Liquichain whitepaper

A manifold chain for free, secure, lightweight, and fast distributed apps

I.The needs for a new blockchain

A brief history of the blockchain

Blockchain technologies have been developed in the 90's. In their 1990's article [1] Stuart Haber et W. Scott Stornetta introduced a notion of chained documents allowing to prove that a document has been produced before another one without practical way to alter this proof. Each document is written in the form of a hash in a block, the hash of an electronic document is a number obtained from the data of the document by a very specific algorithm; this algorithm is such that it is practically impossible to change the document's data without changing its hash. Since each block contains the hash of the document it certifies, a timestamp and a hash of the previous block, the deeper a document is buried in the chain of blocks, the more difficult it is to modify its timestamp as doing so would require modifying all the hash, and possibly the timestamps of its succeeding blocks.

If this method allows to assign a timestamp to a document, it also makes it practically impossible to modify the document itself. In 2009, in his article about Bitcoin [2], Satochi Nakamoto uses this chain of block to introduce a distributed database between machines belonging to people that neither know or trust each other, but who, by implementing the Bitcoin protocol, can perform crypto currency exchange between them and be convinced that no one is cheating (provided there are more honest participants than dishonest ones).

Transactions are written in blocks that a chained to each other: each block contains the hash of the previous block; hence the deeper a transaction is buried in the blockchain, the more difficult it is to modify it.

With Bitcoin, the blockchain is only used for cryptocurrency exchange, in 2013 Vitalik Buterin introduced with Ethereum [3] the notion of smart contract in the blockchain. Besides being used for crypto-currency exchange, transactions are now capable of calling functions called "smart contract" that consist in source code shared by all the network nodes. This source code can modify data associated to it and shared also on all the network nodes.

Ethereum allowed to democratize some decentralized apps like the ERC20 coins (full fledge crypto currencies that are deployed and executed on Ethereum) an NFT (non-fungible tokens).

There are several limitations with ethereum:

- The price of ETH, the native ethereum crypto currency that allow to pay transaction fee is very volatile and high enough to make each simple transaction expensive.

- Proof of Work, the consensus used accept a new block in the chain, is very energy demanding and introduce a long delay in transaction validation.

To overcome these limitations many alternative blockchains have been introduced since 2015. New consensus algorithms have been created in order to lower the energy consumption and decrease transaction validation's time among which:

- Proof of stake: In this protocol, used in particular by Polkadot[4] and Cardano [5], the user that can build blocks are chosen in regards of the amount of money they put at stake.
- Proof of elapsed time, proof of History: Used for instance by Solana [6], this protocol reduces the energy consumption by replacing users' intensive CPU usage by pause periods. It is used also by Hyperledger sawtooth [7] which is a permissioned ledger (users can join the chain only if they are validated by existing users)
- pBFT, IBFT, raft : Used by permissioned blockchain like Hyperledger Fabric [7], Hyperledger Besu [8], R3 Corda [9] those protocols use low energy and transactions are fast ace the use some election system for the node that build the blocs.

Even if much progress has been done, there are still several drawbacks inherent to existing blockchains

- Pb1: When a user loses his private key, he loses all his coins
- Pb2: When a private key is stolen, the thief can transfer the coins to any account he wants
- Pb3: When coins are transferred, the receiver do not have to accept the transfer.

If someone steals a private key, he can easily hide his forfeiture by spreading the coins to one account he holds and many randomly chosen active account.

Reducing energy consumption

Even though the new consensus reduced the CPU usage they could still be consuming globally a lot of bandwidth or storage space. One of the major reasons for blockchain bandwidth and storage consumption is that data is often replicated among all the nodes.

- Pb4: A blockchain should be designed to for low consumption of computation power, bandwidth and storage space.

Reducing decentralized apps creation complexity

A purpose of blockchains is to offer a trusted execution environment. Smart contracts that can be run on those environments necessarily have limitation, typically they must consist in functions that produce a deterministic output when applied on a state of the ledger. This implies that when creating a Dapp, developer must carefully separate the code that run on the chain from the code that run off chain, then implement the plumbing between the two.

- Pb5: Blockchain must address the Dapps creation process as whole not only the on-chain part.

Progress in that direction is taken for instance by Hyperledger firefly [11].

Addressing multi organizations business interactions

Dapps are created by organizations accessed by the user of their community.

- Pb6 : Organizations might not want to share data and business logic with each other
- Pb7 : Sharing a common blockchain infrastructure should not impose transactions fees to an organization that use it.

Going beyond formal organizations

With current blockchain setting up a decentralized organization requires important investments to rent servers, create smart contracts and apps, then host and maintain them

- Pb8: Blockchain should allow individuals to easily setup informal organizations.

References:
References

[1] Stuart Haber et W. Scott Stornetta, “How to Timstamp a Digital Document” (1990)
https://link.springer.com/content/pdf/10.1007%2F3-540-38424-3_32.pdf

[2] Satoshi Nakamo, “Bitcoin: A Peer-to-Peer Electronic Cash System” (2009),
https://bitcoin.org/bitcoin.pdf

[3] Vitalik Buterin,”A next generation smart contract and decentralized application platform” (2013)
https://blockchainlab.com/pdf/Ethereum_white_paper-a_next_generation_smart_contract_and_decentralized_application_platform-vitalik-buterin.pdf

[4] Gavin Wood, “Polkadot: vision for a heterogeneous multi-chain framework” (2016)
https://polkadot.network/PolkaDotPaper.pdf

[5] Aggelos Kiayias, Alexander Russell, Bernardo David, Roman Oliynykov,” Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol” (2017)
http://cryptoverze.s3.us-east-2.amazonaws.com/wp-content/uploads/2019/01/10011338/cardano-ada-whitepaper.pdf

[6] Anatoly Yakovenk, “Solana: A new architecture for a high performance blockchain” (2019)
https://solana.com/solana-whitepaper.pdf

[7] Kelly Olson, Mic Bowman, James Mitchell, Shawn Amundson, Dan Middleton, Cian Montgomery, “Hyperledger Sawtooth: An introduction” (2018)
https://www.hyperledger.org/wp-content/uploads/2018/01/Hyperledger_Sawtooth_WhitePaper.pdf

[8] Mike Hearn, Richard Gendal Brown, “Corda: A distributed ledger” (2019)
https://www.r3.com/wp-content/uploads/2019/08/corda-technical-whitepaper-August-29-2019.pdf

[9] Hyperledger Besu, https://www.hyperledger.org/use/besu

[10] “Hyperledger fabric: Open, Proven,Enterprise-grade DLT” (2020)
https://www.hyperledger.org/wp-content/uploads/2020/03/hyperledger_fabric_whitepaper.pdf

[11] “Introducing Hyperledger FireFly, a Multi-Party System for Enterprise Data Flows” (2021)
https://www.hyperledger.org/blog/2021/09/28/introducing-hyperledger-firefly-a-multi-party-system-for-enterprise-data-flows

II.The Liquichain vision

The purpose of liquichain is to address the above problems by providing a blockchain centered on communities and a low-code development studio allowing to build decentralized apps.

Communities are individuals that entrust each other for observing the rules defined in some formal or informal (smart) contract, share some resources and commit to participate to some of the group activities.

An individual joins liquichain through a community, each community he belongs to is a circle of trust. This approach makes liquichain neither an open nor private blockchain, it is a manifold version of the user space where anyone can join the chain if he belongs to one of its communities, but 2 individuals might not have any community in common. This allows for fast transactions between members of a community.

Once individuals know each other through a community, they might start to know each other personally, this level of trust allow cross community interactions as resources shared by friends will help both their community to run their activity. It is data and process sharding between community members and friends that make liquichain energy efficient.

Liquichain eventually allow individuals to build their community, with their organization rule, own fee structure and dapps. The only constraint brought to liquichain communities is to acknowledge the global trust structure and implement the interconnection protocols.

Liquichain comes with several plug and play components

- A consensus called proof of peers allowing for fast inter-communities transactions.
- A messaging and filesharing library allowing each user to share resources of its devices with his friends and communities.
- A voting library allowing communities to implement their decision processes

With interoperability and easy adoption in mind, liquichain is built to be compatible with many existing technogy :

- Ethereum json-rpc interface
- Solidity smart contracts
- Bittorrent, IPFS, ethswarm, S3 compatible protocol for file sharing
- Multichain bridge through multichain.org integration

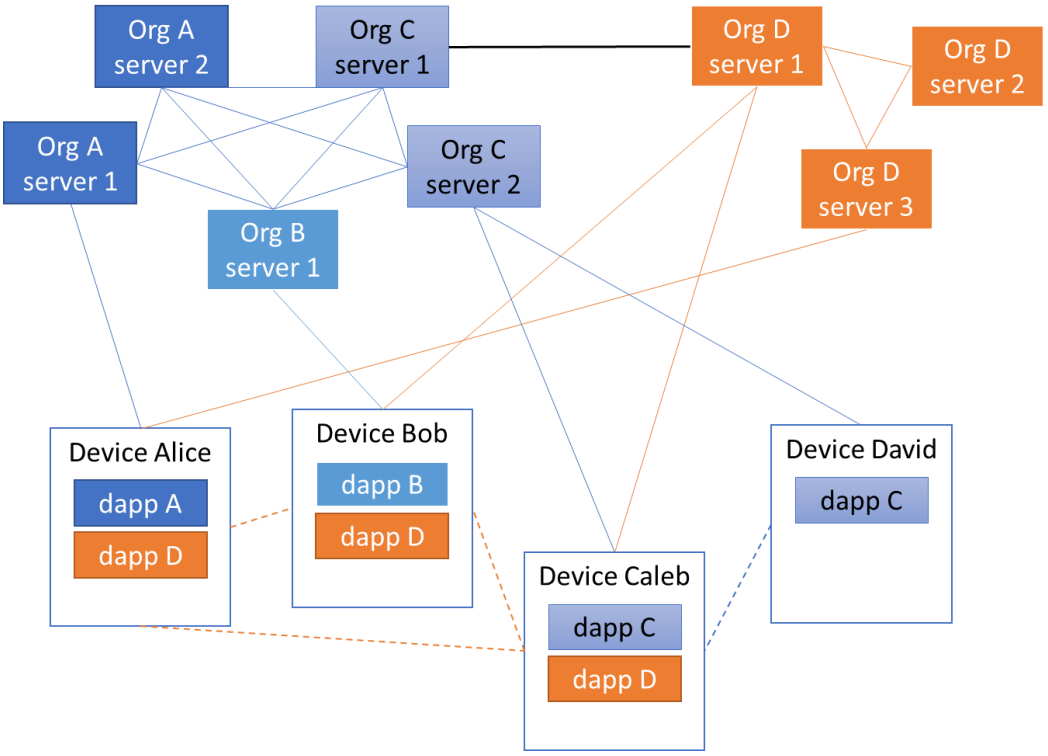
- Payment gateway interface with connector to major marketplaces
- REST interface to organization nodes dapps APIs.

III.Topology of the network

There are mainly 2 types of hardware participating to the blockchain

- Servers, that are machines dedicated to running the blockchain, and are typically own by an organization, are mostly online with some stable public IP. Servers are typically a recurring cost for their organization but are part of the resources needed to get income.
- Devices (mobile, table, laptop, desktop) that are machines for which the blockchain is just an app, with limited resources and difficult to reach directly for communication. Devices are typically a one-shot cost for their owner then bandwidth and electricity are recurring cost, but they are usually not part of the resources providing income.

Organizations' servers are running permissioned private chains that allow to execute the smart contracts of the dApps they provide. In case decentralization of authority is an important feature, the organization can join a group of organizations (consortium, association, partnership…) and extend its private chain to a permissioned chain involving all the organizations of the group (like organization A, B and C in the diagram below)



This diagram shows a network of 8 servers and 4 devices. Organizations A, B and C are running a permissioned blockchain that handle transaction and store states of 3 dApps : dapp A, dapp B and dapp

C. Organization D is running a private blockchain for its dapp C. As nodes of the liquichain blockchain all organizations obey the following constraints:

- They are EVM based blockchain exposing standard eth-rpc API.
- They run the liquichain smart contract that hold the public identification of all the peers and the identity validation network.

Devices are running dApps and communicate with the servers providing the dApps and the peers running the same dApps. As described in the next section, the devices participate in transaction forging and storage of smart contract state.

Up to the network limits imposed by NATs and requiring TURN servers, we can have a network composed only of devices. This allows for informal organization to run dApps without having to invest and maintain servers.

IV.Liquichain Nodes

As we saw in the previous sections, they are 2 kinds of nodes in the chain network: servers and device.

Server node

Liquichain provide a flexible server node that allow organization to reuse their existing infrastructure in terms of database, messaging system, of frontend technology. It also offers a low-code development studio for creating new decentralized apps. The server nodes are running private EVM based private blockchains that belong to one or more organizations. What make them part of the liquichain network is that they provide, like the device nodes a set of services to all liquichain members:

- Online presence service
- Messaging routing service
- Storage service
- Voting service
- Payment service

Device node

Nodes running on the device are executing the dApps, they offer the same liquichain services as server but have several differences:

- They might have limited resources
- They are expected to go offline often and have only feew connection to other peers
- They can run several liquichain apps provided by different organizations

V.Vote

Liquichain proposes a flexible voting system that can be used as a building block of decentralized autonomous organizations.

Vote delegation and domains

The organization can assign resolutions to domains (economy, finance, …) so that voters can delegate their vote to the person of their choice only for some specific domain.

Domains are represented as unsigned integers and can be thought as organized in categories by splitting their representation in fixed size chunk. For instance, using 4bytes integers, we could be split domains in four 1-byte domain levels:

1.0.0.0 Food and beverages
1.1.0.0 Primary
1.1.1.0 Mainly for industry 1 Intermediate
1.1.2.0 Mainly for household consumption 2 Consumption
1.2.0.0 Processed
1.2.1.0 Mainly for industry 3 Intermediate
2.0.0.0 Industrial supplies not elsewhere specified
2.1.0.0 Primary 5 Intermediate
2.2.0.0 Processed

This allow to easily add domains and organize them In a way for voter to be able to delegate a domain to Alice but one of its subdomain to Bob.

The domain with id 0 represents the entire space of resolutions.

Delegation can be configured as an active or passive process

- In the passive case, delegation information is shared, and a voter can participate to a vote even if offline during the vote.
- In the active case, a voter can keep the delegation information private and for each vote send a delegation to the person that will vote for him.

Proposals

Proposals are set of resolutions that all belong to a single domain (which could be 0), each resolution can be individually voted. They consist of a description and a vote type consisting of a set of choice (yes/no/abstention, choice of X among N,…) and an tally method.

The Tally method consist in an aggregation procedure followed by a result selection.
For instance:

- In a “yes/no” choice, the aggregation could the 2 dimensional vector (x,y) obtained by using the substitution “yes”->(1,0), “no”->(0,1) and summing the votes, and the result obtained by selecting Max(x,y) provided x+y is greater than a quorum.
- In a “yes/no/abstention” choice, the aggregation could the 3 dimensional vector (x,y,z) obtained by using the substitution “yes”->(1,0,0), “no”->(0,1,0),”abstention”->(0,0,1) and summing the votes, and the result obtained by selecting Max(x,y) provided x+y+z is greater than a quorum.
- in a “choice of 2 among A,B,C,D” the aggregation could the 4 dimensional vector (x,y,z,t) obtained by using the substitution “A”->(1,0,0,0), “B”->(0,1,0,0),”C”->(0,0,1,0),”D”->(0,0,0,1) and summing the votes, then selection the 2 highest coordinates.

Although the selection method can differ from vote to vote, the aggregation always consist in summing vectors with 0 or 1 entries.

While no constraint is set on the description of a resolution, a smart contract enforceable resolution can be created by representing the resolution as a difference to be applied on a reference document. Liquichain provide a way for organizations to easily create proposals by mapping domains to sections of document (bylaws for instance) and creating proposals from modification to this document.

Anyone can write proposal, but only proposal with sufficient support can be schedule for voting.
Each organization can define the way proposal are accepted depending on the domain they belong.

For instance, organizer could require a proposal for status modification to get support of 75% of its shareholders, but a proposal for budget modification to get support of 10% of its members.

In that example Member and Shareholders are custom groups that organization can administrate, the administration being also delegable to persons or smart contracts.

Vote

Several Vote processes are available depending in the privacy, cost, and the simplicity required.

For vote where privacy of voters is not a concern, liquichain proposes voting smart contracts based on openZeppelin Governor smart contract (https://docs.openzeppelin.com/contracts/4.x/api/governance).

A completely private vote is performed is several steps:

Step 1: Vote creation. From the proposal a Vote smart contract is created. The organizer set the starting and ending registration and vote dates and publish off chain the list of eligible voters.

Step 2: Eligible voters interested in participating to the vote register themselves by sending a public key corresponding to a randomly generated private key following the procedure in [1].

We denote by *G* a cryptographic group of order *q* and *g* a generator. Each voter selects a private key *x_i* which is a number belonging to *Z_q* and publishes its public key *g^(x_i)*.

Step3 : After the registration period, the organizer assigns a list of voter ids to registered voters and publish a Merkle tree of the voters' id and voting power.

In case we don't want to be able to resolve the votes of the voters that have delegates, the voter id tree can be built from the list of registered voters and their power in a way that each leaf have at least one sibling with the same voting power.

Example: Let say we have 3 registered voters A,B,C with voting power 4,15,1 respectively. Then by decomposing 4=3*1+1*1, 15= 3*4 + 3*1,1=3*0+1*1 , A receives voter id 1 with voting power 3 and voter id 6 with voter power 1, B receives voter id 2,3,4,5 with voting power 3 and voter id 7,8,9 with voter power 1 and C receives voter id 10. We have then two voting sets (1,2,3,4,5) each member with voting power 3 and (6,7,8,9,10) with voting power 1.

Step 3: Vote cast is performed for each voter id in a maximum of 3 rounds. In the first-round voter casts their vote by getting the public voting keys of each member of his voting set then compute the blinding key (as defined in [2]):

It is obtained by computing in G the product and quotient of over the voting set:

$$g^{y_i} = \prod_{j=1}^{i-1} g^{x_j} \Big/ \prod_{j=i+1}^{n} g^{x_j}$$

This blinding key is used to compute the vote $g^{x_i y_i} g^{v_i}$ (with v_i being either 0 or the voting power) and send it along with a zero knowledge proof that the value v_i is correct to the vote smart contract. If the smart contract contains all the votes of the participants, the votes are aggregated by the organizers simply multiplying the votes

$$\prod_i g^{x_i y_i} g^{v_i} = g^{\sum_i v_i}$$

and the result is registered.
If after round 1 not all participants casted their votes, then an identical second round is performed with half the delay after exclusion of participants that didn't cast their vote.
If the second round fails, then a backup round is performed where votes are sent to the organizer for centralized tallying.

Step 4 : Tallying. After the vote cast period, the organizer trigger the tallying which simply add the result of each voting set and the failed votes. The proof of which is posted with the result.

Note on proof of vote: The zk-SNARK circuit used by participant to proof that their vote is valid is modified so that is the vote reflect the number of delegated votes.

References

[1] Hao, F., Ryan, P.Y., Zielí nski, P.: Anonymous voting by two-round public discussion. IET Information Security 4(2), 62–67 (2010)

[2] M. ElSheikh and M. Youssef "Dispute-free Scalable Open Vote Network using zk-SNARKs" (2022) https://arxiv.org/pdf/2203.03363.pdf

[3] A. Zapico & al "Caulk: Lookup Arguments in Sublinear Time" (2022) https://eprint.iacr.org/2022/621.pdf

VI.Proof of peers' consensus

We describe here the proof of peers consensus at a high level, see Annex 1 for the details of the messages exchanged between peers and transactions sent to the main chains.

Account creation

When a user, called creator, creates an account (private and public key), he signs with its private key a message containing his name, date and place of birth and a passphrase; this signature is called identity proof of the account.

He then sends an identity validation request to peers that are already part of liquichain. We assume that the creator knows the account address of those peers. The identity validation request contains the creator's account address and a signature of the account address of the peer. It can be sent by any mean like QR-code, URL, a message in one of the organization's mobile app, …

When a peer receives such a request, he can accept or reject it. If he accepts to validate the identity, then he sends back an acknowledgement to the creator that consist in the signature of the creator's account address. If he rejects it, he can later send an alert to the main chain.

Once the creator receives 3 signed acknowledgements from the peers (that we then call friends) he can create an account on liquichain by sending a transaction containing the account address, the proof of identity and the 3 (or more) friend address and acknowledgement. If the transaction fails, the process stops here.

The user must then start the identity validation process with each of its friends.

If this is process is not finished within 2 epochs, the user account is deleted, and the friends get a warning.

Identity validation by friends

The creator sends his friend a signed message containing the account identity proof. The Friend and the creator then enter in an interactive session by any mean (physical meeting, video conference, audio call,…) that both the friend and the creator feel comfortable with. The creator then provide is name, date and place of birth along with the passphrase of the account and the friend validate that it matches the account identity proof. If this matches then the friend send an identity validation transaction to the liquichain smart contract by signing a message containing a concatenation of the creator' account address and its identity proof.

Once the liquichain contract receives 3 friend identity validations we say that the account is valid. Note that some organization might decide to require more validation for their apps. An invalid account is eventually deleted from the chain.

Identity validation by neighbors

Beside the validation by friends, a user may also have his identity verified by peers that are automatically selected by liquichain nodes. Those peers are close, in algorithmic way to the user and we

call them its neighbors in the sense of identity validation (there are also neighbors for money transfer or other smart contract transactions).

Distance between peers relatively to a specific address:

We use the classical XOR distance between two nodes used in a kamdelia network. This allows to define a distance on some address space. For instance, it gives a distance on the space of liquichain wallet addresses. The drawback of using this single distance for storing information and validating transaction by neighbors is that a user could get to know those neighbors and start to agree on validating invalid transaction (like double spending). It is therefore of prime importance to have a different distance per smart contract so each such contract has a different set of neighbors or even have a set of neighbor that get renewed regularly. For this purpose we introduce the notion of XOR distance relative to a specific address.

Let U1 an U2 be the address of user1 and user2 and A be a specific address we define

$$d_A(U1,U2) = d(\text{AES-256}(A,U1)), \text{AES-256}(A,U2))$$

Where d is the XOR distance and AES-256 is the 256 bits Advanced Encryption Standard with the first argument being the key.

Using the distance related to the address of its latest epoch block, liquichain maintains in a verifiable way its network topology at all time by making sure each node has enough friends and neighbours for performing fast and secure transactions.

For each peer liquichain server exposes a announce service that is an extend the bittorrent tracker protocol. This service act as a presence service and allow to collect information about the connection patterns of the nodes. For each peer a time presence pattern is computed, this is a list of 24 values between 0 and 1 that represent the chance of peer to be online.

Using this information, liquichain servers can compute for a given user the estimated number of friends and neigbhors that will be online at a given time (see Annex 3 for computation details). If this number is going lower than a threshold, the server will then send identity validation requests to peers that are the closest to the user. The contacted peers then have a delay to perform the identity validation. If they do not answer within some delay then a notification of misconduct is triggered (see misconduct section).

Validators

The friends and neighbors that performed successfully the identification of a user are called its validator for the liquichain smart contract. More generally the peers that allow to host the smart contract state and perform transaction validation are call the validators of the smart contract.

When a smart contract is provided by an organization, its node can be automatic validators of each of its users. On the other side some smart contract could run entirely on friend and neighbors devices and not involve any server.

Localizable transaction:
For a given smart contract, a method is called localizable if it only modifies the state of smart contract variables that are maps with address map.

A smart contract is said localizable if non localizable methods do not modify any variable modified in a localizable method.

A transaction is then localizable if it is a call the localizable method of a localizable contract.

Whereas non localizable variables of the smart contract can only be stored in the server nodes of the organization running the smart contract, the localized variables can be stored on the validator devices instead. When a peer becomes a validator of a user, he downloads the current state of the user from the user itself and validate it from the other validators (details in Annex 4).

If the validators are define as closest ones to the user with respect to an epoch block address then each epoch a new set of validator replaces the validators from the previous epoch.

Transaction validation

We describe here the proof of peer consensus algorithm for registering a licoin transaction between 2 accounts. This can be generalized to any smart contract localizable transaction that involves the sender and optionally a second address (called the receiver).

When a sender, with address U1, want to send a transaction to a receiver U2he starts by establishing trust paths then he creates a transaction and send it to its online validators.

When receiving the transaction, the validator verifies its signed by a user he is validator for, that the nonce is the previous incremented by one, and locally execute the transaction to make sure it is valid.

He then stores the result of the transaction as a state change and forward its validation to the next users on the trust path.

When a user in the trust path receives a validated transaction, he signs it and forward it to the next user down the path.

When the receiver gets a validated transaction signed by the user on the path he can:

- Decide that he has enough trust and send a signed ack
- Wait for getting more validated transaction
- Refuse the transaction by sending an ack and trigger a misconduct notification.

The ack then travels back the to the validator that then update the state, increase the nonce and dispatch the validated ack back to the user and the other validators.

REferences

[12] “Decentralized identities v1.0” (2021) https://www.w3.org/TR/did-core/

[13] “Verifiable Credentials Data Model 1.0” Manu Sporny; Grant Noble; Dave Longley; Daniel Burnett; Brent Zundel. W3C. (2019). W3C Recommendation. https://www.w3.org/TR/vc-data-model/