BlockDAG

Whitepaper

December 2024

Introduction

The BlockDAG network combines the best aspects of modern decentralized ledger technologies. Today's decentralized ledgers are split into two camps. In one camp, the UTXO model provides scalable, privacy-friendly support for simple payments. In the other camp, the account-based model supports smart contracts, laying the foundation for decentralized finance and other decentralized applications.

UTXO vs Account Models?

In ledgers such as Bitcoin, the basic unit of value is the unspent transaction output (UTXO). Each UTXO has a value and an owner. In the everyday world, if you buy a $5 coffee with your $10 note, you hand that note to the cashier, who gives you back your coffee and her $5 note as change. In the same way, if you were to buy a $5 coffee with your UTXO worth $10, you would create a transaction with one input: your $10 UTXO, and two outputs: a $5 UTXO sent to the cashier's address, and another $5 UTXO sent back to an address you control, as change.

In ledgers such asEthereum, by contrast, each unit of value is associated with an account, which has an associated balance. To buy that $5 coffee, you would simply transfer $5 worth of value (cryptocurrency) from your account to the cashier's account.

Each model has certain advantages. The UTXO model is attractive because it is potentially more scalable. Since transactions have discrete outputs, they can be verified independently without having to locate and evaluate account balances. The UTXO model provides enhanced privacy: users can create new output addresses for each transaction, while the account model links balances directly to user accounts. Nevertheless, the account model is attractive for other reasons. The account model facilitates the use of smart contracts, programmable automata that can control and manage assets. Smart contracts lie at the heart of decentralized finance (DeFI) and other sophisticated decentralized applications (dApps).

The BlockDAG network supports both UTXO and account models. BlockDAG's UTXOs support fast, scalable payments, while BlockDAG's EVM-compatible account-based subsystem supports standardized smart contracts. A library is provided to connect these two worlds, providing developers with the best of both worlds.

Traditional Chain vs DAG Ledger?

Most decentralized ledgers are structured as blockchains, linear sequences of blocks, where each block is itself a linear sequence of transactions. (These transactions could be either UTXO or account-based.) This linear structure is intuitively appealing: the order in which transactions are executed can be important, and all parties to those transactions had better be able to agree on that order.

Nevertheless, linear blockchains can struggle with scalability issues, as every node must process every transaction, leading to slower transaction times, and putting pressure on fees. Moreover, many transactions are unrelated to one another, so there is no need for nodes to order them, because executing them in any order yields the same results.

Motivated by the costs of establishing a total (linear) order on transactions, researchers have developed several algorithms in which transactions are partially ordered: unrelated transactions can be verified and added to the ledger in parallel. The resulting ledger is not a linear chain, it is instead a directed acyclic graph (DAG). BlockDAG uses a version of the Phantom GhostDAG protocol to support a DAG-structured ledger.

Informally, a block is entered into the ledger after it is linked to enough earlier blocks so that it is highly likely to persist. As the DAG grows, the GhostDAG algorithm ensures that a universally accepted linear order eventually emerges. (See the Phantom GhostDAG paper for precise technical definitions.) Figure 1 shows how BlockDAG integrates UTXO and account models into a DAG-structured ledger
BlockDAG is fully Ethereum Virtual Machine (EVM) compatible, allowing for seamless integration and deployment of Ethereum-based smart contracts. This compatibility ensures that developers familiar with Ethereum's toolset, including Solidity and other Ethereum-compatible programming languages, can easily build and deploy decentralized applications (dApps) on the BlockDAG network.
Developers can migrate existing Ethereum dApps and coins to BlockDAG with minimal changes. BlockDAG also supports popular Ethereum development tools, such as Truffle, Remix, MetaMask, and Hardhat, simplifying the development and deployment processes. BlockDAG supports Ethereum coin standards like ERC-20 (fungible coins) and ERC-721 (non-fungible coins), facilitating coin issuance and smart contract functionalities.
The heart of BlockDAG's algorithm is the GhostDAG protocol, which is explained later in this document.

The UTXO-EVM Bridge

BDAG, BlockDAG's native currency, can be transferred from the UTXO domain to the EVM domain, and vice-versa. At any time, a coin lives in one domain or the other, but not both. The exchange rate between the two sides is 1 to 1.

The user transfers coins from the EVM domain to the UTXO domain using the following steps.

1. The user requests the EVM side to destroy ("burn") the coins prior to transfer.
2. The EVM side orders the EVM coin storage subsystem to burn the coins, then it informs the synchronization layer that the EVM-side coins have been burned.
3. When the synchronization layer confirms that the burn is final on the chain, it informs the asset management layer, which, after suitable checks, unlocks the same number of coins on the UTXO side.
4. If all steps have succeeded, the UTXO side then notifies the UTXO coin storage to assign the newly minted coins to the user, and confirms to the user that the UTXO-side coins have been created.
5. The synchronization layer confirms to the EVM side that the UTXO-side coins are unlocked, and the EVM side confirms to the user that the transfer is complete.
6. If the synchronization layer takes too long to confirm the burn, then it informs the EVM layer that the transfer has failed, and the EVM layer informs the user.

Transfers in the other direction are symmetric.

How coins are transferred from the EVM domain to the UTXO domain

Transaction Process Flow

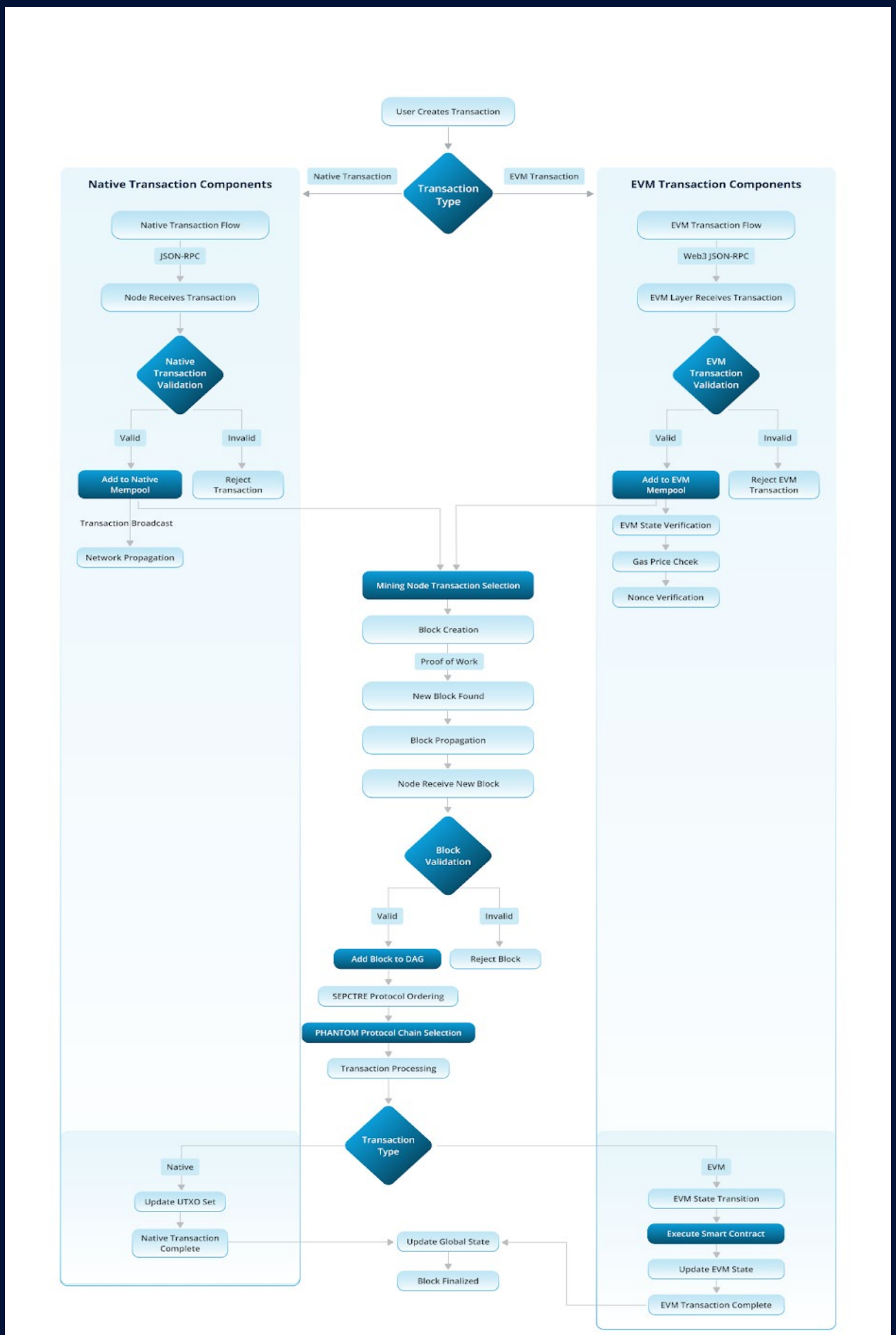


Figure: Lifecycle of a BlockDAG Transaction.

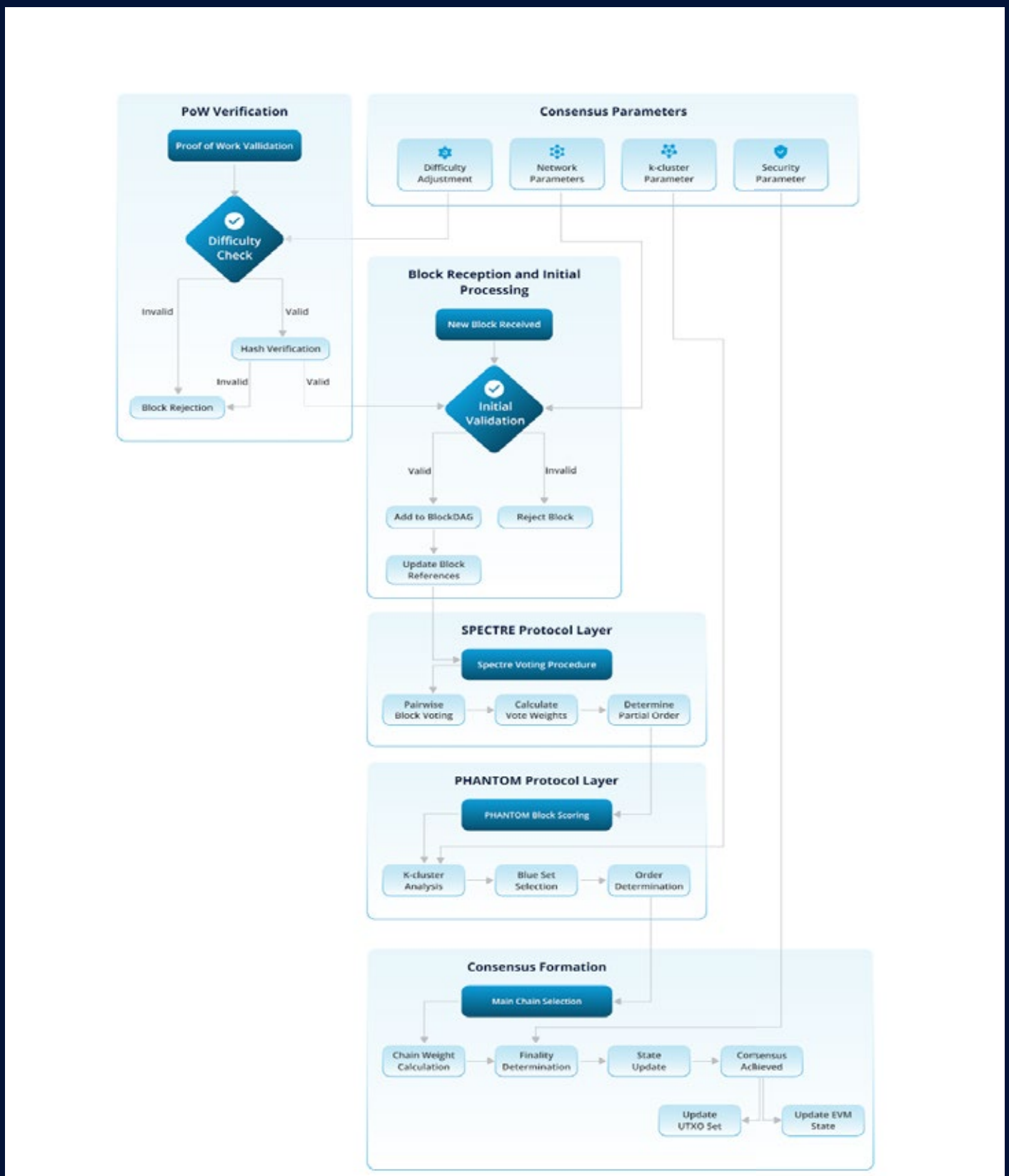
Consensus

The heart of BlockDAG ‘s consensus is a proof-of-work (PoW) algorithm. Just like most PoW chains, a miner prepares a block by assembling a list of transactions together with a block header. One field in the block header is called the nonce. The miner’s challenge is to find a nonce so that the result of hashing the entire block yields a hash value consistent with the chain’s current difficulty level.

In conventional blockchains, only one miner at a time can solve the hash puzzle to produce a new block. BlockDAG, by contrast, allows concurrent block production, where miners executing in parallel can both produce blocks. For details on how this is done, please consult the section on the GHOSTDAG protocol

Blocks are considered final once they have been referenced by enough later blocks in the DAG. Even though multiple blocks can be produced at the same time, the network eventually reaches consensus on the order and validity of transactions.

We can break the miner consensus protocol into several layers.



- The block reception layer accepts blocks from the peer-to-peer network, performs preliminary validation checks, and tracks updates to the DAG topology.
- The PoW verification engine tracks the current difficulty level, applies further verification, searches for and verifies the miner's solution to the block's cryptographic puzzle.
- The SPECTRE protocol layer establishes a partial order on the blocks in the DAG.
- The PHANTOM protocol layer extends the SPECTRE layer's partial order to a stable and secure total order.
- Finally, the consensus formulation layer determines the new network difficulty for subsequent PoW consensus, as well as updating the chain state to reflect newly-finalized blocks. The chain state consists of UTXOs, EVM account balances, and smart contract variables in the EVM.

Peer-to-Peer Network

Like most blockchains, BlockDAG uses a peer-to-peer (P2P) network to disseminate transactions and blocks across the network. The P2P network does elementary validity checks on transactions and blocks as they are transmitted. The P2P network uses a combination of TCP/IP for reliable, connection-oriented communication between nodes, as well as for block and transaction propagation. It uses UDB for efficient, connectionless discovery of new peers.

Transaction Fees

As in all blockchains, BlockDAG transactions incur fees. The fee structure varies depending on whether the transaction is a UTXO transaction or an EVM transaction.

For UTXO transactions, the fee structure is similar to Bitcoin's: the user attaches a fee to each UTXO transaction, and the first miner who mines a block including that transaction is awarded that fee. Fees are a kind of auction, where transactions bid to be included in a block, so users should attach higher fees to higher priority transactions. The higher a transaction's fee, the greater a miner's incentive to include that transaction in the next block. As a result, fees should be higher when the network is heavily used. Wallets, explorers, and similar tools can monitor current fee levels and suggest fee amounts appropriate for a transaction's priority level.

For EVM transactions, the fee structure is similar to Ethereum's. As in the UTXO domain, the user attaches a fee to each transaction: the higher the fee, the more attractive that transaction to validators, and the higher that transaction's priority. When a user calls a smart contract, that user is also charged a certain amount of gas for each computational step the contract executes. BlockDAG prices gas the same as Ethereum: all transactions in a block pay the same price, which rises in periods of high demand, and falls in periods of low demand. Gas is priced in fractions of BDAG units.

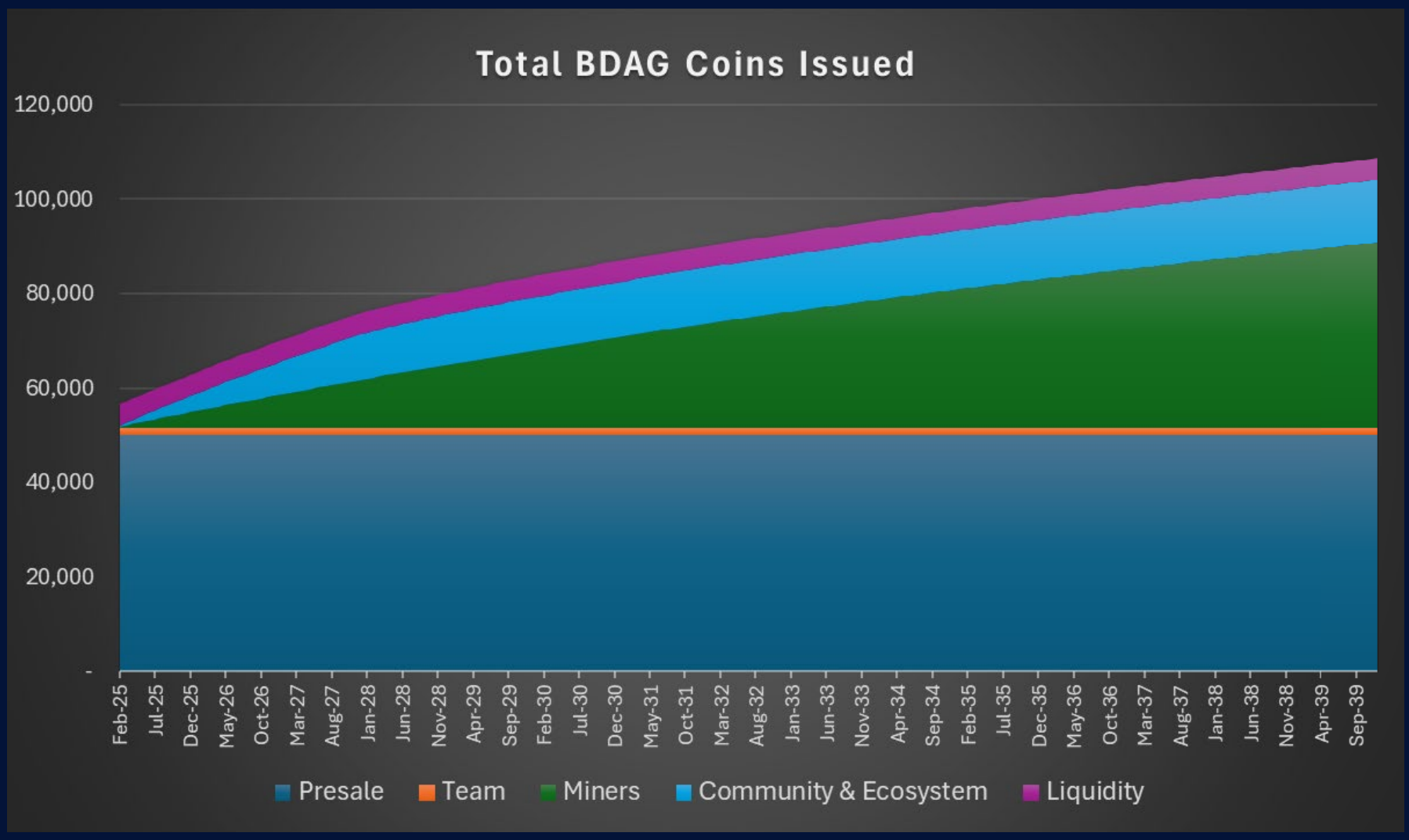
Initially, there will be a per-transaction fee of about $0.01 per transaction, split as follows: 50% goes to the miner and the rest is split between the network and the dApp originating the transaction.

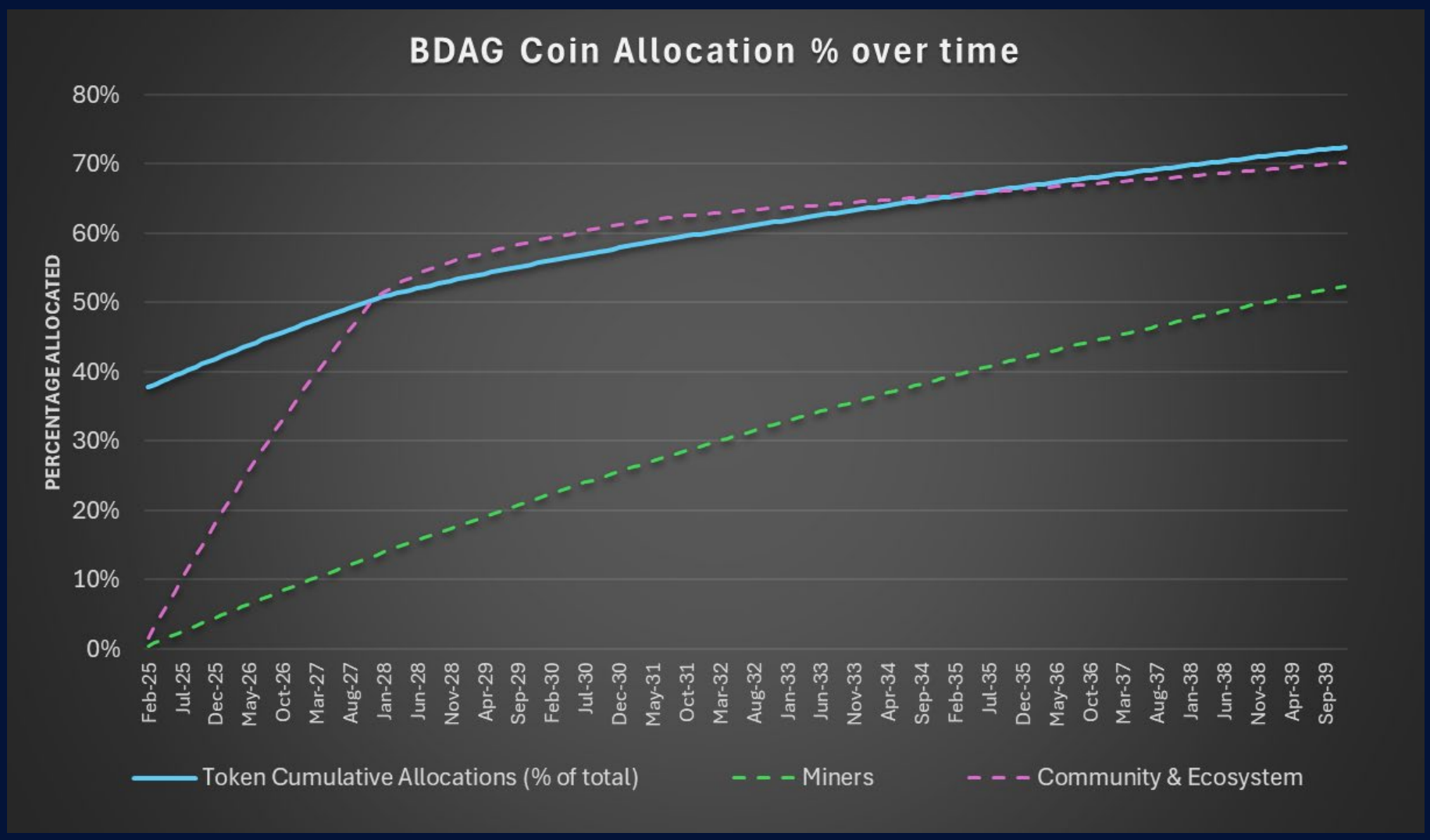
Coinomics

The BDAG coin is the native currency of BockDAG. There is a maximum coin supply of 150 billion coins, allocated as follows.

	Billion	Percent
Presale	50	33.3%
Team	1.5	1.0%
Miners	75	50%
Community & Ecosystem	19	12.7%
Liquidity	4.5	3.0%
Total	150	100%

The following graphs illustrate the forecast coin issuance over time.




BDAG Coin Allocation % over time
PERCENTAGE ALLOCATED
80%
70%
60%
50%
40%
30%
20%
10%
0%
Feb-25
Jul-25
Dec-25
May-26
Oct-26
Mar-27
Aug-27
Jan-28
Jun-28
Nov-28
Apr-29
Sep-29
Feb-30
Jul-30
Dec-30
May-31
Oct-31
Mar-32
Aug-32
Jan-33
Jun-33
Nov-33
Apr-34
Sep-34
Feb-35
Jul-35
Dec-35
May-36
Oct-36
Mar-37
Aug-37
Jan-38
Jun-38
Nov-38
Apr-39
Sep-39
Token Cumulative Allocations (% of total)
Miners
Community & Ecosystem

Security

Security is a cornerstone of BlockDAG. The system employs the following advanced cryptographic techniques to secure transactions, validate blocks, and protect against malicious activity.

- Public/Private key Encryption: BlockDAG uses asymmetric encryption for transaction signing and validation. Each participant has a private key for signing transactions and a public key for others to verify those signatures.

- Hashing: Blocks and transactions are hashed using a secure cryptographic algorithm (e.g., SHA-256) to ensure data integrity and prevent tampering.

- Proof of Work (PoW): BlockDAG employs a PoW-based consensus mechanism. Miners solve cryptographic puzzles (proof-of-work) to validate new blocks and add them to the DAG, ensuring the network is resistant to attacks like double-spending or 51% attacks.

- DDoS Protection: Mechanisms like rate-limiting and anti-spam filtering are built into the network layer to prevent Distributed Denial-of-Service (DDoS) attacks.

- Sybil Attack Resistance: By requiring proof of computational work, BlockDAG resists Sybil attacks, where an attacker tries to flood the network with fake traffic.

The security of the BlockDAG consensus protocol follows from the security of the underlying GHOSTDAG protocol used to generate and order blocks. AS noted, blocks are produced using a standard PoW consensus protocol. Unlike legacy PoW protocols, such as the one used by Bitcoin, the protocol allows miners to produce blocks in parallel. The GHOST DAG protocol is then used to order those blocks into a chain.

BlockDAG relies on the GHOSTDAG protocol to distinguish between blocks mined properly by honest miners and those possibly created by dishonest miners. As described elsewhere in more detail, the protocol exploits the fact that new blocks that are well-connected to blocks already deemed to be honest are highly likely to be honest themselves. New blocks that are only lightly connected to honest blocks are likely to be dishonest. Exploiting this observation, PHANTOM converts the DAG's natural partial order on the blocks that constitute the chain to ta total order. This order is stable in the sense that it is eventually agreed upon by all honest observers, and as long as a majority of miners remain honest, it would be extremely difficult to change the block order once it has been established.

BlockDAG also relies on the security of the bridge between the UTXO and EVM domains.
When an asset is transferred from one domain to the other, the bridging mechanism checks that the asset is valid in the source domain (UTXO or EVM) ensures that that asset can no longer be traded in the source domain, and that equivalent assets are created in the target domain (EVM or UTXO). The bridge maximizes flexibility while ensuring that assets can neither be created nor destroyed.

BlockDAG Nodes

BlockDAG supports two kinds of nodes, archival full nodes and more specialized miner nodes.

Full Nodes

Full nodes are the backbone of the BlockDAG network. They store a complete copy of the blockchain (DAG), validate transactions, and participate actively in the consensus process. Full nodes are critical for maintaining the decentralized integrity of the network and ensuring that all transactions adhere to the consensus rules.
Full nodes are archival: they are responsible for storing all past all past blocks. They help ensure security by validating blocks, and they communicate with other full nodes to propagate transaction and block information
Minimal hardware requirements for running a full node are listed below.

Miner Nodes

Miner nodes are specialized nodes that participate in the consensus process by solving the cryptographic puzzle at the heart of the PoW algorithm. Specifically, a miner collects pending transactions from the P2P network's transaction pool, validates them, constructs a candidate block, and attempts to solve that block's PoW puzzle. If it succeeds in time, the miner adds the new block to the DAG. Miners compete to earn block rewards and transaction fees. Miners also propagate newly mined blocks across the network for validation by full nodes. Miners secure the network by contributing computational power to prevent attacks such as 51% attacks.
Miner nodes are essential for maintaining the integrity and security of the BlockDAG network, as they ensure that blocks are added to the DAG through a competitive, decentralized process. They also contribute to the network's scalability by enabling parallel block creation in the DAG structure.

Mining Rewards - Issuance Schedule

Mining rewards follow a continuous, geometrically reducing schedule that ensures higher mining rewards early on, reducing over time as the coin value appreciates and network fees increase.

Figure: Mining Rewards per month, including alternative reward schedules that were considered.

Figure: Cumulative mining rewards issued, including alternative reward schedules that were considered.

Governance

BlockDAG will launch with a not-for-profit foundation whose charter is to advance a global economy that is transparent, inclusive and decentralized based on BlockDAG's ledger technology. The BlockDAG Foundation is a not-for-profit organization that supports open-source development within the BockDAG ecosystem. Initially, the Foundation will be responsible for managing a number of ledger-related tasks, with the intention of transitioning to a fully decentralized ecosystem within a few years.

Technology Stack

Here is a breakdown of the core technologies used by BlockDAG. This section is organized as a list for easy reference.

- **Blockchain Framework:** Custom DAG-based framework.
- **Consensus Protocol:** Proof-of-Work (PoW) based DAG algorithm, with enhancements to ensure EVM compatibility.
- **Smart Contract Platform:** Ethereum Virtual Machine (EVM) for smart contract deployment and execution, allowing seamless integration with Ethereum-based dApps.
- **Programming Languages:** Go (Golang): Core blockchain development, Solidity: For smart contract development, leveraging existing Ethereum-based libraries and frameworks.
- **Storage:** Distributed, decentralized storage system for transaction history, DAG structure, and state databases. LevelDB is used for local node storage.
- **Networking Protocols:** TCP/IP and UDP for peer-to-peer (P2P) networking, block propagation, and transaction validation.

Setting up the development environment for BlockDAG involves installing and configuring the required tools, frameworks, and libraries:

- **Local Development Environment:**
    - Operating System: Supports major OS environments like Linux (preferred), macOS, and Windows.
    - Development Tools:
        1. Go: Core language for blockchain development.
        2. Node.js: For integrating client-facing applications and smart contractinteraction using Web3.js or Ethers.js.
        3. Solidity Compiler (solc): For compiling smart contracts.

- **Node Setup:**
    - Developers will be required to run BlockDAG nodes for testing and interaction with the network. Test nodes can be configured locally to simulate the behavior of the mainnet.
    - Docker: For containerized deployment of nodes in different environments.

- **Wallet Setup:**
    - MetaMask can be used to interact with the EVM on BlockDAG for transactions and contract deployment.

Although BlockDAG is a new network, built from scratch, its EVM compatibility allows for integration with Ethereum-based projects and tools.

- **Smart Contracts:** Solidity-based Ethereum smart contracts can be deployed and executed directly on BlockDAG, providing access to a wide array of DeFi applications.
- **Bridges for Cross-Chain Transactions:** Future plans for interoperability include building bridges to connect BlockDAG with Ethereum, Binance Smart Chain, and other popular blockchains, enabling seamless asset transfers and interaction between ecosystems.
- **dApp Porting:** Existing Ethereum dApps can be ported to BlockDAG with little to no modification.

The following tools and SDKs will be available.BlockDAG CLI: Command-line interface for interacting with BlockDAG, allowing developers and operators to monitor the network, create transactions, deploy smart contracts, and manage nodes.

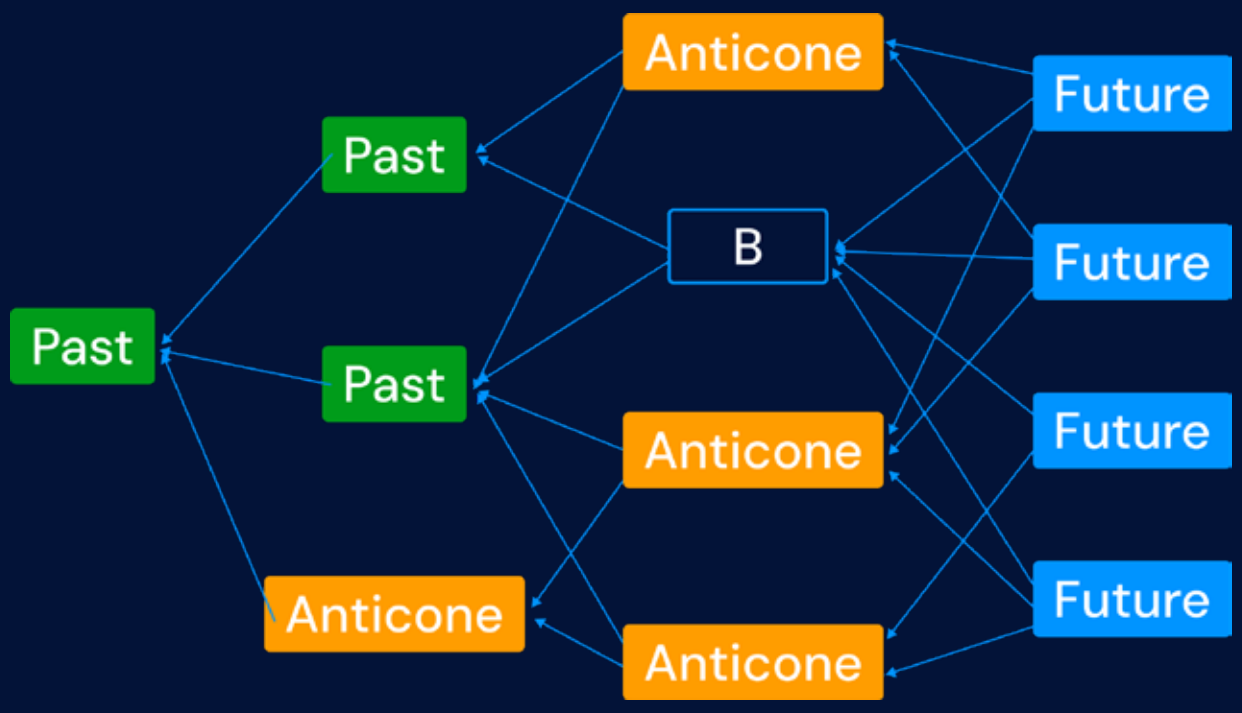
- **BlockDAG APIs:**
  - Tools to interact with the blockchain via API, enabling easier integration for third-party services or centralized applications.
- **BlockDAG SDKs:**
  - Web3.js and Ethers.js Integration: JavaScript libraries to interact with the EVM layer of BlockDAG for smart contract interactions, dApp development, and wallet operations.
  - Go/Nodejs SDK(Future Plan): A Go-based SDK for building custom applications and interacting with BlockDAG at a low level.
- **Wallets and Explorers:**
  - MetaMask, Trust Wallet & Plus Wallet integrations will allow users to manage their coins and interact with dApps on BlockDAG.
  - BlockDAG Explorer: A custom block explorer will be developed to track transactions, blocks, and smart contracts on the network.

The GHOSDAG Protocol - Simplified

This is a high-level, informal overview of the GHOSTDAG protocol at the heart of BlockDAG's UTXO side. Readers interested in technical detail should consult the original white paper. Here, we focus on what GHOSTDAG does, and why it works, but not so much on how it works.

Ledgers don't just record transactions, they order them as well. For some transactions, ordering is critical: many of us need to make sure our paycheck is deposited before our landlord cashes our rent check. In most blockchains, all transactions are ordered with respect to one another. Each transaction is part of a block where all transactions within a block are ordered, and the blocks themselves are validated and ordered by the miners. For transactions that depend on one another, this ordering is critical to prevent double spending. Suppose Alice dishonestly transfers the same coin to both Bob and Carol. If the transfer to Bob is ordered first, then Bob, not Carol, gets that coin.

Nevertheless, most transactions do not need to be ordered: if Alice pays Bob at the same time Carol pays David, there is no logical need to order those transactions because neither depends in any way on the other. This observation matters because there are costs to imposing such a total order on transactions, that is, ensuring that for every pair of transactions, one is ordered before the other. One cost is a missed opportunity for parallelism: if multiple miners are working at the same time, only one can succeed, which limits the rate at which new blocks can be added. Another cost is latency: all transactions must propagate through the network to a winning miner before they can be included in a block.

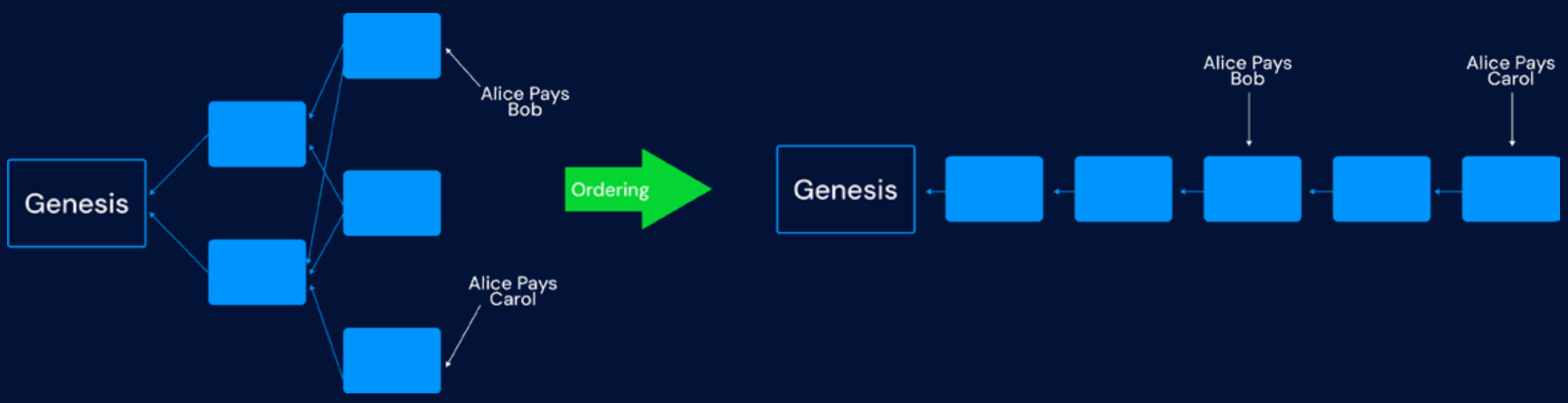


This figure shows how a block B divides the DAG into three parts: future(B) is the set of blocks that reference B, directly or indirectly, past(B) is the set of blocks that B references, directly or indirectly, and antigone(B) is the set of blocks that B does not reference.

What if we were to relax the requirement that all blocks have to be ordered at the time they are included in the ledger? Instead of constructing a linear chain of blocks, let us instead construct a directed acyclic graph (DAG) of blocks. When a miner creates a block B, that block contains references to all the latest blocks known to that miner. Let's call past(B) the set of blocks in the DAG reachable from B starting from B's references. The blocks in past(B) were certainly created before B. After B is added to the ledger, call future(B) the set of blocks in the DAG from which B can be reached. The blocks in future(B) were certainly created after B. Finally, borrowing a term from physics, call anticone(B) the set of blocks in the DAG that are neither in past(B) nor in future(B). These concepts are illustrated in this figure.

If B is created by an honest miner, B's anticone should be small. Suppose it takes time D for a transaction to traverse the network of miners. If an honest miner mines B at time t, then any honest block created before time t-D will have been seen by that miner, and will appear in past(B). Similarly, any honest block mined after time t+D will include B in that block's past. As a result, any honest blocks in B's anticone must have been created in the interval [t - D, t + D]. If B's miner is honest, there is a known limit k on the number of honest blocks in B's anticone.

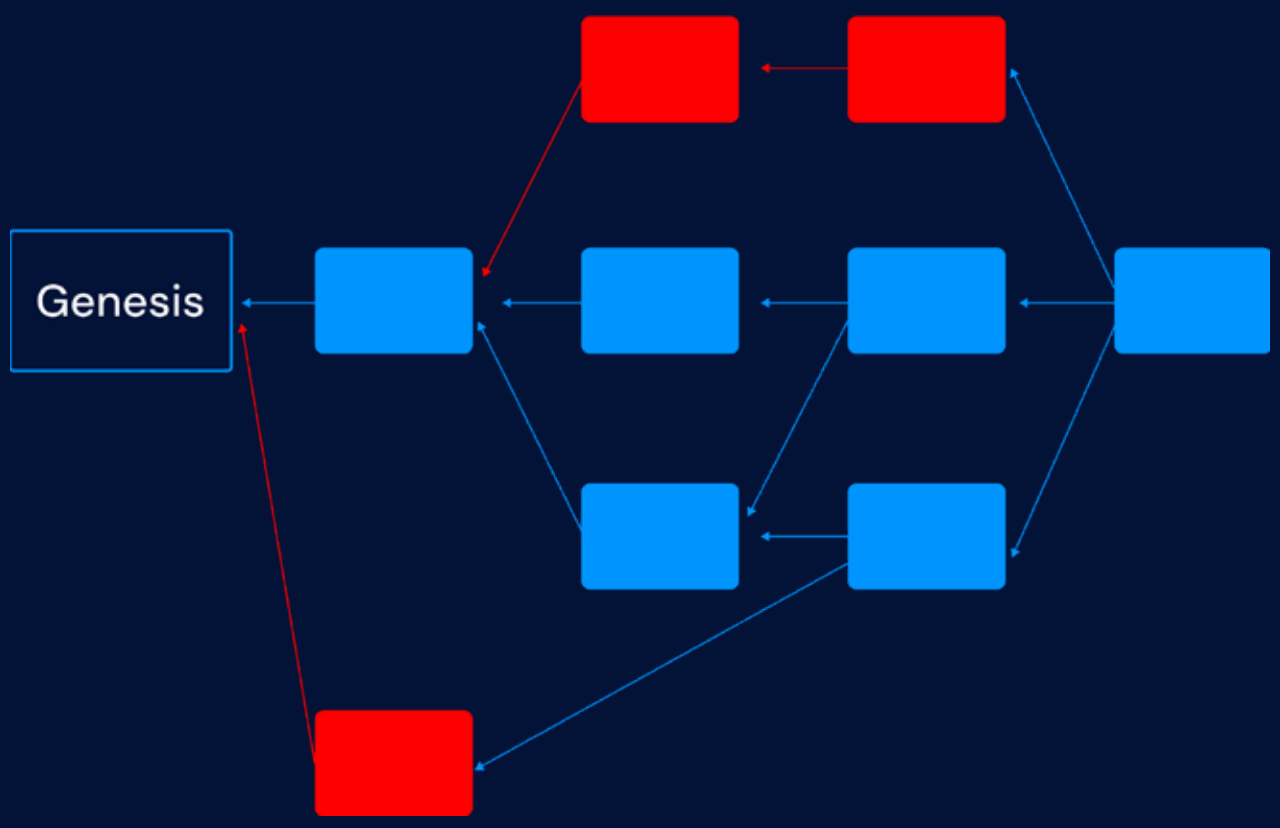
If one block can be in another block's anticone, then what prevents a dishonest miner from double spending? Suppose an honest miner mines block B in which Alice pays a coin to Carol, while a dishonest miner mines block B' in which Alice pays the same coin to Bob. If B and B' are in one another's anticones, then each block individually appears valid. Is this a flaw in the protocol?



This figure shows how conflicting transactions are resolved. On the left, Alice has attempted to double-spend a coin by sending it to both Bob and Carol, placing the transactions on unrelated blocks. On the right, the parties validating the chain gradually impose a total order on the blocks, and the later of the two conflicting transactions is invalidated and discarded.

This problem is resolved by gradually imposing a total order on all blocks in the DAG. Given a block B, we know blocks in past(B) should be ordered earlier than B, and blocks in future(B) should be ordered later, but what about the blocks in B's anticone? The GHOSTDAG protocol provides a sophisticated tie-breaking algorithm for ordering B with respect to the blocks in its anticone. Once established, GHOSTDAG guarantees that this total order is very unlikely to change. GHOSTDAG's total order is used to reconstruct the ledger state: each transaction is considered in turn, and any transaction incompatible with its predecessors is simply ignored. This figure illustrates how a double-spending conflict would be resolved.

The dishonest miners are assumed to have less computational power than the honest miners, so blocks mined by dishonest miners are likely to have larger anticones. GHOSTDAG's tie-breaking protocol favors honest miners by ordering blocks with small anticones before blocks with larger anticones, ensuring that honest transactions appear earlier than concurrent dishonest ones. In our example, GHOSTDAG's tie-breaking protocol will place Alice's (honest) transfer to Bob, with smaller anticone, earlier than Alice's (dishonest) transfer to Carol, with a larger anticone, causing the dishonest miner's transaction to be ignored.



The blue-colored blocks in this figure form a k-cluster: each blue block is linked to all but at most k other blue blocks (here, k is 2). These blocks were produced (with high probability) by honest miners. The red blocks are only weakly connected to the blue blocks: they were produced by dishonest miners who did not follow the mining protocol.

Define a k-cluster to be a set of blocks S with the property that for every block B in S, B's anticone in S has size less than or equal to k, where k is a known system parameter. This figure shows an example of a k-cluster within a DAG.

Here is a preliminary sketch of an algorithm. Given a DAG of blocks G and a fixed k, find a k-cluster in G of maximum size. Call this set of blocks the blue set, and the complement, the red set. Order the blocks in the blue set in any order compatible with the DAG order. Then, for any blue block B, add to the order just before B all of the red blocks in B's past that weren't added to the order yet. These red blocks should be added in an order compatible with their DAG order.

The intuition behind this algorithm is that it is easy for honest miners blocks to form a k-cluster, but as long as dishonest miners control less than half of the proof-of-work power, it is very unlikely the dishonest miners can create a k-cluster as large as one created by honest miners. It follows that the blue blocks (those belonging to the k-cluster) are highly likely to have been mined by honest miners. By contrast, red blocks are suspect: they have larger anticones, and are therefore more likely to have been mined by dishonest miners. Whenever there is a conflict between a transaction in blue block and one in a red block, the (honestly-mined) blue block transaction is ordered earlier, and the (dishonestly-mined) red block transaction is ignored.

Unfortunately, matters are not quite this simple. Finding a maximal k-cluster in a DAG turns out to be NP-hard, a technical way of saying that no one knows how to do so efficiently. So instead of finding a maximal k-cluster, the GHOSTDAG protocol efficiently finds a ``very big'' k-cluster, but not necessarily the largest. We do not need to dive into the technical details here, but interested readers are encouraged to consult the original GHOSTDAG white paper.

Recommended BlockDAG Node Hardware

This section lists the minimum recommended hardware setup to run a BlockDAG node. The components are organized as a list for easy reference.

Full Node Hardware

Hardware Components

- CPU: Multi-core processor (16+ cores) such as AMD EPYC or Intel Xeon.
- RAM: 64 GB or more of DDR4/DDR5 memory.
- Storage: NVMe SSDs (2TB+), with high IOPSfor fast data handling.
- Network Connection: 1 Gbps or higher, low-latency internet for efficient block propagation.
- Power and Redundancy: UPS, data backup solutions, and enterprise-grade networking.

Client Configuration

- Consensus Client: Manages GHOSTDAG operations, ensuring proper block ordering and DAG structure maintenance.
- Execution Client: Executes EVM-compatible smart contracts, processes transactions, and manages blockchain state.

Security and Monitoring

- Firewall/IDS: Protects the node from external threats.
- Monitoring Tools: Tools like Prometheus and Grafana for performance and resource monitoring.

Mining Node Hardware

Mining Infrastructure

- ASIC Miners: Specialized ASIC hardware that performs keccak256 hashing is necessary for mining.

Note: Miners must be connected to a BlockDAG full node to be able to mine.

Client Configuration

- Consensus Client: Manages block validation and integration, communicates with miners.
- Execution Client: Handles transaction processing and smart contract execution.

Redundancy and Scaling

- Geographically Distributed Data Centers: Ensure fault tolerance and high availability.
- Load Balancing: Prevents overload and ensures consistent performance across mining operations.

Security and Compliance

- Enterprise-grade Security: Firewalls, DDoS protection, and regular audits.
- Regulatory Compliance: Local energy consumption and data protection regulations.

Works Cited

Sompolinsky, Yonatan, et al. “Phantom GhostDAG.” Semantic Scholar, https://api.semanticscholar.org/CorpusID:21100.

Sompolinsky, Yonatan, et al. “SPECTRE: A Fast and Scalable Cryptocurrency Protocol.”Cryptology ePrint Archive, https://eprint.iacr.org/2016/1159

Legal Disclaimer

1. Disclaimers and Limitations of Liability

To the fullest extent permissible by the applicable law, the issuer of the BDAG Coin and any of their subsidiaries, affiliates, and licensors, and their respective employees, agents and contractors make no express warranties and hereby disclaim all implied warranties (including, without limitation, regarding any crypto coins, smart contract, etc.), including the implied warranties of merchantability, fitness for a particular purpose, non-infringement, correctness, accuracy, or reliability. Nor does the issuer of the BDAG Coin provide any warranties over any third-party services such as wallets, or marketplaces which you may use to access the BDAG Coin. You accept the inherent security risks of providing information and dealing online over the internet.

The issuer of the BDAG Coin will not be responsible or liable to You for any losses You incur as the result of your use of any blockchain network or any digital and/or electronic wallet, including but not limited to any losses, damages or claims arising from: user error, such as forgotten passwords or incorrect smart contracts or other transactions; server failure or data loss; corrupted wallet files; or unauthorised access or activities by third parties, including but not limited to the use of viruses, phishing, bruteforcing or other means of attack. Crypto coins are intangible digital assets that exist only by virtue of the ownership record maintained on the Blockchain. All smart contracts are conducted and occur on the decentralised within the blockchain, which is early stage and/or experimental technology. The issuer of the BDAG Coin makes no guarantees or promises with respect to smart contracts. The issuer of the BDAG Coin is not responsible for losses due to blockchains or any features of or related to them or any electronic and/or digital wallet.

The issuer of the BDAG Coin and their subsidiaries, affiliates, and licensors, and their respective employees, agents and contractors, will not be liable to You or to any third party for any indirect, incidental, special, consequential, or exemplary damages which you may incur, howsoever caused and under any theory of liability, including, without limitation, any loss of profits (whether incurred directly or indirectly), loss of goodwill or business reputation, loss of data, cost of procurement of substitute goods or services, or any other intangible loss, even if they have been advised of the possibility of such damages.

You agree that the issuer of the BDAG Coin's total, aggregate liability to you for any and all claims arising out of or relating to the BDAG Coin, is limited to the amounts You actually paid the issuer of the BDAG Coin in the twelve (12) month period preceding the date the claim arose. The issuer of the BDAG Coin sold the purchased BDAG Coin in reliance upon the warranty disclaimers and limitations of liability set forth herein, which reflect a reasonable and fair allocation of risk and form an essential basis of the bargain. Some jurisdictions do not allow the exclusion or limitation of incidental or consequential damages, and some jurisdictions also limit disclaimers or limitations of liability for personal injury from consumer products, so the above limitations may not apply to personal injury claims.

2. Governing Law and Jurisdiction

Any action related will be governed and interpreted by the Laws of the Seychelles, and shall, in the case of any legal action, be subject to the exclusive jurisdiction of the Seychelles, and You waive any objection to this jurisdiction and venue.

3. Arbitration

You and the issuer of the BDAG Coin agree that any and all disputes arising out of or in connection with the BDAG Coin will be resolved exclusively by means of individual arbitration. You and the issuer of the BDAG Coin agree that such disputes will be settled in accordance with the Centre for Effective Dispute Resolution ("CEDR") Model Mediation Procedures, and a mediator shall be nominated by the CEDR. You and the issuer of the BDAG Coin are waiving your rights to normal recourse to the Courts of Law.

4. No Class Action

You and the issuer of the BDAG Coin agree that any claims brought against each other will be brought in their own individual capacity, and not as a member of a class of claimants in any legal action.