zkStacks whitepaper
a private scalable Bitcoin Layer3

Introduction

Stacks 2.0 is a layer-1 blockchain integrates decentralized apps and predictable smart contracts empowered by Bitcoin's security and stability.

With the continuous development of numerous on-chain assets, Decentralized Finance assets require fast, frictionless, trustless, and real-time exchange services, which leads to new decentralized exchange protocols.

These protocols based on decentralized applications and their use cases would be the strongest and most widely used blockchain network to effectively ensure and scale blockchains in near future.

These protocols have significant development, but still it has some drawbacks:
1) higher transaction fee
2) transaction and execution needs to wait couple of blocks confirmation
3) ability to store a hundred millions of transactions

Scalability has always been an issue in networking, having a lot of popularity through thousands of engineers for many years, now it's reflected in the blockchain by transactions confirmation time, fees and storage size.

Blockchain ledger is fast growing though the time is ticking, meanwhile users expect faster response time, efficiency and productivity.

The two main approaches stay on top to solving the issue:
1) scaling techniques like sharding and distributed computing
2) scaling base layer by another one as known as Layer 2

Whether progress was made on base layer scalability, layer 2 scaling techniques enable privacy, security, high performance, less transaction fee and lower block size, it's a near to mid-term scaling solution for smart contract platforms like Stacks.

Existing Layer 2 techniques

Channels establish a connection between two parties, and allow sending payments back and forth between each other instantly. These channels are meshed together in the form of a network and could be used for smart contract state transitions. It's required both parties to actively monitor the base layer to ensure that the counterparty is adhering to the protocol, advised by a lighting node - watchtower.

Plasma chains are similar to smart contracts but take only transactions from the main Ethereum chain to free up work and improve verification. All variations of Plasma are predicated on the assumption that the assets deposited have an explicit owner.

Optimistic Rollups happens in sidechain, there is a virtual machine that processes transactions and smart contracts, all day-to-day. In that manner they move computation off-chain, but keep data on-chain, which is one of the major differences from plasma, and the key to solving the data availability problem. The solution is to deploy a smart contract to the mainnet which holds all the funds deposited. Sidechain users and operators ensure valid state transitions are committed to the mainnet contract.

Zero-Knowledge Proof

In cryptography, a zero-knowledge proof is a method by which one party (the prover) can prove certain information to another party (the verifier) by not simply revealing it.

Interactive Zero Knowledge Proof requires the verifier to constantly ask a series of questions about the “knowledge” the prover possesses. The process requires live interaction of verifier and prover and repeatable mechanism.

Fiat–Shamir heuristic is a technique for taking an interactive proof of knowledge and creating digital signatures based on it. This way information or fact can be verified publicly without prover being online all the time.

Zero knowledge proof and argument systems are protocols that replace human auditors as a means of guaranteeing computational integrity over confidential data and achieve both transparency and exponential verification speedup, simultaneously, for general computations.

Zero-Knowledge proof has the 3 major properties:
1) Completeness: A statement is true if an honest prover can convince an honest verifier.
2) Soundness: If the prover is dishonest, they can’t fool the verifier.
3) Zero knowledge: The verifier will not know what the statement actually is.

Zero-Knowledge Rollups

Zero-Knowledge rollups (ZK-rollups) bundle transactions into batches that are executed off-chain which reduces the amount of data that has to be stored on the blockchain. A summary of changes required to represent all the transactions in a batch is propagated to mainnet. The correctness of changes is produced by validity proofs, a cryptographic certainty.

Zero-Knowledge Succinct Non-Interactive Argument of Knowledge (ZK-SNARK) protocol creates a Common Reference String (CRS), a public parameters for proving and verifying validity proofs. The security of the proving system depends on the CRS, if information used to create public parameters is stolen, a bad actor may be able to generate false validity proofs.

Some ZK-rollups attempt to solve this problem by using a multi-party computation ceremony (MPC), involving trusted individuals, to generate public parameters for the ZK-SNARK circuit. Each party uses randomness to construct the CRS, which they destroy immediately.

To increase the security of the CRS setup, trusted setup is involved. As long as one honest participant destroys his input, the security of the ZK-SNARK system is guaranteed. This approach still requires trusting those individuals to delete their sample randomness and not undermine the system's security guarantees.

Zero-Knowledge Scalable Transparent Arguments of Knowledge (ZK-STARK) proves the validity of off-chain computation without revealing the inputs. ZK-STARK is considered an improvement on ZK-SNARK because of their scalability and transparency.

ZK-STARK works without the trusted setup of a Common Reference String (CRS) relying on publicly verifiable randomness to set up parameters for generating and verifying proofs.

ZK-STARK validity proofs increase quasilinearly complexity of the underlying computation versus linearly in ZK-SNARK. This means it is very useful for high-volume applications.

Currently both types ZK-rollups live on Ethereum network, they follow same architecture:
1) Mainnet contract stores rollup blocks, tracks deposits, and monitors state updates. Another on-chain contract (the verifier contract) verifies zero-knowledge proofs submitted by block producers.

2) Transaction execution and state storage happen on a separate virtual machine independent of the EVM. This off-chain VM is the execution environment for transactions on the ZK-rollup and serves as layer 2 for the ZK-rollup protocol. Validity proofs verified on mainnet guarantee the correctness of data stored in the off-chain VM.

3) Transactions data stays off-chain while state transitions of every transaction processed are stored in the mainnet. It doesn’t need to publish much transaction data on-chain because validity proofs already verify the authenticity. Storing state transitions on-chain is still important because it allows permissionless, independent verification of the L2 chain's state which in turn allows anyone to submit batches of transactions, preventing malicious operators from censoring or freezing the chain. Meanwhile access to state data is important to users to query their account balance or initiate transactions (e.g., withdrawals)

4) L2 transactions are finalized only if the L1 contract accepts the validity proof. This eliminates the risk of malicious operators corrupting the chain (e.g., stealing the funds) since every transaction must be approved on mainnet.

5) Most ZK-rollups use an operator node to execute transactions, produce batches, and submit blocks to L1. While this ensures efficiency, it increases the risk of censorship: malicious ZK-rollup operators can censor users by refusing to include their transactions in batches.

6) There is the ability to allow users to submit transactions directly to the rollup contract on mainnet if they think they are being censored by the operator. This allows users to force an exit (withdraw) without having to rely on the operator’s permission.

7) Users sign transactions and submit to L2 operators for processing and inclusion in the next batch. In some cases, the operator is rotated by a rule to prevent censorship, operators deposit funds in the rollup contract, with the size of each stake influencing the stakeholder's chance to be selected to produce the next rollup batch. Transaction data is published on Ethereum as calldata which publishes compressed transaction data as history logs which is a cheaper way to store data on-chain.

8) The state, which includes L2 accounts and balances, is represented as a Merkle tree. A cryptographic hash of the Merkle tree’s root (Merkle root) is stored in the on-chain contract, allowing the rollup protocol to track changes in the state of the ZK-rollup. The operator who initiated the state transition is required to compute a new state root and submit to the on-chain contract. If the validity proof associated with the batch is authenticated by the verifier contract, the new Merkle root becomes the new canonical state root as well as batch root — the root of a Merkle tree comprising all transactions in a batch. When a new batch is submitted, the rollup contract stores the batch root, allowing users to prove a transaction (e.g., a withdrawal request) was included in the batch. Users will have to provide transaction details, the batch root, and a Merkle proof showing the inclusion path.

9) The new state root that the operator submits to the L1 contract is the result of updates to the intermediate states. The operator hashes the updated account data, rebuilds the merkle root, validity proof (known as merkle proof) of every transaction to validate their existence in the merkle root and submits it to the on-chain contract.

Merkle proofs are used to decide upon the following factors:
1) hash belongs to the merkle root
2) to prove the validity of data being part of a dataset without storing the whole data set
To ensure the validity of a certain data set being inclusive in a larger data set without revealing either the complete data set or its subset.

Merkle trees make extensive use of one way hashing. Merkle proofs are established by hashing each level of hashes altogether and climbing up the tree until you obtain the root hash. One way hashes are intended to be collision free and deterministic algorithms. In order to verify the inclusion of data [D], in the merkle tree root, we use a one way function to hash [H] to obtain H(D). In order to validate that D is present in the tree, there is no need to be revealed without any implicit security repercussions.

zkStacks

zkStacks is a subnet designed to use ZK-STARK to ensure decentralization and privacy, Byzantine Fault Tolerant consensus - miners are responsible for issuing subnet blocks, users validate and subnet miners control withdrawals.

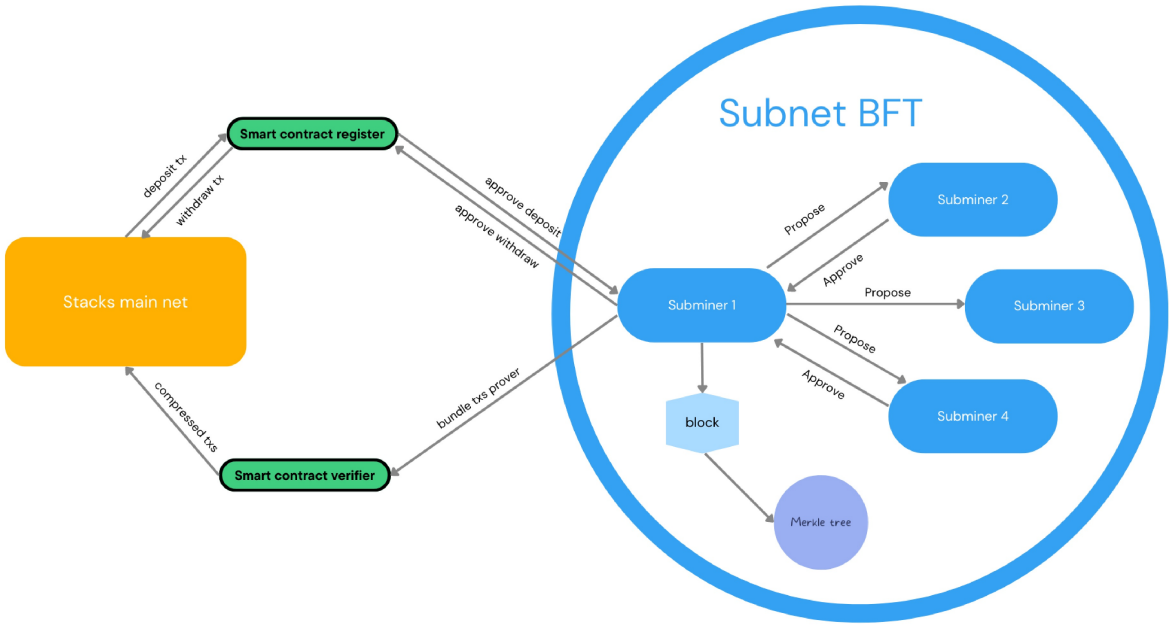
Subnet

Subnet is designed to transact Stacks assets, allowing users to move assets in and out. While a user's assets are in a subnet, they trust that subnet's consensus rules while It interacts with the mainnet using a smart contract.

A subnet itself is a secondary or substitute network that helps to scale the overall Stacks network. Each subnet node has its own Virtual Machine (VM) to ensure transaction validity, set of miners and consensus rules independent of the mainnet. Deposit and withdrawal require users to submit a layer 1 transaction to invoke the dedicated smart contract method.

Subminer bundles transactions and adds to the merkle tree, proposing the block to other parties, at least 67% are required to approve the proposal, ensured by Byzantine Fault Tolerant consensus.

Following block created in addition to the subnet, merkle root hash and merkle proofs for every transaction are propagated to a smart contract at the mainnet which verifies the merkle root



hash and ensures every transaction is part of it. It's done by checking the merkle proof of the transaction being part of the merkle root hash. In that manner the subnet is fully decentralized and it does not need a trusted or federated setup to operate. Verifier perform exclusive checks to ensure merkle proofs and their validity then create a block which actually does not contain raw transactions but the proofs. Users could request and validate transaction validation by querying the subnet by merkle root hash and merkle proofs. Ultimately, it’s up to the individual user to decide which subnet they want to use, and no matter which they choose, all subnets settle back on the Stacks blockchain, which in turn settles on Bitcoin. It opens up a huge amount of possibilities for new apps, more especially a Decentralized Exchange (DEX).

Mining and Decentralization

Subnets dictate how the membership looks like. If someone wants to create a subnet and mine it, they are able to in a fully decentralized manner. The chain will have an open mining system and a limited token supply with no pre-mine ones. zkStacks aims to be the most decentralized and secure as possible, whereby it’s open to everyone to participate in the chain.

Performance & VM

The subnet runs a Clarity VM, subminers will ensure transactions are valid and process through VM which will scale the mainnet to save resources. But it has built an abstraction layer for the public functions that are in the Clarity VM so people can reimplement those public functions and add their own flavor. As language design Clarity has many advantages like:
1) it’s interpreted and broadcasted on the chain not compiled
2) it’s decidable not Turing complete
3) recursion is illegal
4) there is no raw loops
5) overflows/underflows are catched

Tokenomics

Token ticker: ZKX
Fixed supply: 1,000,000,000 (1 billion tokens)
Distribution: open mining, no premine or token sale

Development fund: a portion of each block reward will flow to zkStacks Treasury; a community development fund to accelerate the number of engineers and applications building on zkStacks. The zkStacks Treasury will be managed by ZKX token holders via a DAO (Decentralized Autonomous Organization) using StackerDAO, a one-stop shop to create and manage DAOs secured by Bitcoin via Stacks. There are a number of applications already planning to build and launch on zkStacks.

All projects funded by zkStacks Treasury aim to benefit the network as a whole. Anyone will be able to submit a proposal for a project related to software, infrastructure, or app development,

marketing, community outreach, educational initiatives, and more. Developers that want to connect zkStacks to Lightning Network and build related apps will be highly encouraged to apply for grant funding.

Future Goals

In the future, **the ultimate goal** would be to connect zkStacks to the Lightning Network, via an LNSwap like implementation or other, to give developers and users the best of both worlds; expressive smart contracts via Stacks, and instant payments via LN. With TARO bringing tokenized assets to LN, it would be ideal if zkStacks could provide privacy options for TARO users.

Resources

https://ethereum.org/en/developers/docs/scaling/zk-rollups/
https://docs.ethhub.io/ethereum-roadmap/layer-2-scaling/zk-starks/
https://github.com/hirosystems/stacks-subnets
https://www.hiro.so/blog/an-update-on-hyperchains-a-scaling-solution-for-stacks
https://book.clarity-lang.org/ch00-00-introduction.html
https://grants.stacks.org/
https://github.com/stacksgov/Stacks-Grant-Launchpad
https://github.com/stacksgov/Stacks-Grant-Launchpad/wiki/Grant-Process-and-Payments
Application Submission: The only way to submit a new application: Stacks Foundation Grant Launchpad - Feel free to use this Grant Application Template to develop your application before submitting via the Launchpad.