Off-chain trade protocol

Privacy considerations

The traders reveal the number of total trades to their peer but not the trade amounts. The amount of all pending trades (bond used) is revealed to the peer for verifying if the bond is not overused. Only in case of parallel trades the user reveals the trade amount of his other trade(s).

We use a blinded contract which holds only the hash of the real contract and the trade amount which is used for summing up the used bond of multiple pending trades.

The bond Tx ID is blinded in the public data by using a salt.

For improved privacy users can make new bonds and assuming they do not link the bond txs they avoid that the number of trades get revealed in new trades with the new bond.

Verification

This doc does not cover yet all verifications but tried to keep focus on the basic concept. Guarantees for the bond witness data to be published and available is depending that the peer is not colluding in holding back witness data. This is considered low risk as the trader does not gain anything but risks that in case of contract breach the peer has not enough bonds and thus reduces his own security. If it can be proven the mediator might consider that as a severe breach leading to confiscation of his own bond. The decreaseWitness is published by the peer as it is in his interest. The publishing of the increaseWitness is in the control of a trader himself as he is himself interested to get his available bond amount increased again.

P2P network

We depend on a reliable distribution of the witness data. As the P2P network has only eventually consistency we have to deal with the valid cases that some data might be missing. We cannot interpret missing data as the peers cheat attempts. This topic requires more thoughts how to deal with such situations. The trader has incentive to store his own increaseWitness objects and could deliver that data if missing in the network. But for the decreaseWitness data there would be an incentive to hold back the data.

Lock up a bond

The user needs to make a one time BSQ tx to lock up a certain amount of bond.

Bond

- txId
- salt
- id = hash(bondTxId + salt)
- EC pubKey (taken from blockchain data at BSQ tx input)

With the salt the tx id does not get public. The salt is exchanged with trader so the trader will learn about the bond tx but that is required anyway for verification of the bond.

Trade

When users enter a trade they create a contract holding all relevant data.

Contract

- Alice bond Tx ID
- Bob bond Tx ID
- Alice trade obligation (e.g. give 900 EUR)
- Bob trade obligation (e.g. give 1000 USD)
- Dispute resolution (e.g. mediator XYZ will be the judge in the case of a dispute)
- Alice pubKey
- Bob pubKey

BlindedContract

Blinded contract is used to reveal a used bond in parallel trades. It is only exchanged between traders and only those blindedContracts from pending trades.

BlindedContract

- amount

- hashOfContract

When starting a trade the traders exchange their bond salt.

Decrease a bond

Alice requests and verifies Bob's signature of the decreaseBondWitnessId (signed with his EC key from his bond tx). After she received the signature she publishes the DecreaseBondWitness to signal Bob's bond got decreased. Bob persists the DecreaseBondWitness locally to make it available if requested.

DecreaseBondWitness
- Id (Bob's bondId + WitnessId + flag indicating bond decrease)
- Bob's bondId
- WitnessId =hash(blindedContract)
- Bobs signature
- Alice trade pubKey

She broadcasts that data to the P2P network. Bob is doing the same on his side.
As Bob would have an incentive to hold back the DecreaseBondWitness we cannot rely on his local persistence but need to ensure the whole network stores that data.
If the data would not be available we still might get Bobs BlindedContracts in the next step. He cannot hold it back as he cannot know if the data have been available for Alice or not, and by holding it back he would risk a contract breach. In the worst case another parallel trade was not detected and Bob might overuse his bond. As BSQ volatility also might lead to a situation where the trade amount is not 100% covered, we could tolerate that case as well.

Alice would not start sending funds if Bob has not sent her the signature so that she can publish the DecreaseBondWitness for Bobs' bond.

Exchanging BlindedContracts of pending trades

Alice requests from Bob the list of the BlindedContracts for his pending trades as well as all his IncreaseBondWitness objects. Bob does the same with Alice.

They filter all DecreaseBondWitness objects from the p2p data storage for matching bondId of peer.

- ListOfDecreaseBondWitness

Filtering all IncreaseBondWitness objects for matching bondId of peer.

- ListOfIncreaseBondWitness

We check for those DecreaseBondWitness objects which do not have a matching IncreaseBondWitness (pending trades) and check if that resulting list matches the list of the BlindedContracts which we got delivered by the peer.
If correct we can calculate with the list of the BlindedContracts how much of the bond is available and if the trade is not exceeding the bond.

Alice verifies the DecreaseBondWitness objects of Bob's bond by verifying the signature with the EC pubKey from Bob's bond Tx.

Increase a bond

When a trader confirms receipt of the expected trade funds he sends also a signature of the increaseBondWitnessId using his EC key from his bond tx to the peer.

Alice has received the funds from Bob and she sends him her signature.
Bob persists locally the IncreaseBondWitness and provide it to requests in following trades:

- Id (WitnessId + bondId + flag indicating bond increase)
- WitnessId (hashOfBlindedContract)
- Bob's bondId
- Alice's signature using her trade privKey

We use the pubKey from the corresponding DecreaseBondWitness object to verify the signature.
We do not need to broadcast the data as it is in Bob's interest to make that data available to trade peers to proof that a past trade has been completed and a bond increased again.

General comments (by BoW):

* Consider simplifying to single-ongoing-trade-per-bond while still allowing for bond reuse over time. Maybe that is the 80/20 sweet spot?
* Already mentioned in the private chat, attack by fake-trading with self to get rid off the bond (via confiscation + reimbursement) that was also a backing for a real trade; then scamming the real trade.
* I guess we want to support multiple bonds per user? Then, can I back a single big trade by several small bonds? We can skip that initially but we may want to plan for extensibility.
* Overall, the cryptosystem we are designing here feels scary :O (not that I have any better idea...)