hybrix: A Peer-to-Peer Token Protocol Across Multiple Ledger Systems

Joachim de Koning, Rouke Pouw

20.02.2020
version 1.6

Abstract

A meta-protocol and blockchain integration solution in the form of a decentralized asset for transacting value across different digital currency systems would allow for ledger and blockchain platforms to be used as a value transfer medium without needing any financial intermediaries. Third party services currently assist users to exchange one form of digital cash or asset for another, but a trusted third party is still required to mediate these transactions. Decentralized exchanges mostly operate on a single chain or else solely between compatible ledgers or blockchains. We propose a solution to the problem of these isolated digital currency systems using a meta-level transfer protocol with an extendable and modular design, making accessible any kind of ledger-based economy or other digital cash system for cross-blockchain and inter-systemic transactions. Many ledger systems have their own form of consensus mechanism. Instead of inventing yet another network infrastructure, the *hybrix* protocol will enable using the data layer of the underlying infrastructure in conjunction with its consensus mechanism. This makes it possible for instance to 'hijack' ledger systems and use their combined powers to create a truly cross-ledger asset, with all the benefits of the underlying ecosystems. The underlying ecosystems will also benefit from hosting this protocol. Every *hybrix* protocol transaction yields profit to these respective ecosystems by paying transaction fees to their network supporting miners and stakers.

1 Introduction

Before the inception of Bitcoin, commerce on the Internet had come to rely almost exclusively on financial institutions serving as trusted third parties to process electronic payments. Since the start of the Bitcoin experiment, online commerce has started to change, since cryptocurrency payments incur lower fees, and peer-to-peer payments open up online commerce to people in developing countries [1]. However, large institutions are positioning themselves into finance, presenting services that make the use of digital currency easier. While touting decentralization, in reality these organizations are trying to control the flow of capital across the Internet, and across the world.[2] Technically Bitcoin earlier on had solved some of the problems of the reversibility of transactions and trust issues that plagued online commerce, however, new players in the arena are offering replacements for Bitcoin's peer-to-peer payment solution. Some of these replacements put buyers and sellers at risk of having their financial interactions being controlled, data harvested and monetized.[3] Other decentralized asset technologies have been developed parallel to Bitcoin, and out of it has grown an entire ecosystem of decentralized finance. Strides have been made in experimental and practical forms of decentralized finance and autonomous contractual governance.[4] However, regardless of all these constructive developments, one can observe large institutions like R3, IBM, Ripple and the Libra Association attempting to inject their influence back into online commerce, by creating pseudo-trustless and permissioned solutions for the exchange of value and presenting these as secure and trustless to the public.[5] Simultaneously, solutions proposed by these organizations aim to create profitable top-down enterprises, in which the user becomes the customer and/or product of new financial technology gatekeepers. What is needed is a protocol for value transfer between distributed ledger systems that is open by design, and not controlled by a centralized party. This protocol would provide users for a way to move units of account across distributed ledgers in a truly decentralized way, opening up the ability to exchange value on ledger (e.g. on-chain) back to usable digital currency or token. This enables users to transfer value between ledger systems trustlessly, and provides an alternative to centralized exchangers. In this paper, we propose a solution to the value transfer problem across different distributed ledgers by using a meta-protocol for value distribution across multiple decentralized ledger systems. This meta-protocol utilizes the transaction attachment or data field of the underlying blockchain and these data blocks contain data about the represented value token and computational proof of the chronological order of transactions.

2 Overview

Our proposal is to create a protocol - called *hybrix protocol* - as a cross-ledger colored coin, making it technically borderless and not bound to a single ledger system. The term "Colored Coins" loosely describes a class of methods for representing and managing real world assets on top of the Bitcoin blockchain.[6] However, in our case we redefine this term to describe a token that utilizes any distributed digital ledger as its underlying

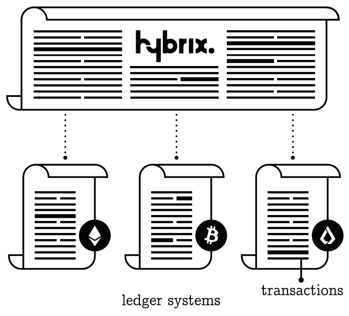

Figure 1: hybrix meta ledger

infrastructure. In developing this protocol it would be beneficial to users that the resulting technology is:

- open by design;
- not controlled by a centralized party;
- enables any user to transfer value between ledger systems;
- provides the possibility for users to issue tokens.

Challenges to a successful implementation of the protocol are:

- proper validation to avoid double spends;
- sybil attacks on the validator network;
- token squatting and index spamming;
- 51% attack on a single chain.

2.1 Framework

The *hybrix* protocol is a second-level token protocol that can transact units of account on a single ledger, or over multiple ledger (e.g. blockchain) systems. Its transactions are stored in a data block inside the attachment section of a zero-value transaction on any distributed ledger system. This gives its users the advantage of moving units of account, ultimately value, to any ledger system or blockchain that best suits their needs. Transactions containing meta data pay the usual fees denominated in the base currency of the ledger to miners, forgers, or stakers in order to register the meta transactions in the blockchain. This means that the assets using the *hybrix* protocol (HRC1 tokens) benefits from having a trusted and secure mining, forging or staking network without the need to re-create its own or use additional resources. Bitcoin, Ethereum and other cryptocurrencies have advanced features (such as scripting and smart contracts) which enable many users to create complex financial solutions. The consensus of these solutions and the manipulation of value is, however, confined to the ledger on which they are implemented. There are some cases where technologies (like atomic transactions) make it possible to connect distributed ledgers. Yet many of these solutions need specific implementations and compatibilities that not all ledger systems have on offer, and thus are limited in scope. We solve this problem by denominating a common protocol that works on any ledger.

2.2 Assumptions

The *hybrix* protocol uses no advanced features of any specific ledger system. This avoids it becoming overly dependent on any single distributed ledger system. Many systems are a moving target when it comes to development of cutting edge technology, and future support for their advanced options is in no way guaranteed. For the *hybrix* protocol to work, we assume the following requirements to be available on all ledgers. Ledger requirements:

1. immutability of past transactions
2. verifiable signing and authentication of transactions
3. all transactions must be publicly available (in order to verify the transaction chain)
4. transactions have
   (a) a unique transaction id
   (b) an attachment field for storing data
   (c) source and target address(es)

2.3 Definitions

The definitions below are ordered according to dependency on previous definitions.

intersystemic transaction A transaction occurring between two distinct ledger systems.

entanglement Informational connection between two transactions on separate ledger systems, that functionally relate them as a cross-ledger transaction.

token recipe A data object containing the necessary variables specifying the *hybrix* token identifier, and economic rule set.

consensus mechanism The way in which computer systems agree on the state of a system - in our case a distributed ledger and/or blockchain.

validator Network actor that analyses past transactions and makes available the legitimacy of these transactions according to the rules of the system protocol.

VAAS Validation as a service is done by network participants that offer to verify a user balance for a fee.

double spend A transaction that illegitimately increases the money supply in a ledger system.

genesis hash A hash calculated by using the recipe data containing a *hybrix* procotol HRC1 token definition.

attachment The data included with a transaction, sometimes called *message* or - in the case of Bitcoin and its derived coins - OP_RETURN. Primarily used on most ledger systems for annotation of the transaction. In the *hybrix* protocol this storage space is used to store protocol data.

OP_RETURN An Bitcoin script opcode used to mark a transaction output as invalid. This makes it possible to store data attached to the output in the Bitcoin blockchain. Since any outputs with OP_RETURN are provably unspendable, OP_RETURN outputs can also be used to burn bitcoins.[7] We hereafter refer to the data field of such a transaction in this whitepaper as *attachment*.

directed acyclic graph In graph theory a directed acyclic graph is a finite directed graph with no directed cycles. Blockchain is an example of a DAG with no parallel branches. A directed graph is acyclic if and only if it has a topological ordering.

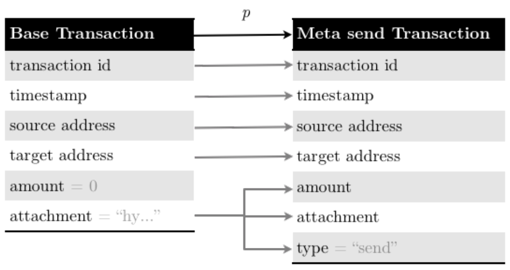

Figure 2: The parsing function p parses the attachment of the base transaction into the required fields.

3 Intersystemic Transactions

3.1 Structured Data on a Ledger

We define an electronic intersystemic token as a block of structured data that is inserted into the attachment section of a zero-value transaction on a distributed ledger (e.g. blockchain) system. This first layer ledger will serve as the base for a meta ledger that can span multiple different base ledgers. The transaction may, or may not, require a fee amount for using the underlying infrastructure of the base layer while the meta layer will require no extra fees. The content of the attachment of transaction on a base ledger can be parsed into a second layer transaction of the meta ledger. A parsing function p will extract the required meta transaction details from the base transactions attachment as well as using details from the base transaction that are still relevant (e.g. time, origin and target of the transaction). Token ownership is secured by the underlying ledger system every time a transaction is done. Each owner transfers their zero-value transaction containing the token data to another owner by digitally signing a hash of the previous transaction and the current transaction. Only a user that controls a ledger address has the ability to spend *hybrix* tokens on it. A payee can verify the signatures to verify the chain of ownership.

3.2 Creating and Spending a Token

The *hybrix* protocol is designed to function over multiple ledgers. As seen in the figure 3, the practical inception of a *hybrix* token occurs when a recipe containing the identifier and economic rule set is signed using the issuer's private key ❶, and a verification hash is stored on a public ledger ❷. After that, the only thing that is added to the recipe is the ledger symbol, and transaction hash of where the verification hash can be found. This is not limited to a single ledger, and can be done for multiple ledgers simultaneously if desired. Subsequently the token is minted on the same (or on another) address using a followup transaction ❸. This transaction contains a reference to the genesis transaction that preceded it on the same ledger system, and the amount of units minted must adhere to the emission rules specified in the rule set. Now that the HRC1 token has been initialized - in our example on *BCH* - it may be transacted on the same ledger using a simple send transaction ❹. This enables sending the tokens to other addresses on the same ledger system.

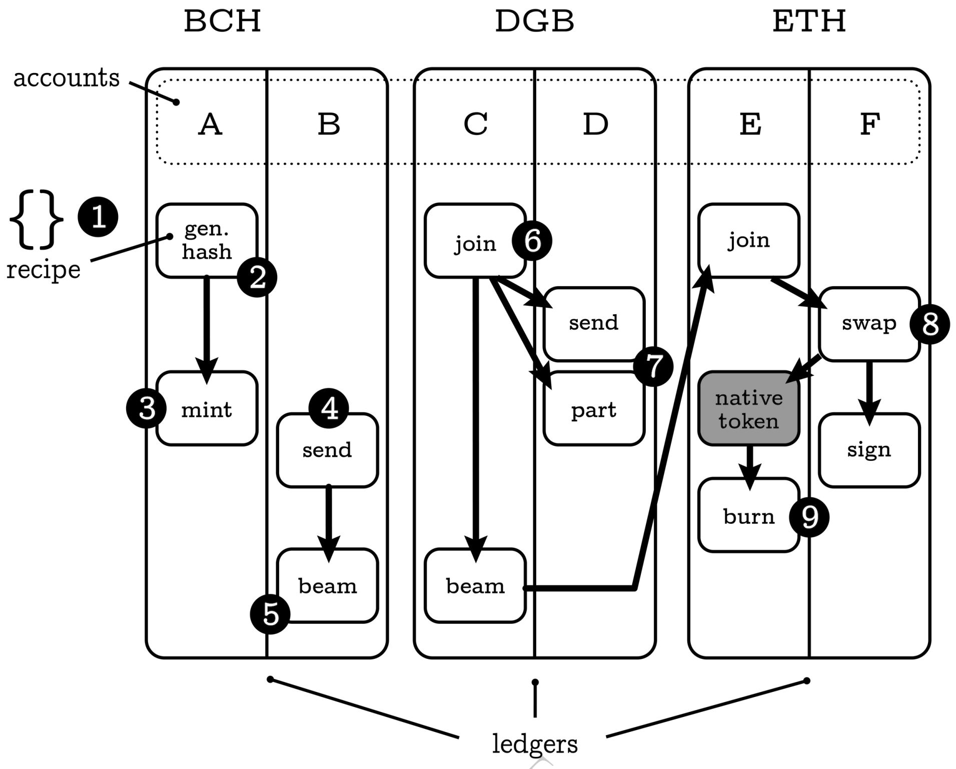

Figure 3: cross-ledger entangled transactions

3.3 Other Types of Transactions

When a transaction contains more data than a ledger system can handle in its attachment storage space, the transaction may be split up, and sent using a transaction accompanied by tailing part transactions that complete the contents of the entire operation ❼. Using a swap transaction, HRC1 tokens can be switched for native tokens that can then be used on the chain for trading, or interacting with smart contracts ❽. A swap transaction is legitimate when the counterparty responds to a swap proposal using a signing transaction. Finally a burn transaction returns spendable HRC1 token balance to address E on the Ethereum chain ❾. From there it can again be spent or beamed to other ledgers. Validators in the network can crawl through the data in the public ledgers, starting from the recipe itself, along the genesis transaction and then choosing a branch that has not yet been validated. By querying a validator node users of the protocol can more rapidly verify that a transaction is valid, instead of having to crawl the entire transaction tree themselves.

4 Genesis of a *hybrix* Asset

4.1 Field Definitions

To instantiate an asset we need a commonly shared method of defining its properties. This is needed to ensure all nodes know how to interpret the data on all the different ledgers, and decode the transactions into classed asset groups and interpret accounting for the individually issued assets. What is needed is a common recipe format that does not rely on any single ledger system, yet can be validated by way of immutable storage of its content hash. Such a recipe will contain variables and field definitions to specify the name of the token, its economics and other details pertaining to its rule set. These variables are stored as field definitions in what we call a *Genesis Recipe*. An example recipe follows.

```
{
  "symbol":"hy.example",
  "name":"Example hrc1 token",
  "info":"This is an example of a
           hrc1 cross-ledger
           electronic token.",
  "import":"hybrixProtocol",
  "version":1,
  "contract":123,
  "mutable":true,
  "burnProof": "btc:185b3...32db3",
  "burnTargets":[
    "btc:1Counter...",
    "eth:0x000000..."
  ],
  "mutations":[ "btc:b4152...e9d10" ],
  "economy":{
    "supplyInitial":8000000,
    "supplyFixed":true,
    "factor":8
"mint":{
  "allowed":true,
  "chains":["btc","eth"]
},
"send":{
  "allowed":true,
  "chains":["btc","eth","waves",
                 "nxt","xem","ignis"],
               "rules":{
                  "default":
                      {"fee":0.25}
                }
},
"swap":{
  "allowed":true,
  "assets":["btc","eth","waves",
              "nxt","xem","ignis"]
},
  },
  "genesis":"btc:b4152...ba5cd,
                     eth:0x8a1...0ae06"
}
```

When a recipe has been created, it is hashed to create a proof that is registered on a ledger using a genesis transaction. This transaction proves the validity of the recipe that governs the token. Each token must have an identifier number that is unique. In case of a collision, validators will only accept the recipe that was proven first by way of the genesis transaction. If allowed by the recipe, it is possible for new tokens to be created by those other than the initial issuer. This is done by burning assets on the underlying chain. In principle the assets burned can only be rewarded according to the rules defined in the recipe. This ensures that price calculation of the burn assets can best be done on the decentralized exchanges, removing the need for an orderbook for the *hybrix* protocol. When a recipe is defined as being mutable, it is possible to update it, and send a new genesis transaction to the blockchain. This must be done from the same cryptographic secret key, as was used with the earlier recipe. The older genesis transaction must also be recorded in the recipe, so the chain of mutations can be followed and approved by validators. When a recipe is updated, the new recipe is passed around to other nodes using over a decentralized transport mechanism. Validators check a new incoming recipe for validity first, by comparing its hashes with available data in the blockchain, and authenticating that the updated genesis transaction has been done using the same secret key as the first genesis transaction. After this has been verified the new data can be applied as an active recipe.

4.2 Recipe Verification

We may verify the example recipe by pulling in the attachment data from the genesis transaction, which is located on the Bitcoin blockchain, and verifying the recipe contents by calculating the hash of it (excluding the genesis entry in the object), and comparing that to the hash recorded in the transaction data. It is also possible to record the genesis validation transaction on multiple distributed ledgers, to ensure deduplication of available verification hash data.

4.3 Autonomous Distributed Token Minting

When any actor can potentially launch a cross-ledger token, we assume this may result in the creation of a lot of tokens. This could be for a project or company asset, mere experimentation, or it could be in order to spam users. It is possible that the transaction cost of the underlying ledger system is not enough of a deterrent for those who want to spam the system. To mitigate this we propose the protocol ties a fee to the genesis of a new token. This fee is distributed to all network validators in order to incentivise the validation of transactions. We propose that issuance fees are thus defined means we need not rely on oracles, smart contracts or other complex consensus mechanisms to autonomously and decentrally govern the process of token issuance.

5 Mutation of Monetary Supply

The monetary supply of a HRC1 token is mutable if this has been allowed by issue in the economic rules of the recipe. There are several ways in which the supply may be mutated. By way of a transaction fee, minting or burning. If a transaction fee is enforced by the ruleset, the supply is subtracted from on every transaction. The fee may be specified in the recipe as a fixed amount, or as a percentage of the transaction. If this is enforced, a transaction will not be valid without this fee. In the case of minting, new units are added to the available supply. If the initial supply of the token is fixed, minting is only successful as long as there are still tokens left to mint. In the case of a limitless potential supply, an inflation of the total amount of units occurs. It is also possible to influence the monetary supply of a HRC1 token by way of minting in exchange for burning another monetary unit. We avoid the problem of needing a rate of exchange by only accepting the burning of native assets on a recipe-defined ratio. A rate of exchange must be defined by a price ticker on the centralized or decentralized exchanges, or the exchange rate is set by an oracle or a gateway service that defines the rate. This means burn transactions using the *hybrix* protocol do not depend on oracles or third parties to be performed successfully.

6 Validation of Transactions

6.1 Validation as a Service

External validation should be handled in a decentralized manner using a consensus amongst multiple validator nodes. Such validators nodes then basically provide verification as a service (VAAS), which in turn results in the inception of a validation market. The nodes crawl the tree of transactions checking them for validity.
Essentially this means that validators are basically nodes that follow the branches of transactions from the genesis transaction to a final user balance ledger entry. They do this for a fee that that is split among validators and examinators of validators. To create incentives for validators to verify chain integrity, we default to creating a reward structure that is defined by the network.

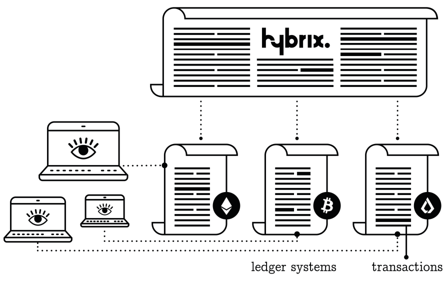

Figure 4: hybrix chain validators

Users transferring large amounts of value have a need to determine the validity of their balances, and they also likely have the capital to pay for this service. In sending a transaction they can opt to pay a higher fee, and this will result in more validators eager to validate the user's chain of transactions. A side effect of this is that it enables users that have less to spend on fees to be validated automatically on occasion, once their balance becomes part of a transaction branch that has been sent on to other users.
A state database containing the verified sub tree is maintained by validators which can be queried by users. A decentralized consensus state database maintained by a pool of validators will consist of a sub tree $\mathcal{T}'_n$ where n increments with each state update, providing a snapshot of the agreed upon valid transaction tree.
To ensure the recovery from a 51% attack on any one single chain, snapshotting by validators could enable network users to request the verification of the current ledger and balances state, regardless of a transaction history tainted by 51% attack damage. Requesting this from multiple validator sources could confirm to the user a properly accounted *hybrix* ledger, in spite of a damaged ledger.

7 Examinations

7.1 Validating the Validators

Validators need to be rigorously examined in order to find out if they are properly doing their job of validating transactions on the chains. In the case that all is going according to plan validators check the transactions and record their findings for the public truthfully. However, in the case of malfeasance not every validator may be trustworthy.

7.2 Previously Attempted Solutions

Several approaches to validated ledger systems have been made to try and solve this problem: who or what validates the work of the validators? Omni for example has chosen to validate from a centralized entity. This markedly reduces the complexity of the solution, but is not desirable in a system where one wants the decentralization to be as great as possible. Next to that the workload for validating the transactions grows over time, which puts all this work on the centralized validation entity. This decreases the security and integrity of the transaction ledger any time the centralized entity is not able to validate momentarily for whatever reason. Ripple, another example, has chosen a select amount of validators that have a sizable stake in the Ripple ecosystem to make sure transactions are done properly. The sanction for being a corrupt validator in Ripple is the possibility of having your stake reduced. This can not be done in our case, however, since *hybrix* is a more decentralized protocol. This means we cannot simply reduce the stake of any party in the network from an authoritative entity. Instead, we only have the ability to invalidate fees in hindsight according to the errors of a validator. If these fees have already been spent, however, executing such a disincentive could create a complicated situation for multiple parties. Transaction reversals, however, should only be an action of very last resort.

8 Common *hybrix* Index

Storing the genesis transaction ID, or other hash information in every transaction would require a significant amount of blockchain storage as the volume of transactions grows. Some token protocols employ such a method, even though it is not most efficient. The token protocol Omni, on the contrary, uses an index number for the asset ID in every transaction. Doing that requires a public index to be available. With Omni the registration of new tokens must be administered in a centralized database, and by a trusted entity. We define a method of posting information to a ledger to replace the centralized index of token identification. This entails `genesis` type transactions that contain a hash of the content of a recipe that defines the way a cross-ledger asset works. This means the `genesis` transaction can be used to verify the recipe file that describes in detail the rules and properties of an asset.

In a hybrix asset recipe the identifier of the asset is an index number. This index number is passed along in every transaction, similar to Omni. The difference is that the main index is a specific address to which new registrations are posted, instead of a centralized database and API. To avoid Sybil attacks[8] and squatting of index numbers, we propose the use of initiation burn values to accompany the creation of a new asset class. This way the validity of an asset can be verified by looking up the corresponding burn value.

Storing the index in every transaction takes up either 1 to 4 bytes of space. This makes the indexes with the least amount of bytes more desirable, since every transaction with it will be slightly less expensive to store in a digital ledger. On the other hand index numbers of a higher order that need more bytes are also less scarce. For this reason we utilize a costs table to ensure index numbers cannot simply be taken.

Thus follows that registering a token on the *hybrix* protocol requires a fee paid in *hy* tokens. This fee is then distributed to network validators over the course of a year by the *hybrix* protocol. Initially this process will be handled manually by the developers of the hybrix protocol and subsequently automated.

According to the amount of bytes used for the index, registration can be done in several tiers, depending on the cap one would like their token to be listed in. The caps that a *hy* token may be listed in are: Primary, Medium, Minor, Minute. We make a distinction so that larger organizations pay for the network to operate smoothly. In return they get the benefit of utilizing less storage per transaction in the ecosystem, the higher the tier.

Of the Primary tier, only 255 places are available. This is the largest registration cap, and is intended for governments and large-scale international businesses that need a very high security in validation, expect a large flow of transactions, and large market exposure. Fee for listing is 150000 HY. Renewal fee is 15000 HY annually.

The Medium tier has 65535 places available. This cap is intended for businesses that have a sizable market share or IPO listing, need proper validation, expect many transactions and have medium market exposure. Fee for listing is 15000 HY. Renewal fee is 1500 HY annually.

The Minor tier has 16777215 places available. This cap is for organizations or startups that have a small market share, need occasional validation, expect a few transactions and have little market exposure. Fee for listing is 150 HY. Renewal fee is 15 HY annually.

The Minute tier has 4 294 967 295 places available. This cap is for experimentation, small projects and individuals who want to put hybrix to work for them, expect very few transactions and have almost no market exposure. They can expect no validation, unless they request it from the market. Fee for listing is 15 HY. Renewal fee is 1.5 HY annually.

The above schema makes registering new IDs more expensive as a blockchain or ledger system's base asset attains a higher market value, and avoids indiscriminate spamming of the blockchains. Thus the common *hybrix* index is built using proof-of-payment operations. This provides us with two benefits. First of all the amount of registration spam is easily weeded out by analyzing the burn value, and checking to see if it adheres to the requirements for creating a new token. Secondly it injects value into the *hybrix* market by allocating the initial go-to-market cost on chain, and distributing it among the validators. It has been shown that burning or re-allocating economic value this way stimulates the market with value and trust, as also proven in previous blockchain projects like Counterparty [9]

9 Deterministic Libraries and API Connectors

We connect to a large variety of blockchain APIs using a peer-to-peer network daemon called *hybrixd*[10]. This platform contains recipes that include API connectors for a plethora of distributed ledger and blockchain systems. The calls made by this platform also abstracts the responses of every system it connects to back to a common format. The platform further contains a mechanism for activating deterministic libraries in a specific manner so that these can only be used for signing transactions, and expose abstracted functions to the platform. Automated scripts remove all unnecessary code and fetch calls from external deterministic libraries, and create a compressed and wrapped version of every library as a byte-encoded package. These small packages are used by *hybrix-jslib*[11] the javascript client-side library to sign transactions in a secure way. Deterministic functions are used to generate key pairs for all included ledgers from a single base seed. We use the fact that any address $\alpha \in A_L$ is generated from a key pair $k \in K_L$ using a one way transformation:

$$\psi_L : K_L \to A_L \tag{1}$$

For a meta ledger we define a seed $\tilde{k} \in \tilde{K}_{\bar{L}}$ that can be used to generate a corresponding key pair in each base ledger using the function $\chi_{\bar{L}}$:

$$\chi_{\bar{L},j} : \tilde{K}_{\bar{L}} \to K_{(\ell_{\bar{L}})_j} \tag{2}$$

We can then use the following substitution to reduce the required key pairs and addresses for each base ledger in the example in section **??** to a single seed key $\tilde{k}_B$:

$$\begin{aligned} k_B &:= \chi_{\bar{L},C}(\tilde{k}_B) \\ k_{B'} &:= \chi_{\bar{L},C'}(\tilde{k}_B) \\ \alpha_B &:= \psi_B(k_B) \\ \alpha_{B'} &:= \psi_{B'}(k_{B'}) \end{aligned}$$

As everything that has to do with deterministic transactions is handled from the single seed key as defined above, no data needs to be stored in order for a user to be able to recover key pairs and use these to control the ledgers the keys give access to. A module system in *hybrixd* makes it possible for developers to have their code integrate to a variety of transport mechanisms and external APIs. Next to that connector modules that query external APIs expose these connections as abstracted paths to which calls can be made that are homogenous. Modules can query each other as well as be called from the top-level API. On top of this stack the *hybrix* protocol is implemented as a module. The full-stack combination is a *hybrix-jslib* client, as well as a complete *hybrixd* node. Where less computing and storage resources are available a *hybrix-jslib* client can be used to sign and interpret transactions and get necessary data from a publicly available *hybrixd* node API.

10 Conclusion

We have proposed a system for meta-level transfers across multiple distributed ledgers without relying on centralized exchanges or decentralized atomic transaction compatibility. By design the system is open and transparent. The process of moving value between ledger systems is not controlled by a centralized party, as transactions can be created and signed client-side and sent peer-to-peer among users. We started with the usual framework of second-layer tokens specified by storing data attached to transactions, which provides a method of accounting on top of existing ledger systems, but is incomplete without a way to prevent double-spending. To solve this, we proposed a peer-to-peer cross-systemic-ledger network of validators of which said actors are incentivized to cryptographically verify the public history of second-layer transactions. Their work is rewarded by users in the network that need their ledger balance verified, and this simultaneously verifies ledger balances of users that are the source of the ledger state to be verified. Impostor attacks on the protocol are mitigated by using the consensus mechanisms of the underlying ledger systems to thwart attempts at double spending. Sybil attacks on the validator network are minimized by way of a service model on necessity basis. Apart from this, the user is autonomously able to verify the chain of transactions to ensure the authenticity of their token balance. We proposed a mechanism for creating new tokens without the need for a centralized index. This provides users the possibility to autonomously issue and mint second-layer electronic tokens. Network fees to issue new tokens make token squatting an expensive undertaking, in order to keep the second layer protocol free of spam. To ensure the recovery from a 51% attack on any one single chain, we proposed a recipe based protocol that enables a token issuer to update the possible chains on which a token can operate. Aside from this, snapshotting by validators enables network users to verify the current ledger and balances state, regardless of transaction history tainted by 51% attack damage.

Notes

[1]"How Bitcoin is Changing Online Commerce", https://www.forbes.com/sites/johnrampton/2014/07/02/how-bitcoin-is-changing-online-ecommerce

[2]"Risks of Centralization in Crypto Ripple and NEO", https://medium.com/@astralcrypto/risks-of-centralization-in-crypto-ripple-and-neo-4ee2de358c1

[3]"The Real Threat From Facebookś Libra Coin", https://www.forbes.com/sites/francescoppola/2019/06/30/the-real-threat-from-facebooks-libra-coin

[4]"The Blockchain: An Experiment in Governance Without Power", https://www.coindesk.com/blockchain-experiment-governance-without-power

[5]"JPMorgan Plots Blockchain Payments to Fight Transferwise, Ripple Threat", https://www.ccn.com/jpmorgan-blockchain-payments-settlement-fintech-ripple

[6]"Colored Coins", https://en.bitcoin.it/wiki/Colored_Coins

[7]"OP_RETURN", https://en.bitcoin.it/wiki/OP_RETURN

[8]"Wikipedia: Sybil attack", https://en.wikipedia.org/wiki/Sybil_attack

[9]"Why Proof-of-Burn", https://counterparty.io/news/why-proof-of-burn/

[10]"hybrixd", https://github.com/hybrix-io/hybrixd

[11]"hybrix-jslib", https://github.com/hybrix-io/hybrix-jslib