Bitcoin (Blink) - Peer to Peer Global Finance System v1

Joby Reuben
joby@projectblink.org

Purva Chaudhari
purva@projectblink.org

Abstract : Bitcoin's PoW is replaced with a propagation competition on blocks sent across a certain set of validators under a time interval stamped with cryptographic proofs to claim fees and solve forks as per proof weight. To achieve adaptable scalability, each block's size is determined in consensus among elected nodes to reduce transaction waiting time. Gossip systems are replaced with a privacy-centered direct messaging system by constructing encrypted paths to deliver unconfirmed transactions and confirmed blocks. Apart from bringing speed, we resolved the need for a single transaction fee token by bringing forth a novel non-custodial per-token staking system to allow users to pay in any token. The "bitcoin" as a currency will ensure network security and hold Layer-1 tokens with staking and yielding fees. Since Bitcoin script adapts a Turing-incomplete language, the fees imposed for renting UTXOs make the transactions cheaper and the chain's ledger size optimized. We propose solutions for regulation revolving around taxation within the self-custody wallet ecosystem without compromising users' privacy.

1 Introduction

The Bitcoin Network [1] and other altcoin blockchains with newer consensus and programmable money are unable to compete with centralized payment providers in speed and volume due to their sheer inability to scale with centralization issues. Many consensus models rely on external validation concepts such as finding a nonce and proving stake instead of rules tied to propagation itself. Imposing heavy reliance on users to acquire native chain tokens diminishes the adoption of blockchain, thus keeping them far from the wonders of this technology. Decentralized networks can effectively adapt to users' needs by 1. Increasing block size 2. Decreasing block time 3. Eliminating low-efficient nodes 4. Increasing the requirement for node joining 5. Choosing producers[1] by performance. Retail Staking with non-custodial solutions encourages users to stake their bitcoin to become a world reserve currency for every financial instrument.

Instead of storing UTXOs for an indefinite period, which compromises storage, renting UTXOs and replacing them with a fingerprint after they have expired without altering the block's Merkle root provides cheaper fees. With Bitcoin's unlocking script and use of sCrypt [2], developers can create custom scripts with regulatory options involving various types of taxes within its UTXOs, performing identity verification off-chain, with signatures instructing nodes to validate regulated payments with self-custody of tokens. Decentralized Altcoins can be bridged one way to facilitate payments and ensure utmost security. A floating rate stable coin [3] without pegging to fiat can be issued with bitcoin as collateral for staking and yielding fees in the future. Basic banking solutions can be developed in Bitcoin Script, whereas common computable programs can be deployed to a Layer 2 State Machine [4], where nodes update the state by providing a Proof-of-Fee-Receipt paid in any token in the Bitcoin network.

[1] Nodes that mine/mint blocks

2 Bitcoin (Blink)

2.1 Election

Block size denotes the amount of data that can be propagated across every node on the Bitcoin network hence, its success rate is directly dependent on the bandwidth each node allocates for confirmed block transmission. Block size is not limited and is fixed in every new block, which denotes that every validator of the block can send and receive the data size. Variable block size helps to scale the network by increasing transactions per block when nodes upgrade and announce their bandwidth. A ZK-SNARK-based proof takes a node's bandwidth as a public input signal, which is compared to the threshold range of the desired bandwidth of the network. To prevent tampering, the node will commit a salt, which is a large random number that will be added to the bandwidth to generate the xyz hash which will be verified in the proof. Thus, the proof will attest its bandwidth output to the UTXO's script and get validated. The network in consensus can forbid low bandwidth

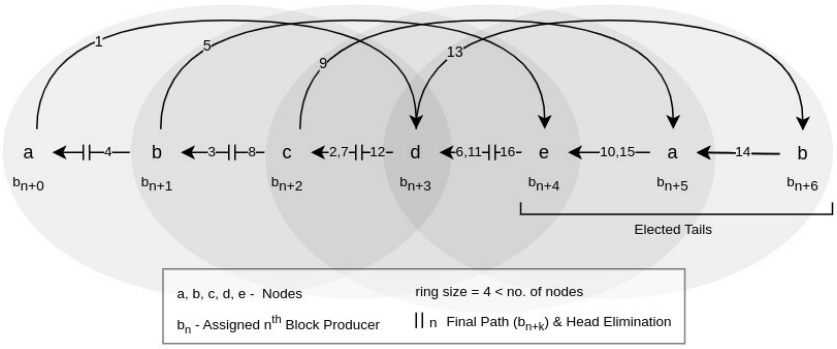

Figure 1: Ring Validation

nodes from participating in the election to produce blocks. Bandwidth requirements are increased for nodes to get elected for the next block production can assure the capacity to hold more transactions to scale further. Elections will be conducted based on nodes' bandwidth and each node's honesty weight. Node identities are masked by their public keys encouraging privacy. For each block, the elected node will produce, and a set of nodes will validate and attach to the longest chain [1]. This set of nodes is finite and unique for every block and is called a "ring validator" (see Fig.1). After each block's confirmation, the head of the ring is eliminated and a tail is elected which can be validated by nodes during propagation. The ring head after receiving the confirmed block will acquire bandwidth to gossip its block over to other full or SPV nodes to secure its rewards. The tail election's random seed is taken from concatenating Merkle Chain root and Hash Proof[2] [5] of the recent head's confirmed block. Thus, the election is conducted for every block where a tail node is assigned.

2.2 Staking

The chain native token - bitcoin can be staked for nodes' public keys with specified token IDs, where the collateral can be used only once for a block. This results in a stake per token per block bringing the throughput (tps), per token basis. The proofs are based on Zk-Snark-based Semaphore protocol [6] which will allow users to prove their membership in a group and send signals without revealing the original identity. Each token per block's collateral requirement is given in market price by taking the median volume of previous set of blocks (ring size). Staked bitcoins can be withdrawn at any time, with no vesting

[2]Popularized by Solana Proof of History consensus

period, except when the block is being created. This brings in a retail and non-custodial solution as opposed to security deposit-type PoS chains. Delegators won't lose their stakes as slashing is done directly on the fees during forks. For specific nodes, bitcoins can be staked to collateralize or lock in their allocated block. Clients can provide their general users with a savings wallet in which they can benefit from an annual percentage yield by providing liquidity for staking. In this way, for a specific token's transaction to be included in a block, the first transaction should prove the collateral specific to the token.

2.3 Regulation

Regulation via centralized exchanges & custodians risks funds and doesn't encourage a self-custodial ecosystem. Regulations are carried out by whitelisting verified hashed public addresses. Regulatory agencies can verify a public address by either doing minimal KYC or something such as mobile number-based OTP verification to encourage privacy through client wallet applications. UTXOs are stamped with region proof on their unlocking script based on specific spending conditions that will only allow a transaction on-chain if taxes are deducted properly. Bitcoin scripts can work efficiently and securely, as opposed to turing-complete smart contracts.

Tax models such as capital gains slabs can be issued by regulators trustlessly and are validated during the user's script execution. Each Regulator can specify their choice of token/currency from which the market price is derived for the transacted token leading to the regulator's choice of market pairs. Confirmed open-order transactions with token swap scripts can provide a real-time market price to calculate profitability and impose taxes. Onchain order-book decentralized exchange can perform trustless swaps and reduces the risk of centralization by price or market-feed oracles from outside sources. External taxes such as TDS and sales tax can be imposed off-chain by wallet providers with flexibility. These three taxes in a trustless-decentralized way provides almost all the tax models that a regulator can siphon into a centralized payment infrastructure.

2.4 Messaging

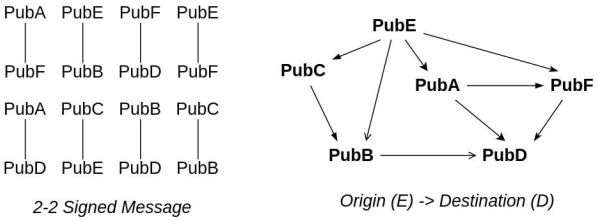

Figure 2: Topology

Delivery of unconfirmed transactions to nodes plays an important role in finality. Shared mempools defile the network with duplicated data, resulting in a poor choice of transactions by accepting higher fees to include in a block. A direct-messaging system should be deployed with messaging instructions specific to each party as opposed to gossip protocols. Paths are attached with unconfirmed transactions directly from the constructed network graph, which is available to all nodes with public keys as pseudonymous identities protecting privacy. Two peering parties mutually sign a 2–2 random message for every x block (ring size) and are gossiped across the network to identify the connection as online. All the signed random messages prove that each pubkey signature can display a network topology map (see Fig.2) from a node's point of reference. Paths are encrypted end-to-end with routing [7] instructions so that the origin cannot be traced. Nodes route

transactions to the destinations where producers can attach the transaction to their allocated block. Since the stake information is available in the public ledger, client wallets can construct transactions along with path information that routes to the nearest blocks for instant finality. Transactions are always atomic, providing a solution to queuing issues. Responsibilities are provided to all participants, where nodes only receive the transactions that they need to include, and client wallets should construct shorter paths to provide the best user experience.

2.5 Propagation

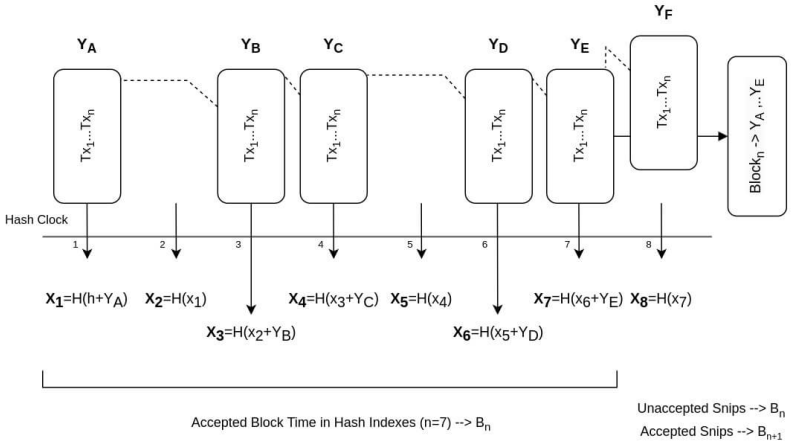

Figure 3: Snips

A Block is collectively validated but constructed as snips - divisible block chunks by the producer and directly messaged to the block's ring validators with routing instructions to propagate and get confirmed. Each snip references the previous snip's hash similar to the chain of blocks for proper identification of each block's snips. For a block, a competition to deliver all snips under x time interval in finite number of hashes is required to win rewards and avoid slashing of fees. When a block fails to win, it will do an intra-block fork and not mint its last of snips which will contain the fees and rewards. Hash proofs are attached for every snip (see Fig.3) during propagation among its assigned ring validators to declare the state of each block's competition and resolve forks. Failed block fees and rewards are slashed by sending them to a burn address by the next blocks which also results in the addition of negative weights that indirectly slashes the node's bandwidth costing capital. Null-Blocks are self-minted by its ring validators with proofs.

To synchronize time, each node's single-thread hash rate per second of a specific hash function is proved cryptographically onchain and taken in multiples of a common hardware's hash rate. An Individual hash-rate proof is provided along with bandwidth proof for every x blocks (ring size) before it expires which trustlessly synchronizes all nodes as a single hardware producing continuous hashes concated with all snips. The ZK IHR proofs are algorithmically similar to the bandwidth proofs, where the hash rate provided by the node is compared to the threshold range of the desired hash rate and salt is used to prevent tampering attacks. This time-based competition can be termed "Proof of Speed".

2.6 Rewards

Rewards are given for each snip hash concated with transactions validated by Hash proofs. Since newly mined bitcoins (5BTC/10mins) for a period of time are definite and each block's time is capped in blink implementation, new bitcoins can be supplied exactly

proportional to the current Bitcoin issuance rate with halving. The halving of Bitcoin issuance is done every $R \cdot (1.26 \cdot 10^8)$ hashes, where R is the common hardware's single thread hash rate. Each hash proof represents work done by nodes on validating and propagating transactions. When a fork arises due to rejected snips, the rest of the block time and its allocated rewards are slashed. Meanwhile, when a block is fully minted before the block's time finishes, the rest of its allocated rewards are distributed to the current halving period. This incentivizes nodes to attest and receive snips at the earliest to get an increased block reward in the upcoming block production. Tax outputs and rewards are attached as zero input transactions (coinbase tx) within the snip it contains. Fee outputs are created as a separate snip which denotes the end of a block.

During staking, producers announce their accepted tokens for which they will directly withdraw the commission. For other tokens, delegators can stake with a condition that their stake in bitcoins will be traded for the collected fees. During the commission withdrawal of non-accepted tokens, the producer will deposit collected fees to delegators and inflate the stake 1:1 ratio to withdraw the collateral. Users can pay transaction fees in any token, delegators incur the risk, and producers get paid in tokens of their choice to validate transactions.

2.7 Renting

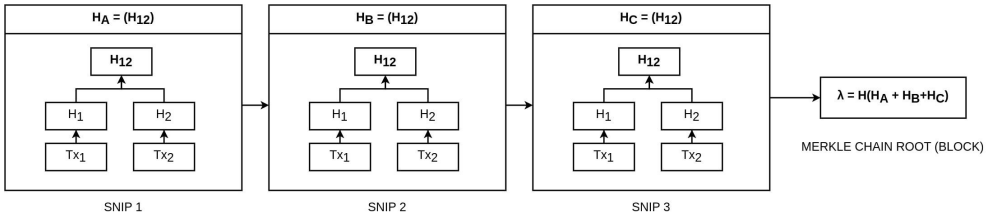

Figure 4: Merkle Chain

Instead of taking the Merkle roots of all the transactions inside a block, a snip's Merkle root is taken and linearly hashed to find the Merkle Chain root (see Fig.4). Since snips can be rejected by validators and cannot be tampered with once it is streamed to ring validators, it is unsure to predict a Merkle chain root making it a pseudo-random value. Inside a snip contains parsed transactions whose hashes are taken to find the snip's Merkle root can be pruned if the UTXOs are spent, burnt, or expired. Each UTXO expiry block height is embedded in its script, and can be scanned by nodes, and pruned to optimize their data storage. Client Wallets can store each of their users' transaction history and can be audited onchain using Merkle chain roots. Renting rates can be given in market price independently voted by producer nodes per byte per block. Users cannot directly pay for rent, but rather each new UTXO created is charged a transfer fee in the range of 0.05% - 0.005% decided based on the total volume of all transactions settled on previous set of blocks (ring size).

Transfer fee charges more fees for higher value utxos and less for lesser value utxos bringing ease to transact for retailers. Each UTXO's transfer fee will set an expiry date for itself. UTXOs doing state updates [8] will not be charged a fee, where users are incentivized to combine UTXOs to a single balance holding UTXO with an increased expiry value saving validators disk space.

2.8 Forks

Confirmed blocks as snips are streamed in a backward ring manner (block_n to block_{n+k}...block_{n+1}). For each block to prove block time by providing Hash proofs, a set of validators is initialized

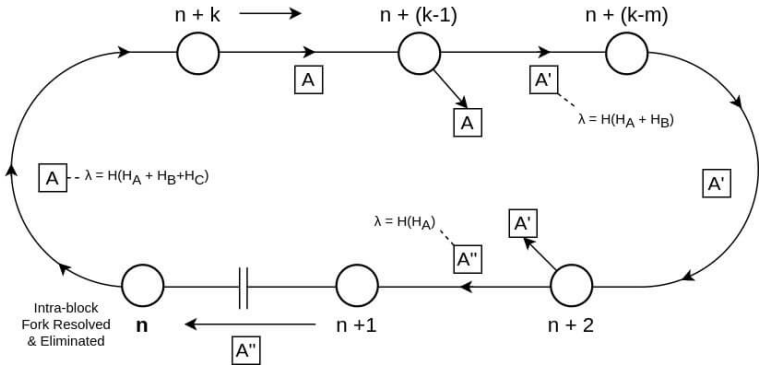

Figure 5: Resolving Forks

by the ring's size denoting the unique number of blocks and their producers. Intra-block forks arise when ring validators reject snips of a block (see Fig.5) which can be resolved by the very next block providing instant finality to users. Intra-block forks can be minimized by keeping the ring size variable and always lesser than the total number of nodes, a viable alternative to sharding. Block size and time are decided on consensus by the block's ring validators bringing a healthy propagation. Bandwidth upgrades are needed for the nodes which initiate the intra-block fork and are punished by the network by imposing negative weights. Each new block will include the previous block's Hash proof which resolves the intra-block forks by verifying the accepted snips and updating the chain using Merkle Chain Roots. During the offline activity of ring validators, Inter-block forks arise which can be resolved by attesting VDF Proofs of offline activity and thereby adding negative weights to it. Hence, each block will have its ring validators who are assigned by onchain accounted bandwidth and honesty weights, where the heaviest Hash-proofed version of the block_n is added to the longest chain.

2.9 Treasury

For the active development and sustainability of the project, a DAO Treasury is set up to fund developers and the community. A minimal commission is imposed on producer fees and deposited to a treasury script. Memberships are non-fungible and non-transferable 1. Temporary (Core-Developers), 2.Permanent (Investors, Community), 3. Contracts (Employees, Operations) given in x amount per y term for z period. Except for Permanent members, votes are taken to decide membership and treasury decisions. Temporary members can be kicked out for failure in active contribution whereas Permanent members cannot be kicked out due to their external contribution (funds) towards building the project. Permanent and Temporary members cannot be added after the first mainnet but can appoint heirs to their registered membership. Contracts are paid out initially and the rest is provided to other memberships as dividends according to their weight. Participants are only rewarded for their active contribution, not indefinitely like holding a fungible token or speculative participation. A decentralized open-sourced organization structure is maintained with decisions involving votes bringing forth sustainable growth for the future of project blink.

3 Future

The future of the Bitcoin network includes building finance-specific L1 applications such as exchanges, bridges, lending & borrowing, insurance, token minting, and bank-mirrored wallets. Since these applications are developed inside an unlocking script, it

requires preimage construction off-chain and settles onchain - inheriting the security of Bitcoin that centralized applications don't offer. Privacy can be improved by obscuring amounts similar to Monero with tax validation assisting regulators and masking financial information of a specific country. Emphasis on Zk delivery and validation of snips can reduce influence attacks in the future.

To introduce limitless scalability, an offline digital cash-payment system will be developed with suitable hardware wallets. To provide a general-programmable environment, a State Machine featured with multiple high-level languages using LLVM [9] IR code will be deployed as a layer that can update its global & contract state by providing a gas limit through a receipt-proof paid-in Bitcoin Network. The smart contracts will not contain balances and EOAs, but rather purely executed for business-logic to build DApps without a financial scope. To store client-side assets & data files to build fully decentralized applications, a CDN system will be developed in Bitcoin Network without duplicating data among nodes and provide faster content delivery to end-user. Moreover, research will be conducted to merge various Bitcoin and altcoin chains to a single decentralized global scalable infrastructure for a clean experience on the whole of finance and computing without hassles that are prevalent in current web3 ecosystem.

Implementations can be contributed openly to `https://github.com/projectblink`

References

[1] S. Nakamoto, "Bitcoin : A peer-to-peer electronic cash system," *Whitepaper*, p. 9, 2008.

[2] "scrypt — a high level smart contract language for bitcoin sv." `https://scrypt.io/`.

[3] "Dai stablecoin purple paper." `https://web.archive.org/web/20210127042114/https://makerdao.com/purple/`.

[4] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[5] A. Yakovenko, "Solana: A new architecture for a high performance blockchain v0. 8.13," *Whitepaper*, 2018.

[6] "semaphore-protocol/semaphore: A zero-knowledge protocol for anonymous signalling on ethereum." `https://github.com/semaphore-protocol/semaphore`.

[7] J. Poon and T. Dryja, "The bitcoin lightning network: Scalable off-chain instant payments," 2016.

[8] "Stateful contract — scrypt 0.0.1 documentation." `https://scryptdoc.readthedocs.io/en/latest/state.html`.

[9] "The llvm compiler infrastructure project." `https://llvm.org/`.

Appendix

https://github.com/projectblink

February 14, 2023 (Updated)

Short Sample text for explaining the below algorithm about IHR ZK circuit

```
/* Public signals                                                    */
signal input: node_ihr
signal input: ihr_hash
/* Private signals                                                   */
signal input: salt
signal input: required_ihr
/* Output signal                                                    */
signal output: if_pass
/* Range proof check                                                */
signal buffer
signal range_check
if node_ihr > required_ihr - buffer && node_ihr < required_ihr +
  buffer then
    range_check = true
end
/* Verify hash                                                      */
signal hash
signal hash_check
/* RIPEMD160 to calculate the hash                                  */
hash = RIPEMD160 (salt, required_ihr)
if hash == ihr_hash then
    hash_check = true
end
if range_check && hash_check then
    if_pass = true
else
    if_pass = false
end
```

Algorithm 1: ZK IHR Circuit

```
begin
end
```

Algorithm 2: VDF Proof

Merkle Chain

```
pre: the snip is added to the data
post: the data is added to the chain
begin
    add_node(snip)
    d ← snip
    if head = null then
        head,tail ← add_data(d)
    else
        tail ← add_data(d)
    end
end
```

Algorithm 3: class MerkleChain

```
pre: the value is added to the vector
post: the vector is generated to a merkle tree and added to the
      chain
begin
    New Vector data
    data ← d
    if size(data) == max_block_size then
        generate_root(data)
    end
end
```

Algorithm 4: class add_data(d)

```
pre: the vector data is added as the leaves
post: merkel tree and its root is generated
begin
    New Vector temp_data
    temp_data ← data
    while temp_data > 1 do
        for i = 0 i < size(temp_data) i+2 do
            Left ← temp_data[i]
            Right ← (i+1 == size(temp_data)) ? temp_data[i] :
              temp_data[i+1]
            combined = Left + Right
            new_temp_data ← hash(combined)
        end
        temp_data ← new_temp_data
    end
    node_root ← temp_data[0]
end
```

Algorithm 5: generate_root()

```
initialized: chain is a object of class MerkleChain and string data
begin
    while true do
        Output "enter data (q to quit)" Get data
        if data = q then
            Break
        else
            addnode(data)
        end
    end
end
```

Algorithm 6: main()

Algorithm 7: Ring

Algorithm 8: Update Weight

Algorithm 9: Fork Resolve & Proof

Algorithm 10: Tail Election & Block Req

Algorithm 11: Ring size change

Algorithm 12: Hash Reward

Algorithm 13: Network Graph

Algorithm 14: Routing Path

Algorithm 15: Stake Requirement

Algorithm 16: Stake Withdrawal

Algorithm 17: Taxes

Algorithm 18: Regulator Script

Algorithm 19: Swap Script

Algorithm 20: Exchange Rate

Algorithm 21: DAO Contracts

Algorithm 22: DAO Payout

Algorithm 23: DAO Dividend

Algorithm 24: DAO removal