Bisq DAO technical specification

This document is a detailed technical specification for the Bisq DAO and BSQ token. For a high-level overview and rationale, please see Phase Zero: A plan for bootstrapping the Bisq DAO.

BSQ token

BSQ tokens are based on Bitcoin and use the Bitcoin blockchain similar to colored coins. We don't use any existing colored coin implementation because our use case requires some extra features which are not supported by those (e.g. decentralized issuance). Besides that, we did not want to introduce any external dependencies to a company or Altcoin. As Bitcoin is the default base currency in Bisq anyway, and our requirements can be 100% covered by the basic features Bitcoin provides we decided to build our custom colored coin solution on top of the Bitcoin blockchain. We don't have the ambition to provide a general purpose solution but have designed the model according to the concrete requirements of Bisq DAO.

Technically a BSQ token is the same as Bitcoin but it adds some additional rules. A BSQ token is denominated as 100 Bitcoin satoshis so it can be divided in 100 subunits, leading to the smallest unit of 0.01 BSQ, which is equivalent to 1 satoshi. Bitcoin requires that the minimum amount of a transaction output is 546 satoshis (dust limit) so for transferring BSQ tokens we inherit that limitation. The smallest possible BSQ amount to transfer is therefore 5.46 BSQ. As BSQ tokens are inherently BTC they will have at least the market value of the Bitcoin satoshis. So 1 BSQ = 100 satoshis >= the market value of 0.00000100 BTC, which equals about 0.02 USD at a market price of 20 000 USD/BTC. The real market value of a BSQ token will be decided by the market once trading has started and will be the sum of that underlying BTC value with the value traders see in the Bisq project.

Wallet

The Bisq application provides an integrated BSQ wallet with basic features for receiving and sending BSQ as well as a transaction history screen. The wallet is based on BIP 44 (https://github.com/bitcoin/bips/blob/master/bip-0044.mediawiki) and uses registered coin type 142 (https://github.com/satoshilabs/slips/blob/master/slip-0044.md). This provides extra protection against the risk that the BSQ wallet could be used accidentally as a BTC wallet (e.g. when restoring from seed words). To avoid that users need to backup 2 different seed words we use the same seed for

both the BSQ and the BTC wallet though they are stored in different files. To avoid mistakes to mix up BSQ with normal Bitcoin we use a "B" as address prefix in the user interface. Internally that prefix does not exist, a BSQ address is a normal BTC address and the transactions are normal BTC transactions.

BSQ token transactions and balances are represented inside the application but there is also a web-based BSQ block explorer (https://explorer.bisq.network/).

Validation rules

BSQ are issued either by the genesis transaction or from an issuance transaction. We inherit all the transaction rules from Bitcoin and add some additional rules. BSQ transactions do not require OP_RETURN, though it will be used for certain specialized transactions (voting, compensation requests,...). Beside the ancestry to the genesis or an issuance transaction there is another important rule: the outputs are parsed in a way that the first outputs are interpreted as BSQ as long there is sufficient BSQ value available from the inputs. So the order of BSQ and BTC outputs is essential at the outputs! For inputs the order is irrelevant. Any violation of those rules would make BSQ invalid.

Fee payments by invalidating tokens

We use that possibility to invalidate BSQ intentionally for "burning" tokens to achieve certain use cases, like the trade fee payment, or fee payments for certain activities, like voting or compensation requests. The "burning" is only an invalidation of the BSQ value but the inherent BTC value stays intact. We use the invalidated BSQ as Bitcoin Satoshis for paying the mining fee. With giving Satoshis (BTC) to miners we are not restricted by the dust limit so we can "burn" amounts as small as 0.01 BSQ = 1 Satoshi (0.00000001 BTC).

Validation process

The validation of the BSQ tokens requires the full blockchain data from the block containing the genesis transaction up to the latest block. To avoid users from needing to download too much data we use a lite-node mode where all BSQ transactions are delivered by the dedicated block provider nodes at startup (we use the seed nodes for that but that can/should change later). The verification is done on the lite node itself (each Bisq application). Every time a new block is created by the miners, the block provider nodes (they operate as full nodes) broadcast the BSQ transactions contained in that block to the P2P network. The data is broadcasted by multiple block provider nodes, and if the data is not consistent the lite node would re-validate from the last snapshot (see below). Such inconsistency is expected in the case of chain reorganisations (https://en.bitcoin.it/wiki/Chain_Reorganization) (reorgs), but also delivers more resilience in case a block

provider node is not providing reliable data.

Furthermore we use snapshots of past blocks which will get distributed with the software release to reduce bandwidth requirements. Thus we do not require to load all BSQ transactions since the genesis transaction but only since the latest snapshot. At certain block-heights (every 100 blocks) the nodes are locally persisting snapshots. Those persisted snapshots will be used after a restart to only request the missing BSQ transactions from the blocks since the latest snapshot up to the actual chain head.

Users can also run themselves a full node either directly from the Bisq application or as a self hosted headless full node. To run a full node requires a Bitcoin Core (bitcoind) full node with RPC enabled. Here are more details on running a full node.

The block provider nodes are operated by the developers and the management for the privilege to run a default block provider node will be a part of the DAO (see phase 3). As said before the seed nodes will carry that functionality because those are the first nodes the user connects to so it improves user experience when the data gets delivered fast at start up.

Implementation phases

We will develop the Bisq DAO in several phases, starting with the bare minimum which consists of:

- Token validation
- Genesis distribution
- Trade fee payment
- BSQ transfer and trading
- Wallet integration
- Blockchain explorer
- Support for lite nodes
- Emergency measurements

After that phase we start working on **phase 2** which includes the core features for management and the periodic issuance:

- Compensation requests

- Voting on compensation requests
- Issuance of new BSQ

In **phase 3** we will focus on the implementation of the arbitration and mediation model. The security feature for locking up BSQ funds will be made available for the remaining yet not decentralized areas.

Once that is completed we can consider Bisq as **feature complete** and fully decentralized.

In **phase 4** the meritocratic approach using reputation will become a central element.

Though over time it will turn out that not all of the contributors are interested or equally talented to make the best decisions and therefore the DAO will require more sophisticated management and governance features and tools. This late phase will not be discussed in details here but it can be assumed that it will become a large field covering out-of-system tools for communication, decision making, project management, delegation (similar to Liquid Democracy), etc.

Phase 1

Genesis distribution

Technically the genesis transaction is a normal BTC transaction with the input from the donation address and outputs to all contributors. There is no requirement for an OP_RETURN output (though maybe we use one for engraving a statement to the genesis transaction).

We will use a part of the funds we received via the Bisq donation address (https://blockchain.info/address/1BVxNn3T12veSK6DgqwU4Hdn7QHcDDRag7) (about 25 BTC) for the 2.5 BTC input to the genesis transaction. Those 2.5 BTC are equivalent to 2.5M BSQ (2 500 000.00) and will be distributed to all contributors who have provided value to the project according to the value of their contribution until a certain deadline (when we publish the paper). The call for requests will be open for a period of 2 weeks.

A contribution is typically one of the following activities:

- Software development
- Communication (promotion, support,...)
- Project management
- Conceptual analysis

- Input for payment methods
- Administration
- Translation
- Design
- Usability testing
- Software testing
- Market makers
- Advice
- Others (we will decide on a case to case basis)

Basically any contributed effort exceeding roughly 4 hours will be considered to be included in the group of receivers for the initial distribution. We will announce that call for requests at the Bisq Forum (https://bisq.community/) and contributors need to send an email with the required information to enable verification if the request is justified. They should give a short description and if possible references to the work (links to GitHub, Forum, etc,...) and provide the spent time and the period when their contribution happened. We will apply a factor for giving early contributions higher weight as well as a factor to give long term contributions more weight. This should reflect the higher risk at earlier periods as well as the higher value of long term contributions. The Bisq team will verify those requests and if it is justified and the requested amount reasonable we will add the contributor to the list of receivers. The hours will get multiplied by a factor to the type of contribution (orientated on typical market salaries). We will then sum up all the weighted hours of all verified contributors and use the percentage of each contributor related to the overall sum for calculating the amount of BSQ they will receive from the genesis transaction. So if a contributor has worked 100 hours and the sum of all contributors is 10 000 hours he will receive 1% of the 2 500 000.00 BSQ from the genesis transaction, thus 25 000 BSQ.

The way how the factors are applied, how the requested amounts get adjusted and the total sum will be kept private in the team to protect privacy of the contributors as well as to avoid pointless discussions. The model for distributing the project's value is a voluntary act of the Bisq team and there is no right for a claim of any contributor as we never gave any guarantee or advertised that as a reward model. We are simply donating back our received donations to those who we think they deserve to get something in return for their support. Also the contributors can request anonymously and it is highly recommended to use GPG. This should protect the privacy of the contributors as far as possible (many will be known due their activity, but at least only the team will know that). For

market makers the verification might get a bit more difficult and we will apply a practical approach how to deal with that. They need initially provide only the onion address of their Bisq application and the number of trades they did. If we see a requirement for it there might be an extra software release where the market makers can prove their claims in a way which protects their privacy but gives cryptographic evidence of their request.

Trade fee payment

Beginning in Phase 1, trade fees can be paid in BSQ (if the user has sufficient BSQ in his wallet) or in BTC. The base fee in BTC will initially be 0.002 BTC. If BSQ is used it will be initially 2 BSQ. If the market price of BSQ is 0.0001 BSQ/BTC the BTC value of the trade fee paid in BSQ would be 0.0002 BTC which is 10% of the fee in BTC so they get a 90% discount. The fee payment is done by making a part of the BSQ invalid and give that part to miners as satoshis (BTC), thus the BTC value is not lost but used as mining fee.

- A 0.50 BSQ fee payment tx could look like this:
  - Input 1: 10.00 BSQ
  - Input 2: 0.1 BTC
  - Output 1: 9.50 BSQ
  - Output 2: 0.09950050 BTC
  - Mining fee: 0.0005 (0.00049950 BTC + 0.00000050 BTC or 0.50 BSQ)

So in that case we only use 9.50 BSQ of the 10.00 BSQ from the input. As the second output is spending more than the remaining 0.50 BSQ it is invalid as a BSQ output and we consider it as a BTC output. The remaining 0.50 BSQ which was not used in the first output will be used for the mining fee, thus reduces the mining fee which is paid from the BTC input (input 2). With that model we can spend fees as small as 0.01 BSQ or 1 Bitcoin satoshi.

The trade fee will be calculated based on the trade amount and the distance from the market price (if available). We use the same model for BTC and BSQ fees. A 1 BTC trade with 1% distance from the market price will use the default fee. If the trade amount is lower or higher we apply a linear adjustment. 0.1 BTC trade has 10% of the trade fee as long as we don’t reach the minimum value for the trade fee. For the distance to the market price we use the square root of the percent value, so 9% would result in a factor of 3. A 16% distance to the market price would cause a 4 times increase of the trade fee.

The fee is calculated according to this formula:

Math.max(Min. trade fee, Trade amount in BTC x default fee x sqrt(distance to market price in %))

BSQ transfer and trading

The BSQ can be sent and received like normal BTC. To avoid to mix up BSQ with normal BTC and risking invalidation of BSQ we use a "B" as address prefix in the user interface. So users who only operate via the UI (as recommended) cannot make mistakes here.

> 
>
> It is definitely NOT recommended to "hack" around with custom created transactions. If people are doing that they have to be sure to understand all details of the validation protocol and are fully responsible if case they accidentally burn their BSQ. This document might not cover 100% of all the details and might miss updates, only the source code is the real reference. We will not provide support for such cases and future changes might not take care of special cases used by custom transactions or implementations.

A BSQ transfer transaction is a normal BTC transaction with mixed inputs of BSQ and BTC. The BTC part is required for the mining fee payment. There is no OP_RETURN output required.

- A transaction to send 10 BSQ could look like this:
  - Input 1: 30.00 BSQ (BSQ sender)
  - Input 2: 0.01 BTC (required for mining fee)
  - Output 1: 10.00 BSQ (BSQ receiver)
  - Output 1: 20.00 BSQ (BSQ change output back to sender)
  - Output 2: 0.0095 BTC (change output)
  - Mining fee: 0.0005

Validation

The validation process of BSQ starts with the genesis transaction. The block height and transaction ID of the genesis transaction is hard coded and the application (in full node mode) starts to request the block which contains the genesis transaction from the Bitcoin Core (bitcoind) via RPC calls. It iterates all transactions until it finds the genesis transaction and adds all transaction outputs as valid BSQ outputs. From there it will iterate all following transactions and if it finds an input which is spending one of the unspent BSQ outputs it will verify the outputs to see if they are valid BSQ. The value of all BSQ outputs must not exceed the sum of all the BSQ inputs. The outputs are sorted by the index and as soon an output has used up all the available BSQ from the inputs the following

outputs are considered as BTC outputs. If OP_RETURN outputs are used there must be only one and it must be the last output. The amount at the OP_RETURN output has to be 0.

If there is BSQ value remaining but not sufficient for an output the remaining BSQ becomes invalid. This is intentionally used for the fee payments. We do not support raw MultiSig transactions (BIP 11) for BSQ. It has to be explored further in future if it is feasible to support that and if there is any need for that.

Full nodes

A fully validating BSQ node has the requirement to run a Bitcoin Core (bitcoind) node to provide the blockchain data for verification. The communication is done via RPC (https://github.com/bisq-network/exchange/blob/master/doc/rpc.md). The details about the setup can be found in the documentation folder of the source code repository. Every user can run a full node either from the Bisq application or as a specialized headless node locally or on a server and connect to that node only.

The full nodes also get a notification from Bitcoin Core at each new block, scan the block for BSQ transactions and broadcast those to the Bisq P2P network. Every transaction with any BSQ input or output (issuance) is considered as BSQ transaction. The full node also listens to network messages from lite nodes which are requesting BSQ blocks from a certain block height. The full node sends back the list of all blocks since that requested height. The bandwidth requirements for that will depend on the number of BSQ transactions but rough estimations suggest that there will be no considerable issues. The Bisq seed nodes are used as full nodes since those are the first nodes to which a user gets connected and we can use the existing connection to transmit the additional data early at startup.

Lite nodes

Most users will likely operate in the lite node mode. They have to trust the seed node operators that they are not all colluding and delivering incorrect data. If at least one operator is honest the lite node can detect a conflict and would re-validate each block from the last snapshot. The UI will notify the user about conflicting data from seed nodes.

A lite node requests at startup from the seed node the missing BSQ blocks and then validates those blocks to achieve a local state of valid and unspent BSQ outputs. At each new block they receive the broadcasted messages from multiple seed nodes (min. 4 operated by different developers) and only if all those messages contain the same data the validation will succeed and the block will be added to the local state. In case of chain splits it can be that one of the seed nodes is on another chain and conflicting blocks get propagated. This would trigger a re-validation of all blocks from the

latest snapshot for the lite node. The last received block would be considered as the current state but the user get displayed a message that there are conflicts and it is recommended to wait for more than one confirmation before considering a BSQ transaction as valid. Only after all full nodes (seed nodes) have the same state again the lite node will exit the "warning" state. If the user waits for a sufficiently high numbers of confirmation (4-6) he will not risk that his validation was based on an orphaned chain and that he could become victim of a double spend.

Snapshots

Every 100 blocks a snapshot mechanism gets triggered. The current state get cloned and kept in memory and if a previous clone exists the previous one will be persisted. At the next snapshot trigger event the latest clone will be persisted and a new clone will be cached again. That way the snapshot is always at least 100 blocks old.

The lite node requests the blocks since the latest snapshot only, so that will be usually max. 200 blocks. Just at the first startup when the lite node has only the snapshot shipped with the binary the requested blocks might consume a bit more bandwidth.

If we have monthly releases there would be about 4500 blocks in one months but even with that we expect not more than 1-5 MB of bandwidth to receive the initial blockchain data.

Phase 2

In phase 2 we introduce the periodic voting and issuance cycle.

Periods are defined in block height. Each period is separated with a break of 10 blocks to avoid issues with reorgs.

- Publishing compensation requests (3630 blocks, about 25 days)
- Voting: Approve/decline compensation requests (450 blocks, about 3 days)
- Voting commitment: The voters publish the decryption key and vote on their vote data consensus (300 blocks, about 2 days)
- Issuance of new BSQ (happens directly and automatically after the vote commitment is completed)

The full cycle will last 4380 blocks which is about an average month if one block takes in average 10 min. The interval of 1 month has been used in the phase zero and can be considered as practical.

Compensation request

Contributors can create a compensation request for the work they contributed to the project. This can be anything what has added value to the project. The contributors have no guarantee that their request gets accepted. So when they start working they need to be aware that there is no guarantee for a reward.

If not sure about the value of their work for the community, they should make small work packages and discuss at the usual communication channels (Keybase, GitHub, Forum,..) to see if the work they are proposing sparks some interest and support. To use upfront payment with escrow would make the process much more complicated (who controls the escrow,...). It also reflects the situation of normal freelance work where work is paid usually after the work is completed and the reputation of the company provides sufficient base for a trust relationship in most cases.

To make a request, a contributor must include enough BTC to issue the BSQ he's requesting (amount requested * 100 satoshis), and pay a 1 BSQ fee to discourage spam. See example compensation tx for 5000 BSQ below.

There will be a user interface in the application where the contributor fills in a form with the required data.

The contributor will publish the request to the P2P network after the fee tx is confirmed with 6 confirmations in the blockchain (6 confirmations to avoid issues with reorgs and tx malleability). The publishing of the compensation request can be done any time during the contribution request phase. A contributor can file several requests for different work packages. Any compensation request published after the first phase has ended (once the break starts) will get queued up for the next phase. Each node will verify the compensation request if it fulfills the rules and only forward valid requests. The UI will display own requests, the active requests of others as well as a history of all past requests.

The range for allowed amounts for a compensation request payout will be 50 BSQ to 50 000 BSQ.

- A compensation request needs to contain following data
  - UID (auto generated unique ID)
  - Contributor's name or nickname
  - Title (must not conflict with existing requests)
  - Creation date

    - Description (short paragraph)
    - Link to either GitHub issues or Bisq Forum for detailed description and deliveries
    - Requested amount in BSQ
    - BSQ Address
    - Tx ID
    - Contributor’s Public key
    - Version
- Data structure of the OP_RETURN compensation request data
    - 1 byte for type (0x01)
    - 1 byte for version (0x01)
    - 20 bytes for hash of payload (using Sha256Ripemd160 from Protobuffer encoded payload)
- Verification rules for compensation request transactions
    - There have to be one OP_RETURN output as last output
    - The amount at the OP_RETURN output has to be 0
    - The first byte in the OP_RETURN data needs to be the type byte: 0x01
    - The second byte in the OP_RETURN data needs to match the nodes version byte: 0x01 (requests made with older versions are invalid)
    - Size of OP_RETURN data is 22 bytes
    - There has to be a BSQ input for the fee payment
    - BSQ used for fee need to be mature
    - The fee needs to match the fee defined for that cycle (can be changed by voting at each new cycle)
    - The block height must be in the correct period
    - It needs to have at least one output to the address defined in the compensation request data

Contributors need to have the latest version installed when doing a request to be sure to have the same version as the verification nodes.

- A compensation request tx for requesting 5000 BSQ would look like this (fee is 1 BSQ):

- Input 1: 30.00 BSQ (needed for fee payment)
- Input 2: 0.1 BTC (needed for mining fee; we also need a BTC output)
- Output 1: 29.00 BSQ (mandatory change output)
- Output 2: 0.00500000 BTC (requested BSQ amount * 100 satoshis goes to BSQ address defined in request)
- Output 3: 0.09450100 BTC (optional BTC change output)
- Output 4 (last): OP_RETURN data as defined above
- Mining fee: 0.00050000 (0.00049900 BTC from input 2 + 0.00000100 BTC or 1 BSQ from input 1)

The input 1 needs to be larger than the fee so we enforce a BSQ change output (output 1). All outputs must not be smaller than the dust limit (2730 Satoshi). We require that the BSQ change is at input 0 and mandatory to have a clearly defined output index for the issuance output. The BSQ change output cannot be after the issuance output as that is interpreted as BTC as long it got not successfully voted. The BTC input at input 2 needs to be at least the sum of the requested BSQ and the miner fee, in our case 0.00500000 BTC (requested BSQ) + 0.00049900 BTC miner fee. Please note that the output 2 is at request time interpreted as BTC. **Only after the request gets accepted by voting does the output get interpreted as BSQ and thus the requester has issued himself BSQ.**

Voting

To make the best decisions require a certain level of information and time. Voting in the DAO is an important service and should be only executed by those who are well informed and take sufficiently time to make well reasoned decisions. Therefore there will be a considerable fee for voting to de-incentivize stakeholders who are not sufficiently interested in the project. The fee will be set to 5 BSQ. The stakeholder can vote on a single vote item or on as many as they want.

In the vote period a stakeholder cannot transfer his BSQ tokens which they used for voting, otherwise they would render their vote invalid. For that reason we should keep the vote period rather short to not lock up liquidity for too long. There might be an effect on the market price as if many stakeholder are using their coins for voting there will be less supply and therefore increase the price. Though that effect should be limited as it is predictable and known in advance and it lasts just 5 days and the loss of the vote would also be not too problematic for some stakeholders, if they decide to prefer to trade their tokens instead.

All valid compensation requests from the current cycle are considered for voting. The stakeholder can choose to accept, decline or ignore a request. For acceptance or decline a simple majority is sufficient (> 50%).

Initially the voting is mainly for the compensation requests but there will be some flexible (yet to defined) option for voting on any topic. Over time we might add more specific vote items like amount of trading fee. To avoid that some stakeholders take benefit of voter apathy and are able to make changes with a very low stake we require a quorum for each vote item. Those quorum values will be defined for each vote item. If the vote item does not reach that limit it will be discarded.

We use blind voting to avoid influence of the current state of the votes to voters who have not yet voted. Without blind voting there would be an incentive to wait for the last moment with voting to have more information.

The voting will take place in 2 phases. The actual voting phase which lasts about 3 days and the decryption reveal phase which takes 2 days.

The voting weight is derived from the amount of the BSQ change output. The user can define with which BSQ amount he wants to vote.

Blind voting phase

The voter encrypts with an encryption key (AES) created per vote his vote data and puts the hash of the encrypted data in the OP_RETURN of the vote Tx.

The encrypted vote data are broadcasted to the P2P network. To avoid an attack scenario where the malicious voter could try to disrupt the consensus of received vote data by broadcasting their vote data to the P2P network at the very end of the period, thus it has higher chances to not arrives equally at all peers we can use a random break at each voter which makes that attack less effective.

- Data structure of the OP_RETURN vote data:
  - 1 byte for type (0x02)
  - 1 byte for version (0x01)
  - 20 bytes for hash of encrypted vote data (using Sha256Ripemd160, proposals are sorted by txId, data input for encryption is byte array of Protobuffer file of list of proposals)
- A vote transaction would look like that (fee is 5 BSQ, stake is 200 BSQ):
  - Input 1: 300.00 BSQ (needed for fee payment)

    - Input 2: 0.1 BTC (needed for mining fee)
    - Output 1: 200.00 BSQ (stake)
    - Output 2: 95.00 BSQ (optional remaining BSQ change output)
    - Output 3: 0.09955 BTC (optional BTC change output)
    - Output 4 (last): OP_RETURN data as defined above
    - Mining fee: 0.00050000 (0.00045000 BTC from input 2 + 0.00005000 BTC or 5 BSQ from input 1)
- Verification rules for the voting transaction
    - There have to be one OP_RETURN output as last output
    - The amount at the OP_RETURN output has to be 0
    - The first byte in the OP_RETURN data needs to be the: 0x02 (type)
    - The second byte in the OP_RETURN data needs to match the nodes version byte: 0x01 (votes made with older versions are invalid)
    - Size of OP_RETURN data needs to be 22 bytes
    - There have to be a BSQ input for the fee payment
    - BSQ used for fee need to be mature
    - There have to be exactly 1 BSQ output for the voting weight
    - The fee needs to match the fee defined for that cycle (can be changed by voting at each new cycle)
    - The block height must be in the correct period

Contributors need to have the latest version installed when participating in voting to be sure to have the same version as the verification nodes.

Vote reveal phase

After the 3 days period for voting is over the voters need to make a new transaction which will reveal their decryption key so that the vote data become readable as well they will vote on their data view of which vote data they have received from the P2P network. As the P2P network comes with eventually consistency there is no guarantee that all vote data arrive at all peers. For calculating the vote result all peers need to have the same collection of vote data to get the same result. To achieve that the voters will create a sorted list (sorted by hash of data) of vote data and

create a hash of that collection. That hash will be put together with the decryption key into the OP_RETURN data of the reveal transaction. If there are conflicting vote data views (some voters did not receive all votes) the majority will be considered valid and the votes from the others will get ignored for calculation of the vote result.

The input for that transaction must be the BSQ output from the vote transaction.

- Data structure of the OP_RETURN vote reveal data:
  - 1 byte for type (0x03)
  - 1 byte for version (0x01)
  - 20 bytes for hash of (encrypted) vote data collection (using Sha256Ripemd160)
  - 16 bytes for decryption key (AES 128 bit)
- A vote reveal transaction would look like that:
  - Input 1: 25.00 BSQ (output 1 from previous vote tx)
  - Input 2: 0.1 BTC (needed for mining fee)
  - Output 1: 25.00 BSQ (transfer to voter)
  - Output 2: 0.0995 BTC (optional BTC change output)
  - Output 3 (last): OP_RETURN data as defined above
  - Mining fee: 0.0005 BTC

Calculate the voting result

After the vote reveal phase is over all Bisq users will calculate the vote result.

The user might also have a different vote data collection than voters. To get a consensus about a unique view of the vote data we look for the majority winner from the vote reveal transactions. We gather all valid reveal transactions and add up the BSQ inputs to find the winning vote data collection. In rare case we would have 2 compensation requests collections with the same BSQ stake we would use the one where the hash converted to a double number results in the smaller number. If that hash of the winning data collection matches to our own data collection we go on with the calculation, if not we need to request the missing data from our peers.

Next we decrypt the vote data with the corresponding decryption key. The vote transaction contains the hash of the vote data so we can assign that to our encrypted P2P network vote data. The reveal transaction has as input the BSQ output of the vote transaction and contains the decryption key, so

we can use that to decrypt the vote data.

We sum up all vote data items and use the BSQ amount as weight to get a total result.

Issuance of new BSQ

After the vote reveal period and the following break has ended all the compensation requests which have received >= 50% of the acceptance votes (compared to declined votes) will become valid for issuance of new BSQ. The second output of the compensation request transaction which has been interpreted as BTC so far will not be interpreted as valid BSQ, authorized due the voting process.

- Verification rules for the issuance transaction
  - The BSQ output is equal to that what has been defined in the compensation request
  - The issuance amount needs to be in the range of the min. and max. allowed amount
  - The block height must have been in the correct compensation request period
  - The compensation request needs to be accepted in the voting process

Scenarios for gaming the voting process

If a voter would not broadcast his vote data to the P2P network or sends it out of channel to selected voting peers he has very little chances that his vote will be in the majority data view and thus renders his vote invalid.

If a voter would not forward received vote data from other peers, he cannot prevent that the vote data gets distributed by other honest voters as long the P2P network is not partitioned.

A voter could try to broadcast at the very end of the period to increase the chance that some peers will receive his data before the deadline and some after the deadline, thus they would ignore his data and that would render different data views. This can be mitigated if we use slightly different random time for the break so he cannot know which peer has. (Credit to Eyal Ron for that attack risk and mitigation solution).

As long as the majority of voters are not colluding and are honest the scheme is secure against manipulation.

Phase 3

Mediation and arbitration system

As discussed in the Arbitration and Mediation System document

(https://docs.google.com/document/d/1DXEVEfk4x1qN6QgIcb2PjZwU4m7W6ib49wCdktMMjLw/edit#) we will split the dispute process into mediation and arbitration.

Requirements for locked up BSQ funds are initially set to 1000 BSQ for a mediator and 20000 BSQ for an arbitrator but can be adjusted by voting. At registration the lockup transaction requires 6 confirmations in the blockchain before it is considered valid.

Both need to fulfill basic requirements (availability, quality of work,...). If they would fail on those they would risk that the locked up funds (or part of it) get confiscated. Mediators can use external tools for building up reputation. Links to a webpage or services like Bitrated (https://www.bitrated.com) can provide such a bridge. An application internal reputation system for mediators and arbitrators might be implemented as well over time but is not planned initially.

Lockup process

To register as mediator or arbitrator one needs to send the required amount of BSQ to an own BSQ address. This special transaction contains OP_RETURN data which are marking that transaction as lockup transaction (OP_RETURN type 0x04). Any spend transaction from this address would render the BSQ invalid as the only valid process to unlock those funds is to use the unlock transaction.

Unlock process

To unlock the funds he makes another transaction to himself with other OP_RETURN data (OP_RETURN type 0x05) which marks that transaction as an unlock request and will become available for spending after the lock time is over. The unlocking period is about 2 months (9000 blocks). The delay for unlocking is required to give the community enough time to act in case of abuse to prepare the steps for a possible confiscation. Therefore the lock period needs to be rather long.

Confiscation

In case a mediator or arbitrator fails (fraud or severe failure in fulfilling the requirements) anyone can make a request for confiscating the locked up funds. This request will have a high fee (100 BSQ) to avoid abuse. It will require a very high quorum (100 000 BSQ) and percentage (75%) of acceptance in the voting process to make sure that this confiscation process will not be abused.

A partial confiscation is also possible. The confiscation will be rolled out as a new release where the confiscated transaction is hardcoded and renders the locked up BSQ invalid.

By using a software update we add another safety factor to avoid abuse (if users don't agree they can simply ignore the update), so users are voting to support the decision for confiscation by

updating the software. If there is not a super majority it would lead to a network fork. These hard requirements should make sure that only non-contentious cases can be considered for confiscation.

Revocation

Revoking a registration requires some lead time, because the arbitrator or mediator can be used in trades or disputes which require some time to get completed. The lead time will be 2 weeks (2000 blocks).

Offers which will get taken after his revocation can only be taken if other arbitrators are selected in the offer as well. In the worst case an offer which has only selected a revoked arbitrator becomes invalid which will get communicated to the user so he can remove the offer. That should be a very rare case if multiple arbitrators are available.

The number of mediators and arbitrators can be influenced by voting by setting the requirements and payments higher or lower. A change of the requirements will not be applied to past registrations. The requirement at registration time will stick the lifetime of a mediator or arbitrator.

Arbitrators and mediators get paid like any other contributor via compensation requests. They payment will be adjusted to lead to a healthy amount of arbitrators and mediators.

Other use cases for locked up funds

There are a few other areas where we will use the same model with locked up BSQ funds to achieve the security required to open and decentralize those. Additionally there will be a voting process as those privileges are usually taken by main contributors, so reputation will play an important role beside the requirement for locked up BSQ funds.

Infrastructure

- Seed nodes (they provide also the BSQ transactions for lite nodes)
- Market price feed provider node: BitcoinAverage price requires a API key and a monthly fee payment. Users can use their own node but then they need to acquire an API key from BitcoinAverage.

All the nodes can be overridden by program arguments, so the user can connect to self hosted nodes. To get the privilege to run one of the default nodes (hard-coded onion address) it requires to lock up BSQ funds and to get accepted in the voting process.

Privileged messages

There are a few P2P network messages which require a private key (public key for verification is hard-coded) to broadcast them. They are mainly in place for emergency cases to be able to limit damage or to fix problems. Only the update message is used on a regular base. - Send out an application update message - Send out an alert message - Send a private message to a particular node - Ban offers by the peers onion address, offer ID, specific payment account data like name, IBAN,...

All those messages can be ignored by the user when he sets a program argument (in case of abuse by the key holder the users can go that route and the messages will be ignored and have no effect).

To get the privilege to control a private key for one of those messages it requires to lock up BSQ funds and to get accepted in the voting process.

Accounts

- GitHub account
- Bisq domain
- Bisq Trademark
- Social media accounts (Twitter, Reddit, Keybase, IRC, Facebook, Telegram, Mailing List, Newsletter)

Most of the social media accounts will be operated by community members. The number of "official" Bisq accounts will be low.

On Github we will use a similar ACK/NACK commitment model like it is used in the Bitcoin Core development process. To receive the ACK/NACK privilege will require locked up BSQ funds and to get accepted in the voting process. Same applies for domain and trademark ownership.

Deployment of the app installer

The application installer is built and signed by the main developers. Any user can run from source code as well. Again we will use the same model as above for giving the privilege to sign a binary.

Anyone who locked up BSQ for getting one of those privileges will get paid as a contributor for that service.

Until those features are implemented the project founder and the Bisq foundation will serve as a trusted host for of those areas.

Phase 4

Reputation based voting

As stated earlier the project should shift the weight for decision making from pure stake based to a mixed model where reputation will get a higher weight (target is 70% but will be decided by voting of the stakeholders).

Phase 5

Further governance and management tools

It can be assumed that there will be requirements for further improvements of the management and governance structure and features. We see it as an open work in progress to try to find the best model and tools to achieve the best results. Tools for communication, decision making, project management, delegation and more might evolve over time. Many of those tools might be provided out of system from other platforms.

Security measurements

To limit risk and possible damage in cases of bugs or exploits we will use several measurements.

Maturity

The newly issued tokens (not genesis tokens) have a maturity period of 1 week (1000 blocks). During that period they cannot be used for trading (the buyer would not accept them as they are marked as immature). This maturity period will give more time for reacting in emergency cases.

Limitation of growth of the total supply of BSQ tokens per month

The total supply of BSQ tokens will be limited by blockchain height. Initially there will be 2 500 000 BSQ from the genesis transaction. We don't expect more than 100 000 new BSQ being issued per month. So we use that for the max. monthly growth. This numbers can be adjusted at each release, so he can adopt to the market price. In case of an exploit where the hacker manages to create new BSQ the max. possible damage would be limited by that value. Any BSQ which have been created after exceeding that limit would be considered invalid.

Private key for activating emergency measurements

There will be a private key (similar like the other private keys for privileged P2P network messages) for sending out an emergency message to all nodes for deactivating BSQ trade. BSQ tokens are

traded only in Bisq. We don’t expect that other exchanges will support BSQ soon as it would require quite a bit of effort for them to support the protocol.

There will be another emergency message for disabling new issuance of tokens. Like with the other privileged P2P network messages the users can ignore those emergency messages by a program argument (in case that the key holder would abuse their power), though in case of a hack users who have ignored those messages would not get considered in a possible compensation program for recovering the losses.

Predefined policy how to deal with unexpected situations

In case of bugs which would cause the loss of BSQ there will be a reimbursement for the victim by issuing new tokens using the compensation request and voting process (the victim files a compensation request and if accepted by voting can issue themselves the lost BSQ tokens). It requires clear evidence and cooperation of the victim. The lost BSQ ("burned") have been taken out of circulation and by issuing new tokens we add them again, so we do not inflate the total supply by such a measurement.

Another case would be if tokens get issued by an exploit or hack. They will get confiscated if it is possible (if they have not been already traded and ownership is not 100% clear anymore). A hard fork adding code to declare certain transactions invalid would be deployed in such a case.

To avoid later discussions about "code is law" we define with that policy clearly that in case of a clear violation to the intended behavior of the DAO we will try to fix it as far it is possible. Confiscation and new issuance are valid tools to achieve that. The network effect and fork risk are in place to avoid any abuse of those emergency measurements.

Definitions

Some terms are used in different context. The following should make the distinction of their meaning clearer.

Compensation request

We refer to that term as the request from the user perspective in a conceptual sense.

Compensation request transaction

This is the Bitcoin transaction which will turn into new issuance transaction once the compensation request got accepted in voting.

Compensation request data

This is the data structure published to the P2P network when creating a compensation request. It gets created when the user fills in a form in the application and confirms to submit a compensation request.

Voting

We refer to that term as the voting activity from the user perspective in a conceptual sense.

Vote transaction

This is the Bitcoin transaction which contains the hash of the encrypted voting data.

Vote reveal transaction

This is the Bitcoin transaction which contains the hash of the vote data view and the decryption key.

Voting data

This is the data structure published to the P2P network when submitting a vote. It gets created when the user sets his voting options in the UI and confirms to submit the vote. It is encrypted and only becomes readable once the voter reveals the decryption key in the vote reveal transaction.

Version unspecified
Last updated 2021-04-03 12:50:57 UTC