum standards:

1. Discoverablity of justification: Any decision should be the outcome of clear processes that enable users to assess the court decisions' legitimacy as easily as possible. This is important for the Forkonomic System because, during a forking process, the users are the ultimate decision-makers of the legitimacy of the Adjudicators by choosing one fork over another.
   We propose that this can be achieved by Adjudication Frameworks having to lay down the requirements for how a decision needs to be presented and justified in a standardized format depending on the subject matter of the Adjudication Framework's decisions/feeds, and the Adjudication Framework's requirements[7]. This helps users to assess an Adjudicator's decision in case of a forking event. For example, an Adjudication Framework focused on data feeds will not require an adjudicator to issue a detailed reasoned decision. However, the method used to get the data feed, including computation, needs to be documented and transparently executed. Whereas, for example, in an insurance disputes-themed Adjudication Framework, the decisions would need to succinctly outline reasons in text form. Moreover, where reasoned decisions are issued, an Adjudication Framework may lay down that a party can request clarification of a decision within a specified period following its issuance, with a clarification from the Adjudicator required within a similarly specified time period. To maximize for discoverability of justification at the outset of a forking process, the authors could also envisage the inclusion of an advisory council that ensures that a dispute's material is readily accessible to users.
2. Compliance by an Adjudicator with its own plain terms that users must have consented to: We consider that, as a minimum, the Adjudicator's

[7] e.g. time limits, language requirements, succinctness

process must be in compliance with its own terms and the terms of the Adjudication Framework. Both sets of terms must be plainly made known and agreed to by its users.[8] The form of consent depends on several factors, including the character of adjudication, but may, for example, be achieved through an arbitration clause/agreement to arbitrate before the Adjudicator in the terms/contract or by a submission agreement to arbitration.

It is hoped that this paper will spark a lively debate on applicable standards to Adjudication Frameworks and its Adjudicators. The Protocol's stakeholders are expected to decide on the exact criteria for Adjudication Frameworks and their allow-listing. Moreover, the exact mechanism of distinct Adjudication Frameworks is expected to evolve over time.

4.5. The price-finding auction

Although the decision over which fork to use is a subjective one, and users can use whichever fork they prefer, it is possible to establish objective facts about what choices users make and which forks carry the most economic value. This information is useful for smart contracts on Layer 1: A smart contract that is fully automated and deployed on a blockchain without the subjectivocratic properties for which the Layer 2 is designed cannot make its own subjective judgment about which chain is better. This information is of particular use to bridges and stablecoins (see the following sections).

To help measure these objective facts, an auction is run in the period between requesting the fork and the fork taking place. Any user holding Forkonomic Tokens can deposit their tokens in a contract trading tokens on one side of the fork for tokens on the other. Users specify the ratio they consider appropriate. For example, if you believe that a token on the "Challenge Succeeded" side is worth 10x a token on the "Challenge Failed" side, you would set your price at 90%. After the fork the smart contract will hold tokens from both sides. It will calculate the clearing price at which all tokens will be distributed to users at their requested price or better. If the clearing price is 95%, a user bidding 96% or higher will be paid 20x their original deposit, on the "Challenge Failed" side. If the clearing price is 50%, the same user would be paid 2x their original deposit on the "Challenge Succeeded" side. Users who bid exactly the clearing price may receive their payout in a mixture of tokens from both sides.

To incentivize participation in the auction, part of the fee for triggering the fork is distributed among participants in the auction, in proportion to their

[8] e.g. Compliance by an Adjudicator with its own transparent terms is a precondition to fairness but generally not considered sufficient to ensure fairness. We consider fairness, which usually implies a highly desirable and generally necessary feature of any adjudication method, but do not generally prescribe it here. We do expect that many Adjudication Frameworks will prescribe fairness, including neutrality and impartiality in terms of jurors/decision makers, procedure and rules. Other standards may include: confidentiality, expertise, ethical conduct, commitment, efficiency, as well as code Feasibility, recognition, and legacy system enforceability.

deposits. So for example if the fee was 1 million Forkonomic Tokens and 2 million Forkonomic Tokens were deposited in the auction, each token paid out will be matched by an additional 0.5 tokens.

5. Bridging Assets

Since the system is an L2, messages can be sent trustlessly between the Layer 1 ledger and the Layer 2 ledger. Like any other Layer 2 ledger, this can be used for bridging assets. However, the forking process has consequences for the way assets can be bridged.

5.1. Forkable assets

Some tokens, most commonly utility and governance tokens, can be issued on the forkable ledger and benefit from subjectivocracy in its purest form. These tokens are automatically duplicated on both forked ledgers, and users decide for themselves which they wish to ascribe value to.

Bridges handling such tokens on Layer 1 can split into two new contracts, and allow users to redeem tokens in the parent contract by crediting them on each of the child token contracts.

5.2. Unforkable assets

Some tokens represent a claim on something beyond the forkable ledger. Here this applies to any asset managed on the base layer of the blockchain or some other blockchain system which the forkable ledger does not have the power to fork. It applies in a different way to "real-world assets" which represent a claim on something outside any blockchain ledger, such as a Euro in a bank account or a physical house.

5.2.1. Unforkable assets redeemed by third-parties

For some assets, the chain on which these assets are considered valid will be managed by some third-party. For example, if a stablecoin issuer issues tokens on a forkable ledger, they will decide at their own discretion on which of the resulting chains they would redeem. The holder of such an asset must trust the issuer to make a reasonable choice of chain and not choose maliciously, but this merely extends the scope of the existing trust assumption that the user has in the issuer.

The same trust assumptions can apply to an asset originally issued on a non-forkable ledger, but transferred to the forkable ledger via a forkable bridge. A bridge contract can be deployed which allows the issuer to choose which side should be honored in the event of a fork. In the event that the issuer declines to choose, such a bridge may fall back on other methods such as an automated fork-choice rule (see below). Alternatively, an automated fork-choice rule may be used as the default, with the issuer reserving the right to override it.

5.2.2. Unforkable assets managed by automated fork-choice contracts

For some assets, the process of choosing between the forks can be automated: Bridges to the L2 can be built so that the unforkable assets are transferred to the control of one fork after a bifurcation has been initiated.

One possible implementation is to measure the forks' token prices on L1 and then delegate the hard assets to the fork with the higher prices. This automation mechanism relies on economic, not purely subjectivocratic security: A malicious actor prepared to bid on the "dishonest" branch beyond the value of the tokens on that branch could make them appear more valuable. This would cause L1 assets whose destination depended on the adjudication proposition at the root of the contention to be settled according to the "bad" decision, not the "good" one. The cost of such an attack increases if tokens with high token values are issued on the forkable ledger, and their combined value is used to measure the adoption of each branch.

Note that the attack described here would not allow the attacker to steal assets that were unaffected by the decision in question; For example, ETH bridged to the L2 and kept in the user's own wallet would belong to the same user on both branches, and their owner could withdraw them regardless of which fork the automated bridge chose as the "winner".

6. Handling delayed adjudication

Since adjudication processes require human action and often discussion, they may take days or weeks to complete.

6.1. Making the querying contract wait

Some contracts can simply wait for a response. When using an escalation game such as reality.eth, the typical case will be uncontested, and therefore resolve quickly, and only occasional escalated cases will need longer periods of time to resolve. In many cases, for example insurance payouts, it is acceptable for users to wait for resolution and a payout. In other cases, users can trade their exposure. For example, in a prediction market, shares in different outcomes can simply continue trading until resolution. If it is clear which way adjudication will go, shares on the winning outcome should trade at close to 100

6.2. Freezing contracts on payment of a bond

Some contracts need action to be taken quickly to prevent harm. For example, a contract that manages the upgrading of another contract in the event that a security vulnerability is found cannot leave the contract it manages operating normally pending the adjudication process: Funds would be irreversibly drained before the process was complete. It needs instead to freeze the functionality of the contract, either stopping it completely or freezing irreversible parts like withdrawals. The freeze is triggered when a user who thinks the upgrade needs to occur pays a bond. If the adjudication process confirms the need for the upgrade, the bond is returned. If it finds that the request for an upgrade was spurious, the bond is forfeited.

6.3. Fast forking

In some cases, contracts need to act immediately and cannot wait for slow resolution processes. For example, a stablecoin may need price information to control liquidations, and must liquidate individual under-collateralized positions quickly in case the situation deteriorates further and the entire system becomes under-collateralized. In such conditions a variant of the system can be deployed as a further layer, creating an additional forkable subset of the ledger. This layer can be forked immediately on payment of a bond, without waiting for adjudication. Assets locked in bridges between the main forkable layer and the new layer can be frozen until adjudication is complete and the bridges receive a message telling them which fork they should follow. Fast forking allows the operation of the "correct" ledger to continue uninterrupted, at the cost of requiring users to be alert to the possibility of a fork.

7. Reversibility

Social consensus has sometimes been used to make arbitrary post-hoc changes to the state of a blockchain. For example, in 2016 Ethereum was forked to restore assets stolen as the result of a contract bug by creating a new version which included an irregular state change. Such changes are not a feature of this system. Forking and social consensus only affect a contract to the extent that it includes code which requests that the system's enshrined adjudication process be used to make a judgment. If a contract does not request data supplied by an Adjudication Framework, which in turn may request a fork of the system, a fork will not affect it, except to the extent that the assets it represents may now have a twin on a fork which was created for some unrelated reason.

There have previously been proposals to incorporate reversibility in either base protocols or token standards. EIP867 proposed a generic template for non-standard state transitions to be used to restore assets lost by bugs and hacks. by Wang, Wang and Boneh proposed a type of new token with the ability to freeze tokens pending governance decisions, and potentially restore ownership of the tokens to a previous owner. The design of such systems raises complicated questions about under what circumstances reversion should be made, and the optimal solution is likely to depend on the use-case. This in turn creates complications in how the reversible part of the system should interact with non-reversible parts, or parts that revert based on different governance methods and timescales.

The *fast-forking* mechanism described in the previous section, in combination with the system's enshrined adjudication backstop, provides the necessary pre-conditions to create a ledger which enables reversibility. A ledger on top of the system's layer 2 has a state root which can be revised. A governance proposal can specify a change which can be applied to create a new state root. This may take the form of a *containment fork* reverting to the state directly before the fork and temporarily freezing the affected contracts, or a *resolution fork* restoring asset ownership or setting other contract data to the state proposed by

governance. A delay in moving assets out of bridges, which can be extended when in a forking state, allows the fast-forking ledger to conduct reversals without affecting the underlying ledger. The underlying ledger's enshrined adjudication can provide a backstop for the mechanism judging which of the forks should be used for bridged assets. The underlying ledger is only itself forked in the case of an escalated dispute. The system is well-suited to meeting the challenges of handling interactions between reversible subsystems.

Further discussion about these constructions can be found at https://ethresear.ch/t/recovering-from-smart-contract-hacks-forkable-reversible-roll-ups/16781

8. Stablecoins

In this section, we propose a design intended to mitigate the problems caused by potential forks. Many of these considerations are also applicable to assets deployed on the Layer 1 ledger as we can never rule out the possibility that a Layer 1 ledger will undergo a contentious fork.

There are two fundamental challenges for building a stablecoin in an environment where there may be a contentious fork:

1. Collateral prices can drop quickly, especially in case of a 50% / 50% split.
2. An outstanding dollar cannot result in two outstanding dollars on two forks, as otherwise, everyone would like to get dollars shortly before a fork.

8.1. The proposed mechanism:

The stablecoin - we will call it FAI (forkonomic DAI or RAI) - will work quite similarly to RAI or LUSD: The stable tokens (FAI) are created by opening Collateralized Debt Positions (CDPs) with the Forkonomic Token as collateral. The collateralization factor may need to be higher than for existing Ether CDPs, as the price of Forkonomic Tokens is expected to be more volatile overall and particularly in the case of forks.

We saw earlier that before the Layer 2 ledger forks, it will trigger a subsidized auction establishing a market price for the Forknonomic Token on one chain relative to the Forknomic Token on the other chain. Immediately after a fork, each version of the stablecoin contract will rebase the FAI value based on the percentage price ratio of the Forkonomic Token of the fork. For example, if the value split between the branches is 30% to 70%, on the 30% branch, the FAI will be re-based to hold only 30% of its value, and the rest (70%) will be held on the other branch. If the overall market cap of the Forkonomic Token does not change during the forking process, all CDPs will be collateralized as they were before the fork.

Each FAI holder then has the choice to push their split FAI value again into one chain by trading FAI$\hat{1}$ of the first branch against FAI$\hat{2}$ of the second branch or the reverse trade. Therefore, the promise to each user to hold roughly $1 remains valid during the forking process. However, there is an edge case: if a user has locked up their FAI in a long-term contract and cannot exchange their FAI from a losing branch, it is possible that the customer may effectively lose

money: If a branch is initially attributed some value - let's say 5% - after a forking process, and if later that branch is abandoned, causing the Forkonomic Token of that branch to be worth 0, then the CDPs can no longer remain active and the FAI will be liquidated. This would result in a 5% loss of value for the user.

The reason is that the original value of FAI was rebased by 5% during the forking process, but this 5% of the value will be lost once the claim is available to the user, as all CDPs would be liquidated.

Given that in most cases, the value of the FAI token is preserved during forking [9] and forking is expected to happen rarely, FAI will provide most benefits of a non-forkable stablecoin. The shortcomings are mostly about the rebasing of value and its implications.

9. Conclusion

The system proposed in this paper has the potential to overcome the limitations inherent in dispute resolution and oracle protocols that rely solely on economic security. It provides a platform on which applications can be deployed with little or no alteration, and easily benefit from subjectivocratic security in a way that has previously only applied to base layers.

The system retains compatibility with the Layer 1 on which it is deployed, and can trustlessly obtain information about the state of the Layer 1 ledger or other Layer 2s deployed on the Layer 1 ledger. Unforkable assets from the Layer 1 ledger can be bridged trustlessly to the Layer 2 ledger, albeit with an economic security bound concerning decisions over which the Layer 2 ledger may fork.

The greater probability that the system will fork compared to the Layer 1 ledger on which it is deployed poses challenges for stablecoin designs, but these problems can be mitigated with the fork-aware designs outlined in this paper.

Free from an economic security bound, the system provides a robust backstop which can strengthen the existing designs of oracles and governance systems and enables new experiments. This in turn allows the creation of more secure Defi systems, stronger protocols for insurance and prediction markets, and better governance for decentralized organizations.

[9] If the fork splits more equally - say 40%/60% - then the potential loss is the highest, but also the likelihood for one chain to lose all its value entirely is the smallest.