WHITE PAPER

DecentraVote

Electronic Voting secured by Blockchain

Version 1.0

DR. ZOLTAN FAZEKAS

Blockchain Expert at iteratec

Contents

Digitization impacts all areas of an organization, including its governance. Countries are successively changing their legislation, allowing shareholders of joint-stock companies, as well as members of cooperatives and associations to attend in general meetings by electronic means. They are thus able to participate in all aspects of the decision making of their respective governing bodies without the need of being physically present. This whitepaper outlines our approach to provide a tamper-proof solution for electronic voting (e-voting) based on blockchain and cryptography[1].

> Electronic voting fosters participation in general meetings.

As opposed to many other use cases, the benefits of blockchain for e-voting have been recognized for some time: censorship resistance, immutability and verifiability of the votes combined with unforgeable digital signatures of the voters and the timestamps issued by the blockchain. These properties, supplemented by smart contracts codifying the voting rules, with a number of network nodes ensuring that they are applied correctly, make blockchain technology so appealing for this use case.

[1] DecentraVote was designed for general meetings with thousands of members and resolutions with a value at stake which can be secured by the underlying blockchain network. Nationwide political elections and other high-risk decisions are not in the scope of DecentraVote.

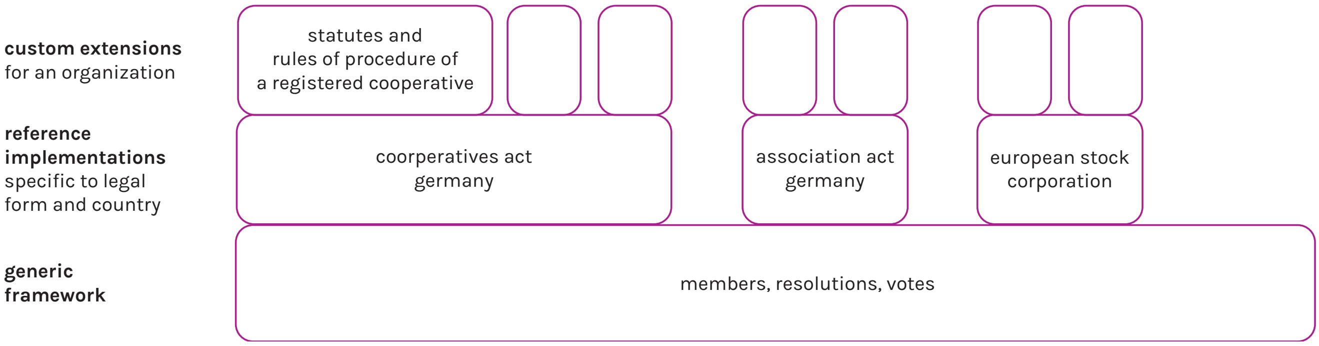
custom extensions
for an organization
reference implementations
specific to legal form and country
generic framework
statutes and rules of procedure of a registered cooperative
coorperatives act germany
association act germany
european stock corporation
members, resolutions, votes

Figure 1: Modular construction of DecentraVote

Governing Bodies and Regulations

The general meeting, the highest governing body of many organizations, needs to convene regularly in order to make binding decisions. The higher the number of members, the more difficult it gets to meet at a single venue. For this reason, the general meeting usually only convenes once a year. If members exceed a certain number, only elected proxies are allowed to attend. This implies, that members can no longer exercise their voting rights directly. By using an appropriate e-voting solution, organizations could schedule general meetings anytime and enable their members to cast votes remotely.

> DecentraVote anticipates and mitigates the risk of internal abuse of power.

The general meeting co-exists with other statutory bodies of an organization like the board of directors. Their competencies and responsibilities are regulated by the respective law applicable to the legal form in the country of registration as well as the individual statutes of the organization. To protect the interests of the general meeting, we don't rely on the chairperson designated by the directors or other officials of the organization to stick to the rules because of their legal obligations and personal liability. Instead, we envisage technical solutions that protect from abuse of power[2].

> DecentraVote complies with legal provisions and the statutes of organizations.

The governance of individual organizations can vary a great deal. Their statutes can define specific roles, committees and rules such as a quorum and the majority required for certain decisions. Because of very different legal frameworks and individual provisions in their statutes, organizations need a customized e-voting solution. They can build it on the generic framework of DecentraVote or use a reference implementation tailored to their legal form in the country they are registered in and extend it with specific provisions of their statutes (see Figure 1).

Decentralized Voting

A secure e-voting system must ensure that only members eligible to vote can participate. It needs to guarantee that they can only cast one vote and can verify that the vote was accepted and counted. It must be able to protect against anybody tampering with the votes cast or finding out who has cast which vote[3]. It has to prevent the detection of interim results before the vote ends and enable everyone to check the outcome of the vote afterwards.

Centralized e-voting solutions are already available[4] and are secure if operated accordingly. Voters can't

[2] We only assume that all actors are rational enough to avoid behavior with the certainty of negative consequences for them. We make sure that such behavior can't go undetected.

[3] Some organizations could also require their members' ability to plausibly deny participation in an anonymous vote. To support that we would only need to replace the deterministic secrets used in DecentraVote by random secrets per anonymous vote and destroy them afterwards.

[4] Polyas (https://www.polyas.com/) was certified in 2016 according to common criteria standards.

fool the system and cast multiple votes, prevent other voters from casting their vote or see the votes cast by others. Nobody, except the staff of the service operator, is able to interfere in the system. But what if they didn't set it up as intended and gained access to the voters' credentials? Since voters don't see what is happening inside the system, they would have practically no chance to notice manipulations. This is an inherent risk of any system run by a single operator. Whoever it is, it can't be fully protected from internal fraud or abuse.

Redundancy is the common way to solve problems with a single point of failure, in our case, the service operator. We can engage two or more independent service operators running the e-voting system in parallel and compare their respective outcomes. In essence, this is the idea behind fault tolerant distributed systems such as blockchains.

> By using blockchain, the need to trust the operator of the solution, is superseded.

Decentralized e-voting solutions using blockchain[5] are not based on the assumption of an honest service operator. Instead of a single trusted entity they rely on several independent service operators. They work simultaneously but none of them can interfere in the system without the consent of the others. Transparent and immutable logs of every change to the data and to the programs enable anyone to monitor the integrity of the system.

Choosing the Right Blockchain

DecentraVote can be deployed on any Ethereum-based blockchain network. Since the security of the solution heavily relies on the underlying network, organizations need to consider which one to use. There are a few options to choose from[6].

In permissioned blockchains the blocks are created by authorized entities, the validators. Their identity is public and their reputation is at stake. They mutually control each other and exclude validators failing to follow the rules. Several of them would have to be involved in an attack to be successful[7]. This increases the difficulty of an attack substantially, compared to systems with a single service operator. The more validators a network has, the more secure it can be considered. Unfortunately, the number of validators in a permissioned network is limited by the consensus algorithms used.

The number of validators in a permissionless network is practically unlimited. Their identity is unknown since every user can create blocks without the need for permission. This imposes the risk of Sybil attacks: it is unclear if validators are controlled by a single actor or represent different actors. To restrict the ability of any actor to create a disproportionate number of blocks, consensus algorithms based on limited resources instead of the validators' identity are used[8]. Acquiring those resources within a short time is difficult, so the attack would need to be prepared over a longer time frame. Coordinating a collusion among participants, who already own the resources necessary for the attack, also involves a considerable risk. The attack wouldn't remain undiscovered forever. It would damage the trust in the network and destroy the value owned by each participant. This is the reason why they rather protect the network by monitoring blocks and discarding those which break the rules.

> The blockchain network determines the security, scalability and costs of votes.

[5] The blockchain-based solution Polys (https://polys.me/) was announced to be made public in 2017.

[6] Besides the Ethereum Mainnet there are several regional and industry specific consortia networks like Alastria (https://alastria.io), Energy Web Chain (https://energyweb.org) or bloxberg (https://bloxberg.org).

[7] In order to manipulate the system 1/3 of them need to collude or be hacked.

[8] The difficulty of obtaining the computing power (Proof of Work), cryptocurrency (Proof of Stake) or other resource that is needed to successfully attack the network determines its security.

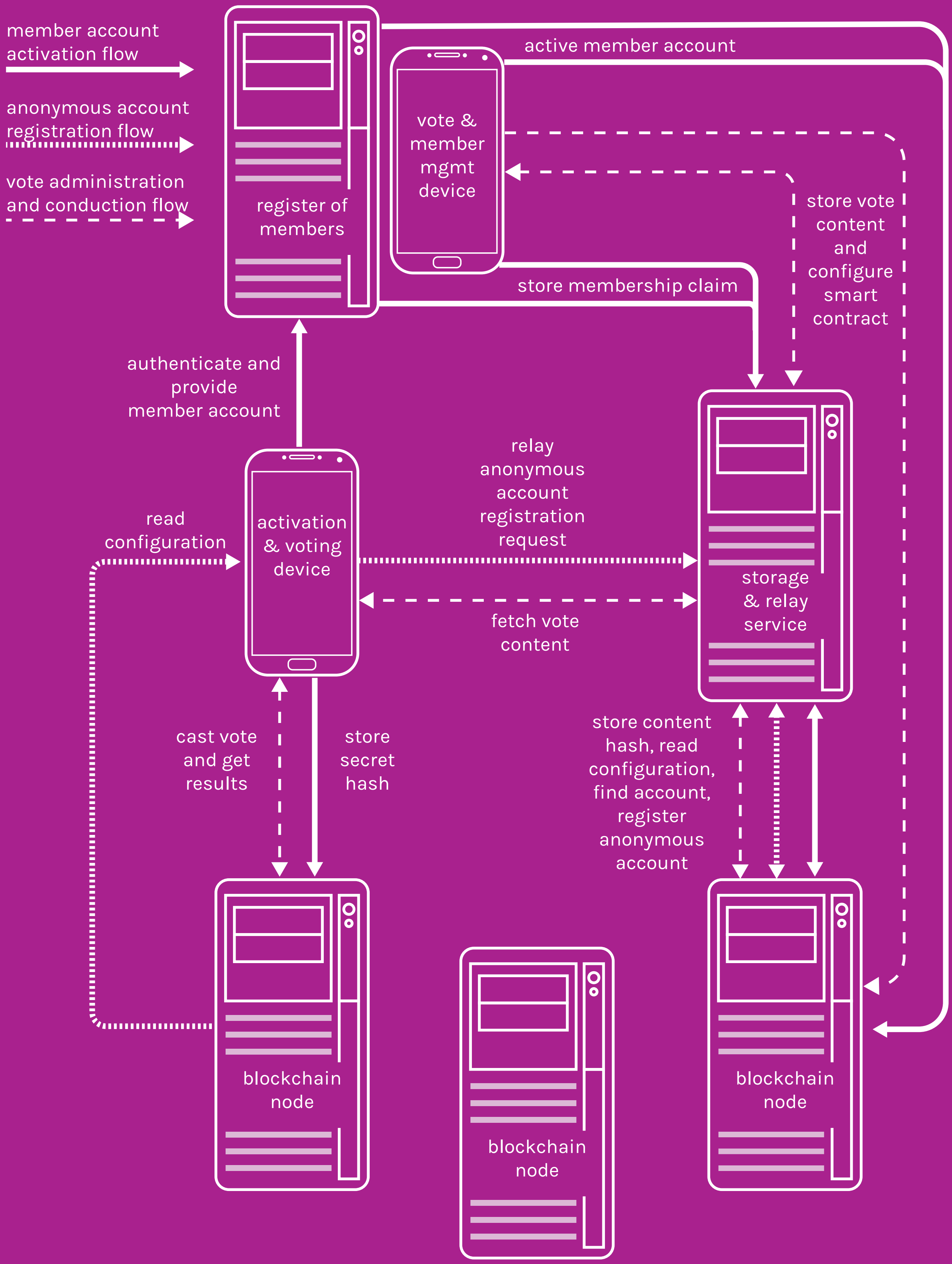
member account activation flow
anonymous account registration flow
vote administration and conduction flow
register of members
vote & member mgmt device
active member account
store membership claim
store vote content and configure smart contract
authenticate and provide member account
activation & voting device
read configuration
relay anonymous account registration request
fetch vote content
storage & relay service
cast vote and get results
store secret hash
store content hash, read configuration, find account, register anonymous account
blockchain node
blockchain node
blockchain node

Figure 2: Overview of DecentraVote components and their interdependencies

The downside of the high number of validators in permissionless networks is the low number of transactions confirmed within a given timeframe[9]. This can be a bottleneck if a vote with thousands of members needs to be completed within a few minutes. In addition, the confirmation time of transactions depends on the price paid. Setting the price low will make the vote cheaper but would also take substantially longer.

Hosting the User Interface

Members can participate in e-voting from anywhere using their own device (see "Activation & Voting Device" on Figure 2) connected to the Internet. They need to install a DApp browser[10] which provides a wallet and an interface for connecting to the blockchain.

The user interface of DecentraVote is a single page application running in the browser which interacts with smart contracts deployed on the blockchain via the RPC node (see "Blockchain Node" on Figure 2) configured in the DApp browser[11]. We need to make sure that the user interface hasn't been manipulated to display fake content of the resolutions or to alter votes before they are signed and submitted to the blockchain.

> The user interface and all information displayed are tamper-proof as well.

DecentraVote contains a storage service (see "Storage & Relay Service" on Figure 2) which hosts the files of the user interface, the content of the resolutions as well as membership claims and all other security-critical software components and data which can't be stored on the blockchain. This storage service is provided by the directors but could be operated by any member of the organization as well[12]. The storage service responds to requests containing a hash with the corresponding data. Each request must be signed by an active member account. This prevents non-members from accessing confidential information of the organization. Instead of blindly trusting the service we use a script running in the browser to check the integrity of its output. The script verifies the user interface files loaded from the storage service before they are executed. This guarantees that the user interface hasn't been tampered with. Before displaying any data requested from the storage service the script verifies those data too. The script performing the verification is part of the index.html file, which is the entry point of the user interface. To verify the integrity of this file, members can save it on their device, inspect its source code and compare its hash with the one published in the smart contract using a hash calculation tool available online or on their operating system. The link to the index.html file stored in their browser favorites includes an URL parameter with the address of the smart contract of their organization. Each time they open the link the verification script fetches the hashes of the files of the user interface from the smart contract and requests the corresponding files from the storage service. It computes the hash of each file locally and compares it with the hash found in the smart contract. If equal, an HTML element corresponding to the file is created. A similar procedure is applied to values displayed by the user interface which have been returned by the storage service.

The storage service can't fake the requested data because their hash is available on-chain. However, it could completely or selectively refuse to respond and withhold the data depending on the requesting member. To avoid this problem the storage service must submit a transaction with the hash of the data it is requested to persist. Thereby it acknowledges the receipt and commits to ensure the availability of that data. If it fails to do so, members can switch to one of the alternative storage service providers listed in the smart contract of their organization. To report a non-responsive service provider, members must submit their request on-chain. If the service provider doesn't respond within a specified number of blocks, members can request its removal from the list of available storage service providers.

Managing Members and Funds

Members are represented on-chain by the address of an externally owned account, the member account. The membership claim containing their name is persisted

[9] For the Ethereum Mainnet the current limit is about 20 transactions per second.

[10] The most popular option is MetaMask (https://metamask.io/), which provides a mobile browser and an extension to existing desktop browsers as well.

[11] Members can use publicly available RPC nodes of service providers like Infura (https://infura.io/) for free or run a node themselves.

[12] Any member can start to provide a storage service after fetching the data from other running storage services for the hashes published on-chain.

by the storage service and is only accessible for members. The member account and the membership claim are tied together by the activation transaction. Each member can only have one account active at a time. In case of loss of their private key, members can replace their old account with a new account after renewed authentication. If their membership terminates, their member account is deactivated on-chain and their membership claim is deleted from the storage service. Bodies of the organization like the board of directors and registers of voters[13] are represented by lists of the corresponding member accounts.

In order to get activated, members provide the address of their blockchain account to a director along with proper authentication and wait until they see that address and the hash of their membership claim appear in an activation transaction. Directors submit transactions activating new and deactivating retired and excluded member accounts (see “Vote & Member Management Device” on Figure 2) in accordance with the statutes of the organization.

> Users in a register of members can activate their member accounts themselves.

If members of an organization are already recorded in a register of members maintained by the directors, the register can help with the activation of those members by acting as an oracle which testifies identity and membership status[14] (see “Register of Members” on Figure 2). In this case, the members must authenticate by logging into the register and providing the address of their blockchain account for activation. The oracle creates and sends the membership claim to the storage service and submits a transaction with the member’s address and the hash of the membership claim to the blockchain to activate the member. The activation can be accepted by the smart contract immediately or become subject of a vote depending on the statutes of the organization. Using the hash submitted along with the activation transaction all members can request the membership claim from the storage service and get informed about the identity of the new member.

When new member accounts have been activated, the smart contract sends them some initial funds[15]. If the balance of the member account falls below a certain limit, members can request a refill from the smart contract. Member accounts must not be used for transactions unrelated to the organization, i.e. members are not permitted to transfer funds to other accounts or interact with smart contracts which haven’t been deployed by their organization. The organization can’t prevent that but can monitor their transactions and terminate their membership if they do so.

As soon as their member account has received funds, members can submit their first transaction sending the hash of a unique secret[16] to the smart contract which adds it to its internal list of potential voters. Using this secret, members can prove that they are included in the list of voters without disclosing their identity.

Preserving Anonymity

Member accounts can only be used for open votes. In order to maintain anonymity and unlinkability, voters must register a new account[17] each time they want to cast an anonymous vote. These single-purpose accounts are called anonymous accounts and can only be used once to cast a vote. Since a newly created anonymous account has a zero balance, it needs to be topped up with a small amount of funds sufficient for paying transaction fees for casting a vote. The top-up is part of the registration of anonymous accounts.

> Users register anonymous accounts without disclosing their identity.

[13] The register of voters can differ from the register of all members. If a resolution concerns concrete members, e.g. the discharge of the board of directors, these members aren’t eligible to vote.

[14] We implemented an oracle using Keycloak (https://www.keycloak.org/) to integrate existing identity providers and maintain the mapping between members’ identity and their blockchain account.

[15] Member accounts need funds for paying transaction fees.

[16] The secret is the signature of the member’s blockchain address created with the member’s private key.

[17] The private key of an anonymous account can be deterministically derived from the member’s secret and doesn’t need to be stored.

The set of eligible voters may vary, either over time, as new members enter or leave the organization, or depending on the type of resolution put to the vote. As a number of amendments to a resolution could be proposed, members will need several anonymous accounts to vote on each amendment using a different account. Therefore, members belonging to the set of voters register multiple anonymous accounts at once. The registered accounts are consecutively numbered. The anonymous accounts with the same number together form a group of anonymous accounts. Each time a vote with the same set of voters is conducted, the next unused group of anonymous accounts is allowed to participate[18].

When registering an anonymous account, members need to prove they belong to the respective set of voters. Instead of disclosing their identity, they create a non-interactive zero-knowledge proof called zk-snark. The zk-snark is constructed as follows. From all the hashes submitted during the activation of the member accounts, we select those corresponding to the respective set of voters. We construct a Merkle tree of the selected hashes and compute a zk-snark from the secret of the member and the Merkle proof of its hash in the Merkle tree[19]. The zk-snark is sent to a smart contract to prove that one of the leaves in the Merkle tree corresponds to the secret of the member without revealing which one it is. After verifying the zk-snark, the smart contract registers the anonymous account sent along with the zk-snark.

We must ensure that members can't take part in a vote using different accounts that are included in the respective group of anonymous accounts. To prevent them from registering more than one anonymous account in a group using the same zk-snark, the zk-snark contains a reference to the anonymous account. To prevent them from doing the same using different zk-snarks, each zk-snark is supplemented by a nullifier. The nullifier is composed of the member's secret and the consecutive number of the group. Since members have only one secret, they can only create one nullifier per group. As the smart contract knows all nullifiers that have been used for registering anonymous accounts, it won't accept them the second time.

Members can't submit the registration request for their anonymous accounts from their regular member accounts or other external accounts they own if they want to preserve their anonymity. Therefore, DecentraVote contains a relay service[20] similar to the Gas Station Network (GSN) which submits top-up request on behalf of members (see "Storage & Relay Service" on Figure 2). The service only relays transactions to smart contracts deployed by the organization[21]. In order to protect against spam, top-up requests must contain the solution of a proof of work. The relay service verifies it by hashing the solution along with the zk-snark and comparing the hash to the difficulty published on-chain. Before submitting transactions, the relay service verifies the zk-snarks locally to prevent transaction fees caused by incorrect ones.

The relay service can't replace the anonymous account, which needs to be topped up, because its address is part of the provided zk-snark which can't be re-created without knowing the member's secret. The only thing the relay service could do, is to withhold top-up requests. A malicious service could relay transactions of members it colluded with and reject requests of all others. In this case, the discriminated members submit escalation transactions[22] to inform each other and to coordinate procedures to switch to an alternative relay service of another service provider.

Casting and Counting Votes

Members can participate in a vote by sending a transaction from their member account or their anonymous account registered for that vote. The simplest form would be a transaction along with the decision of the member as a parameter. The smart contract invoked by the transaction would count continuously how many times it received which decision. However, this way everybody would see the interim result of the vote. To prevent this, the votes cast have to be encrypted and the key to decrypt them should only be revealed when the counting starts.

> Nobody knows the interim results of a vote.

[18] If the set of voters changes, already registered but unused anonymous accounts will be reused. In this case, the anonymous accounts do not need to be topped up again.

[19] The Merkle root referenced in the zk-snark makes sure that the anonymous account is registered for the corresponding set of voters.

[20] Members can use the Tor network to obfuscate their IP address when sending requests to the relay service.

[21] GSN is general purpose and can't rely on any assumptions about the smart contracts the transactions are relayed to. In order to protect from malicious smart contracts, the gas limit of relayed transactions is capped. Since on-chain verification of zk-snarks requires way more gas than allowed GNS is unsuitable for relaying top-up requests.

[22] Escalation can be performed using regular member accounts after getting several requests rejected and waiting long enough to make it impossible to associate the rejected requests to the escalating member.

Since there is only a small number of possible decisions for each resolution, e.g. yes, no and abstention or a list of the election candidates, votes can be encoded in a very simple way: we hash the numeric value representing the voter's decision salted with some keys to encrypt it. To cast a vote, the voters only need to submit the hash they computed. After reveling the salts, the decisions can be determined by the smart contract with a few hash operations. Each vote cast is double encrypted. Voters generate the first encryption key. The second encryption key is published by chairperson before the vote starts. Voters combine the chairperson's encryption key with their own encryption key and salt the hash of their decision with the combination of both keys[23]. When the vote count phase starts, they reveal their decryption key in a transaction sent from the account they used to submit the vote. When the time for revealing the voters' decryption keys expires, the chairperson publishes the second decryption key. By combining the voters' decryption key and the chairperson's decryption key, the smart contract can figure out the decision that has been hashed for each individual vote cast by comparing the hash for each possible decision value with the hash submitted by the voter during the vote.

The vote begins with a transaction of the chairperson publishing the second encryption key. The casting is concluded by another transaction of the chairperson, followed by a standstill period. During the standstill period, the user interface will disable sending new transactions, but transactions already submitted will have a chance to be confirmed. The standstill period includes an appropriate number of block confirmations to mitigate the risk of forks. After the end of the standstill period, the smart contract will reject voter transactions and allow the chairperson to publish the second decryption key. By doing this, the chairperson triggers the counting and the announcement of the outcome by the smart contract.

All transactions submitted during a vote are preserved in the blockchain and can be used to reproduce each single step and the outcome of the vote[24]. An excerpt of transaction hashes can be attached to the minutes of the general meeting.

Enforcing Resolutions

If a new director was elected, replacing a retired member of the board, the remaining directors must trigger a smart contract to change the register of directors, accordingly. If that smart contract was aware of the passed resolutions, then it would only execute changes compliant with them. To ensure this, all resolutions which effect data represented on-chain, are coded in form of a resolution contract. Such resolution contracts can only be executed if a corresponding resolution is passed. This is determined by the associated vote contract after counting the votes cast. In the course of their execution the resolution contracts perform the foreseen changes, e.g. delete a member account from and add another member account to the register of directors. The resolution contracts are referenced in the corresponding vote contracts, so that members can review the code of the resolutions before they vote in their favor or against them. The same principle can be applied to the replacement of hashes of modified user interface files and the update of rules in the smart contracts governing the organization, e.g. the number of directors or the quorum and majority needed for passing certain resolutions.

> The correct implementation of resolutions is enforced by smart contracts.

If the statutes require that passed resolutions need to be put into effect within a certain period of time, the user interface can notify members if directors haven't triggered the corresponding resolution contract once a deadline has expired.

[23] Voters compute the hash of their decision salted with vC, where v is a private key derived from the private key of their regular member account and C is a public key published by the chairperson. As soon as all voters have revealed their public key V (V=vG) and the chairperson has published the private key c (C=cG), the voters can compute the hash of the possible decision values salted with Vc for each published V.

[24] The transactions remain in the blockchain forever, but in the course of the next vote the storage used during the previous one can be deleted and thus part of the transaction fees already paid reclaimed.

About the Author

Dr. Zoltan Fazekas has been working as a consultant, speaker and author on blockchain and distributed ledger technologies for many years. Since 2007 he is heading the Austrian branch of iteratec GmbH. He is the founder of the IT service provider‘s Blockchain Labs, responsible for the blockchain education at FH Technikum Wien, the largest technical university of applied sciences in Austria, and an active member in the blockchain working group of Austrian Standards International.

Phone: +43 676 966 66 92

Email: Zoltan.Fazekas@iteratec.com

About iteratec

We are a technology company and support our customers with tailor-made solutions for complex problems – from creative brainwaves to the digital product. We are passionate technologists and it’s our goal to create added value for people with our high-quality software. We develop individual software systems, design large system landscapes and are leaders in technology. Trend-setting companies have been entrusting us with their challenging agile projects for more than 20 years.

AUTHOR

Dr. Zoltan Fazekas

iteratec GmbH

St.-Martin-Straße 114

DE - 81669 Munich

+49 89 614551-0

info@iteratec.com