Crypto Family Bridge

Edy Haddad haddadedy82@gmail.com

April 17 2021

With the release of the Ethereum blockchain in 2015 and the introduction of smart contract functionality, many new blockchains with such functionality started following, all of which offered great opportunities, development environments, decentralization, and freedom of use. And while these blockchains remain fully decentralized and open for everyone, the communication between them remains centralized and lacking. The Crypto Family Bridge solves that issue by offering a fully decentralized bridge to link all web3 blockchains such and layer 2 protocols as Ethereum, BSC, Polygon, xDAI and many others, all through a fully decentralized and secure bridge. The bridge is run in a decentralized form by nodes that use FT (Family Token) to achieve PoVP 'Proof of Voting Power' to help secure all of the bridge's transactions

Introduction

Smart contract blockchains offer a great opportunity for decentralized security and finance, with simplicity, and ease of use. And while there are some blockchains who are accomplishing wonders in DEFI such as Ethereum and BSC, such blockchains lack the means of secure decentralized communication between one another, thus the absence of inter blockchain communication between the two, and such lack of communication between the two creates two separate worlds with barely any communication between them, and such lack of communication eats away a lot of the work the community and the developers can achieve using inter blockchain communication.

Centralized inter blockchain communication is an imperfect method of communication between blockchains as it lacks security and freedom of use and is limited to certain tokens and use cases, and is subject to down time.

The inter blockchain communication that the Crypto Family Bridge offers solves many of these issues and offers many opportunities. The opportunities and features the bridge offers can be put as following :

1. A global DEFI economy
2. Allows integration of projects from one blockchain to another

3. Allows transfer of any and all tokens and coins from one blockchain to another
4. Easy to use, secure and decentralized bridge preventing any downtime, cheating of transactions, or preventing transactions

How it works (Overview)

The bridge is composed of three key elements, the bridge smart contract found on the blockchains we want to link, the nodes that will read the transaction requests and process it, the FT (Family Token) that will be used on the blockchains to achieve PoVP 'Proof of Voting Power' and to pay transaction fees to the nodes

The bridge smart contract :

For a blockchain to join the bridge it needs the bridge's smart contract deployed on it. This contract will be the one the nodes communicate with to burn and mint tokens from one blockchain to another, this contract also manages the PoVP and voting, holds the tokens and their equivalent on the other chain, manages the joining and leaving of the nodes, manages the staking and unstaking of FT, and manages punishments

The nodes :

The nodes take care of carrying the transaction requests from one blockchain to another (transfer and addToken transactions), they are what manages the minting and burning of tokens on the blockchains through automatic voting.

FT 'Family Token'

FT is the token to be used to manage voting power on the bridge, nodes will need to own a certain amount of FT on the blockchain pairs they wish to link, for example if a node wishes to forward orders and requests from Ethereum to BSC they would need to run a node linked to the bridge's smart contract on BSC and vice versa, so they would have voting power over the order made on the BSC.
A bridge link between two blockchains can handle 42 nodes, 21 nodes forwarding and voting on each blockchain.
Note : Creation of transfer requests, add token requests are all managed by the smart contract, nodes have no authority over such actions and cannot stop them or change them, they can only read them

How it works (Detailed)

Nodes, Staking And Unstaking

Before a node can join and cast votes on the bridge, they need to stake FT on the smart contract to get allowance from the contract to vote on transactions. Every bridge's smart contract has room for 21 nodes, if all these positions are filled new nodes who wish to join will have to stake more than the least

staking node on the bridge to take their place. After a node send their FT for staking on the Bridge's contract they are immediately burnt using the built in burn function in the FT smart contract, and later when withdrawing are minted with the fees the node made .

When a node wishes to leave or unstake a small portion of their stake, they will be able to do that instantly, their stake will be immediately removed from the bridge and their voting power reduced or taken away, however a lock period of 7 days is placed on their funds, after the 7 days lock period is over the node operator can come back to manually claim their locked FT.

Transactions And Voting

Voting and votes manage the authenticity of all kinds of transactions on the bridge (transfers and tokens adding).

Stages of voting :

.Nodes first need to capture the data of the transaction they want to forward from the opposite blockchain(further explained in nodes), in case of a transfer transaction that data includes whom it's sent to, amount, fee, claiming fee, token address, and index (aka nonce), in case of a token adding transaction data includes owner, main contract address, parallel contract address, fee, index(nonce).

.After that data is captured the nodes use it in its respective function on the smart contract, data is placed in and transaction forwarded

.After the node has forwarded the transaction to the smart contract the smart contract his vote will be casted for this transaction

Note : A node cannot vote twice

Note : A transfers authenticity is defined by the voting power of the nodes voting for it and not by the amount of nodes

.After creating or voting on an order the smart contract records the FT of that node (their voting power), if the order has 50% or more than the total staked FT, the smart contract will run a scan, it will check the nodes who voted and how much their current voting power is, and make a final calculation of how much votes that order currently has, if that total amount is greater than or equal to 50% of the voting power then that order is signed and eligible for claiming, this action is triggered by the last node who votes (or in rare cases where the node who makes the order had 50% or more of the voting power), so that node is also now the signing node of the transaction, his voting transaction also takes care of distributing the penalty FT back to the voters and creator of the transaction as well as distributing the fee among them (The fee distribution structure is written in the smart contract) (signing node gets more since it uses more gas to achieve the extra functions of signing and redistributing penalty fees and transaction fees)

Signing, Claiming And Claiming Fees

Once a order has enough voting power backing it, it is ready to be signed and claimed, when a claiming node sends a request to claim the order for the user and earn the claiming fee, the smart contract will run a scan, it will check the nodes who voted and how much their current voting power is, and make a final calculation of how much votes that order currently has, if that total amount is greater than or equal to 50% of the voting power then that order is signed and eligible for claiming, now the question here is why does the user have to claim his order, why doesn't the bridge automatically claim it for him? The reason here is that the bridge is open and anyone can link any custom token smart contract to it, meaning that this could be used to build custom tokens with high gas fees on transfers and minting, and this could be used to attack the nodes and drain their funds.

To prevent such attacks the claiming of the fees was made separate from the nodes. A new fee and a new type of nodes was added. The claiming nodes are a software anyone can run without the need to stake any FT, these nodes looks for any eligible orders, checks the claiming fee, calculates the gas it

would take to claim that transaction, compares the gas expenses with the price of the coin compared with the claiming fee and the price of FT, if the claiming fee on those orders is enough to pay back the node their expenses and make them profit then it will be claimed.
Note : claiming doesn't require voting as the order is already voted on by the nodes, and the first claiming node to claim it gets the claiming fee
Note : An order can be claimed once, after that the smart contract will save it as claimed, and can't be claimed again

How does the user receive their funds after their order is claimed?
When a users order is claimed the smart contract will check if the address of the token the user is receiving is saved as parallel or original on the smart contract, if the token is labelled as parallel then the user has sent tokens from the original contract to the parallel one, so the contract executes the mint function and mints the user new tokens, otherwise if the token's address is saved as original this means the user has burned parallel tokens and is trying to get back the original token, in that case the smart contract will transfer the user his tokens.

Transferring Tokens
Transferring tokens on the bridge is simple, but has its requirements.
.The token has to be supported by the bridge, if a user tries to make a transfer of a token that is not labelled as original or parallel with a parallel or original contract on the opposite chain then the users transfer request will be refused and transaction reverted.
.FT to pay for fees, security and fees on the bridge are handled in FT so a user needs to have them to pay for his transaction fees.
.Give the bridge's smart contract allowance over the funds they want to transfer

After the user has met the requirements the smart contract will take the funds the user wants to transfer and writes down a send order for the nodes to pick up and forward to the opposite blockchain

Adding Tokens And Solution Contracts
So what's the difference between adding tokens and using a solution contract to add tokens, well the difference is that the first can only be done by the owner of the contract and the second can be done by anyone who wishes to add a token they have no ownership on.

If the token's smart contract has the getOwner() function in it, then the owner of the token can automatically execute the addToken function to get his token added to the bridge and his request will be forwarded across the bridge simply like that of a transfer request, while adding his token, the owner has to provide a second smart contract on the opposite blockchain with the getOwner() function and his address as its return variable, that contract should also have given permission to the bridge to mint and burn tokens on it. Once the owner has forwarded his transaction the first chain will write down the original contracts address with its parallel and the opposite chain will write down the parallel chain contract with its original

Note : If the owner fail to meet the requirements his token might be added to the bridge but his transactions would fail, so be careful and test after you add your tokens. An owner can always make another request.
If the token's smart contract doesn't have the getOwner() function or a user wishes to see a token he doesn't have ownership over added to the bridge, they can always use a solution contract, a solution contract is a smart contract that resembles the token, every solution contract can be linked to one and only one token where that smart contract can accept transfers from that token and has given the bridge full allowance over it's funds, the user who creates that solution contract will also be the owner of it. After creating the solution contract the process of adding it to the bridge is similar to that of a regular token

Note : A solution contract only contains three functions, getOwner(), transfer(), and interBlockchainTransfer(), each designed to serve the token supported by the contract and not the contract, the solution contract can't be used to hold tokens on, it is only used to forward transactions and hold forwarded tokens, the transfer() function doesn't change the users balance on the contract but forwards them tokens on the token being supported, the only time the contract holds token is when a user uses the interBlockchainTransfer() function, in that case the contract holds those tokens until the user transfers his pegged tokens back to their original chain, then the bridge would tell the contract to send the user back their funds

Note : The bridge's smart contract only uses the transfer(to, amount), transferFrom(from, to, amount), mint(to, amount), and burn(to, amount), other functions in the smart contract are not the bridge's concern nor an issue.

Nodes Going Offline

A nodes job is to keep the bridge up and running at all times, processing transactions from one blockchain to another, so if a node goes offline before submitting its leave, it would be affecting the voting system on the bridge by having its FT in the voting count but not actually voting, this might result in a delay processing transactions if the transactions can't get +50% voting approval. For a node to confirm it's online all nodes are required to submit a verification transaction every hour, if a node fails to submit its verification within the given time, it will be kicked out of the bridge by other nodes, and it's FT will be locked for 7 days before it can be withdrawn.

Some may ask why is it one hour, shouldn't it be lower to always make sure nodes are online? The answer to that is yes, making the confirmation period for nodes shorter is better to keep track on their availability, however that raised concerns on the expenses of the nodes, especially on blockchains such as Ethereum, that's why the 1 hour time has been set, it's not too much yet not too little, ideal to maintain an up to track bridge