SONIC SVM: A HyperGrid Scaling Future of Solana

Sonic SVM

March 18th 2024

Abstract

The incessant quest for scalability in blockchain systems has prompted the development of novel architectures designed to extend throughput and minimize latency. HyperGrid emerges as a pivotal solution in this domain, offering a scaling mechanism for the Solana blockchain. Through the utilization of state compression technology and Byzantine Fault Tolerance (BFT), HyperGrid aims to achieve a potential infinite transaction throughput by enabling horizontal scaling across multiple grids. Sonic, as a game-specific grid infrastructure, exemplifies the application of HyperGrid by facilitating the seamless onboarding of game developers and managing in-grid transactions. This document delineates the architecture, mechanisms, and operation of HyperGrid and Sonic, providing a comprehensive technical understanding of their roles within the Solana ecosystem.

1 Introduction

1.1 Background

The Solana blockchain is renowned for its high throughput capabilities and relatively low gas fees, and it has seen rapid growth in recent years. The number of wallet accounts grew from 100,000 in 2022 to 1 million in 2023, and is expected to reach 5 million in 2024, 10 million in 2025, 25 million in 2026, and 50 million in 2027. Simultaneously, the growth of dAPPs and defi activity is even faster, with the number of transactions executed per month jumping from an average of 4,188,712 per day in January 2022 to 205,823,984 per day in January 2024. Even by the most conservative estimates, the number of daily transactions is expected to far exceed 4 billion by 2026, and more likely reach tens of billions. Clearly, under such demand pressure, the throughput capacity provided by the current architecture of Solana L1 will face significant challenges.

And the situation may be even more severe. With the continuous emergence of dAPPs, although Solana has more performance advantages compared to Ethereum, it faces severe challenges from a large number of small games or

a single large game, especially Fully On-Chain Games (FOCG), where hundreds of thousands or even millions of users' on-chain interactions will severely impact the main chain performance of Solana L1, especially during certain special operational activities (such as server launches, holiday events, flash sales, etc.), where the instantaneous impact can be very large and frightening. This will not only affect the performance of the entire Solana L1 chain but also the playability and user experience of each game, including game response speed and data availability. It goes without saying that poor user experience is fatal for any game.

HyperGrid is designed to address such scenarios, being able to comfortably handle the large and high instantaneous transaction impacts of game-centric dAPPs. However, this is not HyperGrid's only feature;it will also significantly reduce the minting cost of non-fungible tokens (NFTs) produced by games or other dAPPs by integrating MetaPlex's compressed NFTs technology. On Solana L1, the traditional minting cost of a single NFT has risen from $0.02 in 2022 to 0.50$ in 2024, and is expected to become even higher in the future. Through HyperGrid, the minting cost of a single NFT will be significantly reduced to only 1% or even less of the traditional minting cost. In other words, the NFT minting cost for games running on HyperGrid will be reduced by more than 100 times. This will provide a much more cost-effective operating environment for games or dAPPs running on it.

1.2 Purpose of the Document

This document serves as a technical exposition detailing the architecture, components, and lifecycle of transactions within HyperGrid, using Sonic as an example. It caters to developers, researchers, and stakeholders within the Solana community, elucidating the intricacies of the HyperGrid system.

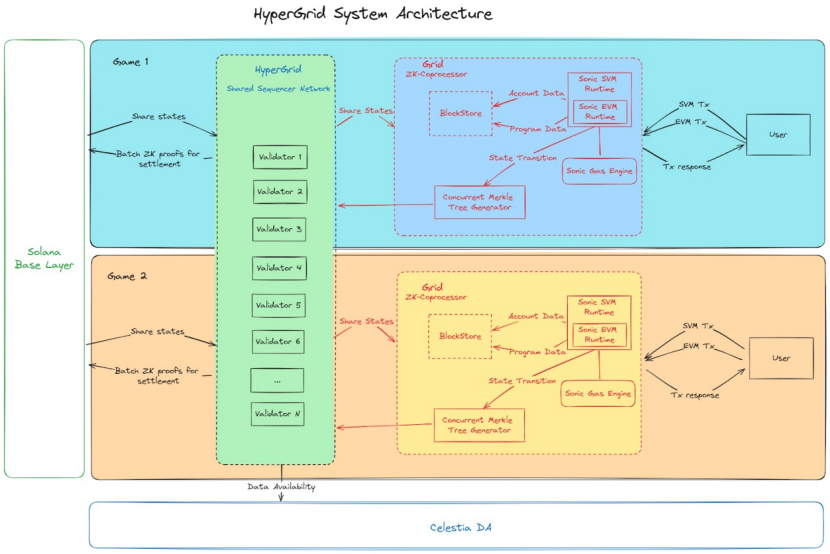
1.3 Scope

The focus is twofold: to describe the architectural underpinnings of HyperGrid and to elaborate on the transactional workflow facilitated by the Sonic grid for gaming applications. The document will articulate the integration strategy for validators, the execution engine, and the novel state transition mechanisms employed.

2 HyperGrid Infrastructure

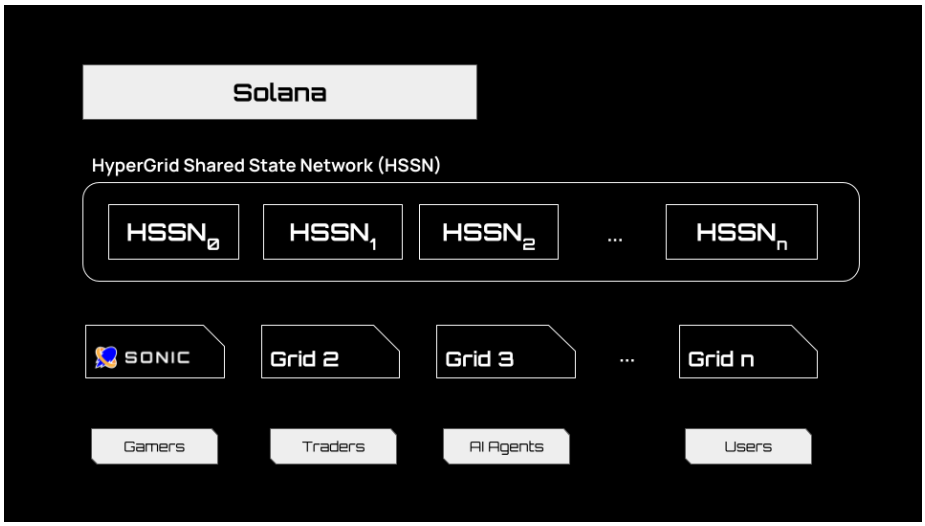
2.1 Architectural Overview

HyperGrid's architecture is predicated on a multi-grid approach, wherein each grid operates semi-autonomously while remaining anchored to the Solana mainnet for consensus and finality. This section will dissect the structural elements and their interrelations.



2.2 Grids and Network Relationships

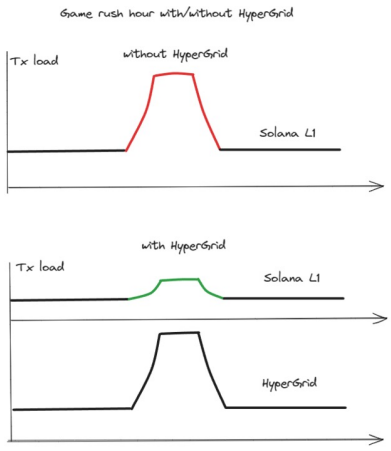
The grid is designed to allow a large number of dAPPs running on HyperGrid, in Sonic's case, large FOCGs, to reduce the impact on the performance of Solana L1 and minimize the performance interference between dAPPs. With the support of the HyperGrid architecture, dAPP developers can choose to use the HyperGrid public grid or horizontally scale out to create their own exclusive grid for their applications, based on their needs. The choice entirely depends on the developers' evaluation of performance and cost. dAPP developers can also shut down the grids they have created without affecting other grids, as the verification, recording, and reading of transactions and state changes produced by each grid are ultimately completed independently through the connection between HyperGrid and Solana L1.



2.3 Bridging and Program Deployment

The HyperGrid infrastructure emphasizes a bridging mechanism through robust security protocols and efficient verification processes, facilitating the liquidity of assets between the Solana mainnet and various grids. It's important to note that there isn't a direct bridge between the grids themselves. Instead, interactions with Solana L1 occur through the unified HyperGrid network, enabling cross-grid data access. The choice of grid form is left to the developers based on their needs.

If a developer's dAPP exceeds an average of 100,000 TPS or a peak of 1,000,000 TPS, it is recommended that the developer sets up a new, independent Grid (also known as horizontal scaling on HyperGrid). This approach ensures that a Grid exclusively used by a single dAPP is maximally isolated in performance from other Solana dAPPs, guaranteeing stable and reliable operation of the dAPP.



We offer two integration scenarios for games: 1. Utilize our Software Development Kit (SDK) to construct a grid. Following this, games can stake a specified fee to register their grid on our "Shared Sequencer Network." 2. Our SDK also supports a comprehensive installation option. This allows for the deployment of an independent "Sequencer Network" alongside the grid. Once set up, this system operates autonomously, without any ongoing connection to our services."

For developers whose dAPPs are in the early stages of development or have low-frequency interactions, deploying on the HyperGrid public Grid is an option. Ultimately, regardless of the chosen Grid form, dAPPs only need to upload and deploy contract or program code that can run on EVM or SVM.

HyperGrid's Sonic SVM offers compatibility with contracts or program codes previously running on EVM or SVM, equivalent to their native performance, ensuring seamless deployment or migration.

2.3.1 Bridging SOL to Grids

The process of bridging SOL to a specific Grid begins with the user sending their SOL to a designated program address controlled by HyperGrid validators on the Solana Mainnet. Upon receipt, the validators engage in a multi-signature aggregation process using BLS (Boneh-Lynn-Shacham) signatures, a cryptographic scheme that enables compact and secure aggregation of multiple signatures into a single entity. This aggregated signature attests to the veracity of the transaction and facilitates its recognition and confirmation on the Grid network.

Upon successful verification by the Grid, a corresponding amount of SOL is minted within the Grid, reflecting the amount locked on the Mainnet. This SOL issued on the Grid is encumbered by the state of the corresponding SOL

on the Mainnet, maintaining parity and ensuring conservation of value across the bifurcated system.

2.3.2 Exiting SOL from Grids to Mainnet

The exit pathways for SOL from Grids to the Mainnet are bifurcated into two distinct channels:

1. Standard Withdrawal Transaction:

- The user initiates a withdrawal request within the Grid.
- The Grid's validators process this event, and a BLS signature is generated, encompassing the assent of the required quorum of validators.
- This signature is then transmitted to the Mainnet, prompting the unlocking and release of the SOL corresponding to the withdrawal amount.

2. Emergency Exit Protocol:

- In the event that a Grid is rendered inoperative, a user may issue an emergency exit request directly on the Mainnet.
- If the Mainnet observes a lapse in the Grid's state commitments—indicative of an interruption in the Grid's operation—the user's funds are secured through an automatic claim mechanism.
- This mechanism is predicated on the last known state of the Grid, and the state root on the Mainnet is adjusted accordingly to reflect the updated holdings of the user post-exit.

This two-tiered system ensures that users have continuous access to their assets, providing a failsafe in the case of Grid anomalies.

2.3.3 Program Deployment on HyperGrid

Developers are required to upload and deploy contracts or program code on the grid that can run normally on EVM or SVM. HyperGrid's Sonic SVM offers compatibility with contracts or program codes previously running on EVM or SVM, equivalent to their native performance. In brief, Sonic SVM supports both EVM and SVM program codes while maintaining all characteristics such as parallelism, security, and speed.

For developers seeking to deploy programs on a Grid, the process is meticulously designed to be indistinguishable from deployment on the Mainnet. The existing tools and workflows employed for Mainnet deployments are replicated within each Grid, providing a seamless and intuitive developer experience. The salient advantage of deploying on a Grid, however, lies in the reduced gas fees and enhanced performance—an attractive proposition for developers aiming for efficiency and scalability.

The deployment procedure is rigorously optimized to maintain the high standards of security and reliability expected on the Mainnet while capitalizing on the performance enhancements innate to the HyperGrid structure.

HyperGrid supports two primary types of executables:

- Solana-Compatible Executables: Given HyperGrid's deep integration with the Solana blockchain ecosystem, native support for Solana-compatible executables allows developers to leverage the high throughput and low latency characteristics of Solana smart contracts.
- EVM-Compatible Executables: Recognizing the vast ecosystem of applications developed for Ethereum Virtual Machine (EVM) environments, HyperGrid also supports EVM-compatible executables. This inclusion ensures compatibility with a broad range of decentralized applications and services developed on Ethereum and other EVM-compatible chains.

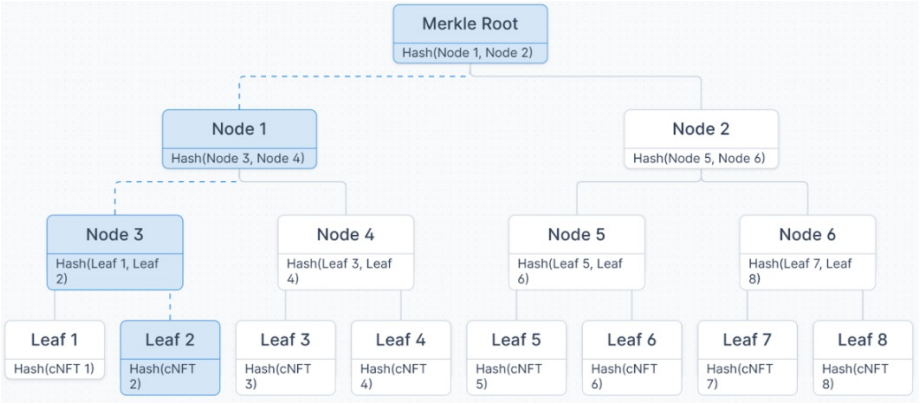
In summation, the bridging and program deployment mechanisms within HyperGrid exemplify the intricate blend of security, efficiency, and user-centric design, serving as the foundation for a scalable and robust extension of the Solana ecosystem.

2.4 State Transition Posting and Verification

The assurance of state integrity and the verification of state transitions are central to the operation of distributed ledger technologies. Within each Grid of the HyperGrid ecosystem, state transitions are orchestrated through a Concurrent Merkle Tree—a data structure that is pivotal to maintaining the ledger's immutability and facilitating concurrent state modifications.

2.4.1 Concurrent Merkle Tree Structure

The Merkle Tree structure is shown in the diagram:



The Concurrent Merkle Tree within HyperGrid is a sophisticated adaptation of the traditional Merkle Tree, modified to accommodate simultaneous write operations. Each leaf node encapsulates a triad of attributes:

- State Height: This denotes the version of the state, functioning as an identifier for each instance of the state's evolution. It ensures that the state can be tracked and verified across multiple iterations.
- State Key: Serving as a unique identifier, the State Key ensures the distinction of each state within the tree, permitting targeted queries and verifications.
- State Value: This is the actual data or state held within the leaf, representing the state at a given height and key.

These attributes are intrinsically chained within each leaf, forming a temporal record that extends beyond mere state representation to encapsulate the state's historical progression. This chaining is paramount to ensuring the integrity of the state's lineage, allowing for an audit trail of state transitions.

The concurrent nature of the tree facilitates simultaneous state updates, an essential feature for a system predicated on high throughput and efficiency. The concurrency mechanism provides that the Merkle Tree can handle multiple state transitions without contention, preserving consistency and the non-repudiation of the ledger.

2.4.2 Proof of History and State Transition Verification

The integrity of state transitions is safeguarded by a Proof of History (PoH), which is constructed by hashing the prior state (state(N)) together with all events emitted during the transition up to the succeeding state (state(N+1)). This hash serves as a cryptographic commitment to the sequence of events and the resultant state.

Validators perform verification by sequentially hashing the components: state(N), event(n), event(n+1), ..., thereby regenerating the hash of state(N+1). Should the locally computed hash match the submitted state root, the validator can ascertain the validity of the state transition with a high degree of confidence.

2.4.3 Commitment to the Mainnet and Data Compression

Upon validation, the state root is encoded into a compressed data account and posted on the Solana Mainnet. This procedure not only conserves space and optimizes network resources but also makes the state root of each Grid universally accessible and verifiable. Through this approach, the existence and validity of any state within a Grid can be ascertained by any participant of the network at any given moment.

This robust verification framework serves as the underpinning for HyperGrid's scalability, enabling users to trustlessly verify state transitions and, if necessary, facilitate an instantaneous exit to the Mainnet in the event of Grid service disruptions. The commitment of state roots to the Mainnet, coupled with the concurrent Merkle Tree structure, ensures that the HyperGrid system remains both transparent and verifiable, embodying the principles of decentralization and trustless operation that are hallmarks of blockchain technology.

1. Standard Withdrawal Transaction:

- The user initiates a withdrawal request within the Grid.
- The Grid's validators process this event, and a BLS signature is generated, encompassing the assent of the required quorum of validators.
- This signature is then transmitted to the Mainnet, prompting the unlocking and release of the SOL corresponding to the withdrawal amount.

2. Emergency Exit Protocol:

- In the event that a Grid is rendered inoperative, a user may issue an emergency exit request directly on the Mainnet.
- If the Mainnet observes a lapse in the Grid's state commitments—indicative of an interruption in the Grid's operation—the user's funds are secured through an automatic claim mechanism. - This mechanism is predicated on the last known state of the Grid, and the state root on the Mainnet is adjusted accordingly to reflect the updated holdings of the user post-exit.

This two-tiered system ensures that users have continuous access to their assets, providing a failsafe in the case of Grid anomalies.

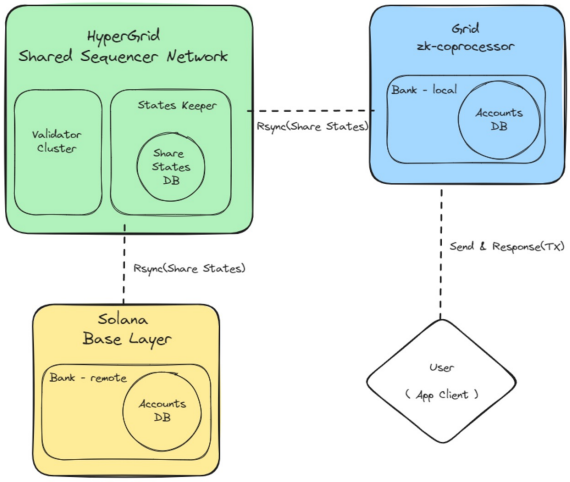
3 Interoperability with Solana L1

3.1 Definition of Interoperability

The interoperability between HyperGrid and Solana refers to the capability of the two systems to communicate, exchange information and assets seamlessly. It enables users and developers to operate across different layers without friction, such as: 1.Conducting transactions on Solana L1 and managing assets on HyperGrid. 2.Engaging in high-performance trading on HyperGrid with secure settlement on Solana L1. 3.Transferring assets and data across different Grids within HyperGrid.

Interoperability is crucial for expanding the overall capabilities of the Solana blockchain. It enhances Solana's scalability, security, liquidity, and availability, and fosters collaboration and development among the dAPP ecosystem running on HyperGrid.

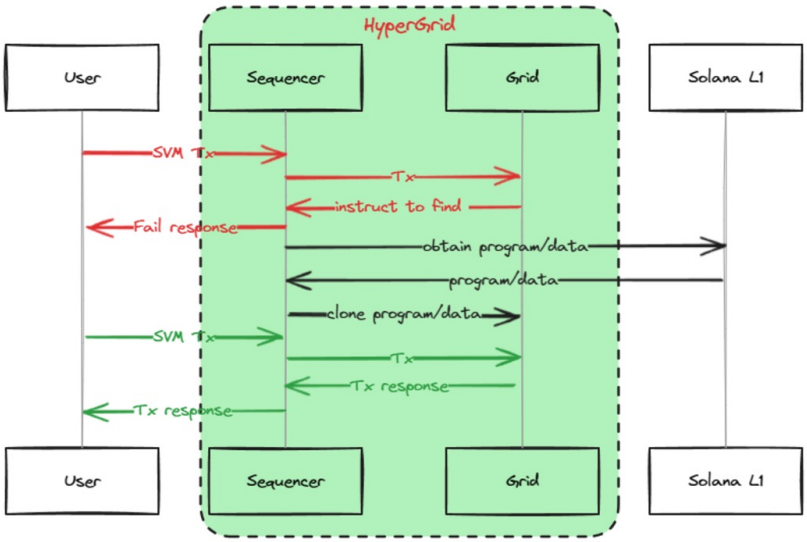
3.2 Interoperability Features of HyperGrid Architecture



This architecture achieves the interoperability feature of any Grid within HyperGrid with Solana L1 through the Sequencer Network, allowing for the effective execution of cross-chain transactions, synchronization of program and account data, while leveraging the privacy and scalability capabilities of the Sonic ecosystem.

The interoperability between HyperGrid architecture and Solana L1 is facilitated through the Sonic Shared Sequencer Network (SSSN). SSSN features decentralized mesh topology routing. Its characteristics include:
1.Mesh Routing: The mesh routing design of SSSN is a significant advancement over star routing. SSSN not only enables program calls, asset, and data transfer between Solana L1 and specific Grids within the HyperGrid architecture but also facilitates program calls between different Grids. This enhances the robustness of the entire system. Even if program code within a Grid is deleted, it might still be found and loaded through SSSN routing in Solana L1 or other Grid networks, ensuring the program can continue to run. This improves the system's fault tolerance and ensures a smooth user experience. Additionally, it greatly enriches the diversity of the dAPP ecosystem on HyperGrid. Game or dAPP developers can quickly build their applications using the program code available within HyperGrid, enhancing the developer experience.
2.Intelligent Scheduling: SSSN can real-time assess the busyness of each Grid within HyperGrid and predict the minute-level busyness based on historical access data to dynamically and intelligently allocate routing resources, thereby improving the performance of the entire architecture. SSSN can also intelligently tag or clean certain abnormal access behaviors (including continuous errors, common attack behaviors, etc.), reducing the likelihood of ineffective execution

or errors across Grids.



This architecture achieves the interoperability feature of any Grid within HyperGrid with Solana L1 through the Sequencer Network, allowing for the effective execution of cross-chain transactions, synchronization of program and account data, while leveraging the privacy and scalability capabilities of the Sonic ecosystem.

4 Sonic: Game-Specific Infrastructure

4.1 Within HyperGrid

Sonic, as a specialized segment of the HyperGrid infrastructure, is tailored to address the unique needs of the gaming industry within the blockchain ecosystem. This dedicated infrastructure encapsulates a suite of functionalities and optimizations that cater to the dynamic and real-time requirements of modern gaming applications.

4.1.1 Facilitating Gaming Ecosystems

Sonic operates as a pivotal conduit between game developers and the Solana blockchain, harnessing the robust capabilities of HyperGrid while introducing game-specific enhancements. It is designed to manage the high-throughput demands and rapid state changes characteristic of gaming platforms, thus offering a seamless experience for both developers and players.

4.1.2 Game-Specific Transactional Framework

The transactional model within Sonic is designed with the nuanced understanding that gaming transactions often involve high-frequency, low-latency interactions, which are critical for in-game events, player actions, and state updates. By leveraging the HyperGrid framework, Sonic provides an environment where transactions are not only processed swiftly but are also cost-effective, thus removing barriers to entry for game developers and users alike.

4.1.3 Optimized State Management for Gaming

Sonic's infrastructure is fortified with a state management system that is both resilient and flexible, ensuring that in-game assets, player states, and game logic transitions are maintained with integrity and are verifiable in real-time. The grid's ability to handle concurrent state updates means that player actions are reflected instantaneously across the network, critical for multiplayer and interactive gaming experiences.

4.1.4 Developer-Centric Approach

Recognizing the diverse needs of game developers, Sonic offers a suite of development tools that mirror the experience on the Solana Mainnet but are augmented to serve the specific requirements of game development. This includes specialized APIs, smart contract templates, and debuggers that align with industry-standard game development pipelines, thereby simplifying the process of building, deploying, and managing blockchain-based games.

4.1.5 Strategic Positioning within HyperGrid

Sonic is not merely an adjunct to the HyperGrid system but a strategic component designed to augment the scalability and functionality of the Solana network. Its integration into HyperGrid enables a symbiotic relationship where the advancements in gaming infrastructure feedback into the core network, driving innovation and performance enhancements across the entire ecosystem.

By addressing the intricacies and demands of the gaming sector, Sonic positions itself as an essential grid within HyperGrid, offering targeted solutions that advance the frontier of blockchain gaming. The role of Sonic extends beyond that of a service provider; it is an enabler of new possibilities in the intersection of gaming and blockchain technology.

5 Decentralized Validator Network

The robustness of any distributed ledger technology is largely contingent on the security and integrity of its validation mechanism. The HyperGrid platform fortifies this premise through a decentralized network of validators, underpinned

by a staking model that harmonizes with the overarching architecture of the Solana blockchain.

5.1 Staking and Security

The HyperGrid ecosystem adopts a staking protocol wherein validators are required to stake SOL as a collateral to participate in the network. This staking mechanism is not merely a passive lock-up of funds but an active commitment that validators make to ensure the security and continuity of all Grids within HyperGrid. Drawing parallels to the re-staking philosophy of EigenLayer, the collective staking pool serves as a foundational layer of security across the HyperGrid system.

5.2 Validator Roles and Responsibilities

Validators within HyperGrid are tasked with a multifaceted role that extends beyond conventional block production. Their responsibilities are delineated as follows:

- Transaction Sequencing: Validators sequence incoming transactions from the grids, ensuring order and consistency in the processing pipeline.
- Slot Proposal: Acting as facilitators, validators propose slots (equivalent to blocks) to each Grid. This can be seen as an outsourcing of execution, where the Grids are responsible for the computation of state transitions.
- Execution Verification: Post execution, validators verify the outcomes against the expected state transitions. This verification step is critical to maintain the ledger's integrity.
- State Commitment: Upon successful verification, validators are responsible for posting the verified state transitions to the Solana Mainnet.

This robust sequence of actions entrusted to validators is crucial in maintaining a trustless and decentralized environment, where no single entity holds undue influence over the network.

5.3 Incentive Model

In recognition of their pivotal role, validators are remunerated for their services with HyperGrid tokens. This incentive model is meticulously designed to encourage honest participation and discourage malicious activity. The reward structure, including detailed numbers and equations, is slated for an iterative development process. It is to be calibrated to align with the network's performance and security requirements, ensuring that the validators' interests are inextricably linked to the health and success of the HyperGrid ecosystem.

To initiate a Validator, an operator is required to stake 100 SOL tokens. Once established, this Validator becomes eligible to receive votes from other users. These users participate by purchasing Sonic Tokens, which are then locked up to become $veSonic. By staking their $veSonic tokens with a Validator, users are able to partake in the rewards generated by the Validators. The distribution of these rewards is then carried out between the Validator and the users who have staked their tokens, creating a symbiotic relationship within the system that encourages participation and investment in the network's health and growth.

Within the HyperGrid ecosystem, Validators also perform the responsibility of Shared Sequencers, thereby future L2 platforms built on HyperGrid are set to prioritize choosing Shared Sequencers from among these Validators. As part of their compensation, Validators receive a portion of the Sequence Fees, contributing to their revenue stream.

Additionally, the system incorporates Observation Nodes tasked with verifying the accuracy of transactions across various L2. Serving as watchdogs to ensure the security of the system, these Observation Nodes play a critical role in detecting any centralized malicious activities, which could trigger a Slash mechanism. For their surveillance efforts, these nodes are rewarded with $Sonic tokens.

After that, holding $veSonic signifies capturing value within the ecosystem, such as embedded swap fees, a share of NFT Marketplace transaction fees, and Launchpad revenue shares. This encapsulates the inherent value proposition of $veSonic. Furthermore, as a token of contribution to the ecosystem, $veSonic holders are eligible for additional ecosystem rewards. Prioritizing $veSonic holders, future games issued on HyperGrid will offer them rewards, allowing users to choose which game's rewards they wish to obtain, enhancing the interactivity and engagement within the ecosystem.

5.4 Dynamic Equilibrium

The staking and reward system within the HyperGrid validators network is not static but is engineered to adapt and evolve. The dynamic nature of this system is essential to respond to the changing landscape of blockchain security, transaction volume, and computational complexity. Validators' remuneration and staking requirements will be regularly assessed and adjusted to reflect these variables, ensuring that the network remains resilient against threats and efficient in its operations.

The HyperGrid validators network, witnessing a surge in transaction volume, might slow processing times and increase computational demand. To counteract, it may enhance rewards for validators, encouraging more to join and share the load, ensuring quick, secure transactions. Conversely, facing security threats, it could raise staking requirements, compelling validators to commit more collateral, deterring malicious actions.

In essence, the decentralized validator network is the bedrock upon which the HyperGrid's security paradigm is constructed. Through a judicious combination of staking, transaction sequencing, execution verification, and a carefully calibrated incentive mechanism, HyperGrid aims to sustain a secure, scalable, and robust infrastructure for the Solana ecosystem.

6 Execution Engine and Runtime Integration

Solana boasts a very solid architectural design, offering a variety of options for both clients and execution environments. HyperGrid, in this regard, selects the optimal one as its execution engine based on criteria such as high throughput capabilities, significant improvements in execution efficiency, and scalability and flexibility.

6.1 Paradigm Shift in Execution Technology

6.1.1 High Throughput Processing

Achieving the capability to process millions of transactions per second by optimizing consensus algorithms and streamlining the transaction verification process, thereby significantly reducing the latency traditionally associated with blockchain transactions.

6.1.2 Increased Execution Efficiency

Minimizing computational overhead through the use of complex data structures and state management techniques. Significant resource reductions for transaction execution have been achieved by implementing advanced caching strategies and state compression algorithms, thereby enhancing the overall efficiency of the HyperGrid network.

6.1.3 Scalability and Flexibility

A scalable architecture designed to dynamically adjust resource allocation based on transaction volume and network demands. This flexibility ensures that HyperGrid can maintain optimal performance under various loads, meeting the needs of an ever-growing ecosystem.

6.2 Integration with Runtime Environments

The integration of Firedancer within the HyperGrid ecosystem is complemented by its compatibility with diverse runtime environments. This section outlines the strategic implementation of runtime integration to support a broad range of smart contracts and decentralized applications (dApps).

6.2.1 Neon EVM Runtime Support

HyperGrid effectively addresses the issue of running EVM Transactions on Solana through NeonEVM, enabling HyperGrid to run both EVM and SVM contracts or programs simultaneously. It's particularly noteworthy that EVM Transactions are executed within the SVM runtime environment through an EVM interpreter, thus supporting parallel execution just like SVM Transactions. This process can be succinctly represented as EVM Transactions being translated by NeonEVM via RPC into Transactions understandable by SVM, and ultimately executed by Sonic SVM.

As repeatedly emphasized in this document, Sonic SVM, including the EVM Interpreter, still maintains all characteristics such as parallelism, security, and speed.

6.2.2 Smart Contract Deployment and Execution

Firedancer facilitates a seamless environment for the deployment and execution of smart contracts. By providing robust APIs and development tools, it ensures that developers can efficiently build, test, and deploy dApps with reduced gas fees and enhanced performance. The execution engine's design prioritizes developer experience, aiming to lower the barriers to blockchain development and foster innovation within the ecosystem.

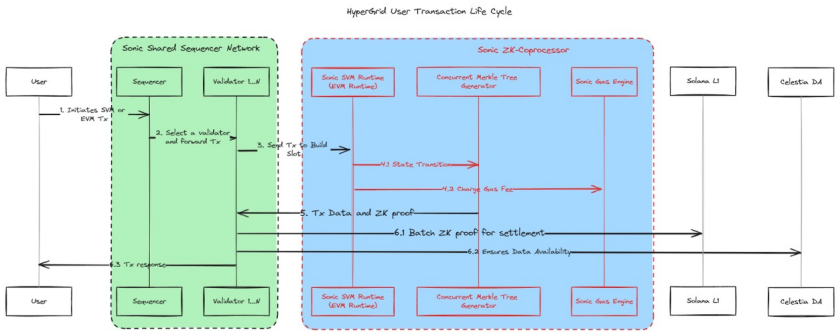
7 Zero-Knowledge Proofs in HyperGrid

The HyperGrid leverages Zero-Knowledge Proofs (ZKP), akin to Solana's Proof of History, to anchor the Grid's state root in transaction history and event sequences. This sophisticated cryptographic technique allows validators to verify the integrity of the state root without revealing the underlying data, ensuring both privacy and trustlessness.

7.1 Proving and Verifying Transactions

When a transaction is executed by the Sonic SVM, its resultant state changes are not directly committed to the Solana L1 blockchain. Instead, these changes are incorporated by the Sonic ZK-CoProcessor as leaf nodes in a Merkle Tree, employing a concurrent approach. This process effectively organizes and aggregates the transaction changes in a structured manner, allowing for efficient and secure storage. The Merkle Tree, a fundamental cryptographic component, enables the compact representation of these state changes.

Subsequently, the root hash of the Merkle Tree, which succinctly summarizes the entirety of the transaction changes, is submitted to the Solana L1 blockchain. This submission is critical for the integrity and security of the system, as it undergoes verification by the validators on the Solana network. The validators



are responsible for checking the authenticity and correctness of the transaction changes encapsulated by the root hash.

This verification process is part of the proofing procedure, where proof of the validity of the transactions is established without compromising on efficiency or security. By leveraging Zero-Knowledge proofs through the Sonic ZK-CoProcessor, the system ensures that the verification of transactions can be conducted without revealing the specific details of the transactions themselves. This approach not only enhances privacy but also significantly reduces the computational load and storage requirements on the Solana L1 blockchain, maintaining its scalability and performance.

8 Transaction Processing Lifecycle

The Transaction Processing Lifecycle in HyperGrid is a well-orchestrated series of events that ensure the swift and secure handling of user transactions from initiation to settlement.

8.1 Validator Involvement

Validators play a critical role in this lifecycle, beginning with transaction sequencing and culminating in the posting of the state root to the Mainnet. The lifecycle underscores the speed and efficiency of the Firedancer Execution Engine, which processes transactions within each Grid, facilitating rapid state transitions and real-time updates.

In the validator network, when users initiate transaction requests (tx requests), the network selects a validator according to predetermined rules to handle the request. The chosen validator then uses the Sonic ZK-Coprocessor to construct a slot, generating a Zero-Knowledge Proof (ZK proof). Subsequently, this validator submits these ZK proofs in batches to Solana L1 for settlement, while concurrently submitting transaction logs to Celestia to ensure data availability.

This process demonstrates how the use of advanced cryptographic technologies and distributed network architecture can achieve efficient and secure transaction processing mechanisms.

8.2 State Transition Flow

The flow of state transitions is meticulously tracked, from the initial state (state(N)) to the subsequent state (state(N+1)), with all intermediate events being hashed to form a Proof of History. This hash is verifiable by validators, ensuring the legitimacy of the state transition process. Which is represented as below:

$$PoH_{\{n,n+1\}} = H(S_n \| E_{\{n,n+1\}} \| S_{n+1}) \tag{1}$$

HyperGrid has designed and applied this function algorithm.

9 Data Availability and Celestia Integration

Data Availability (DA) is a critical facet of blockchain architecture, ensuring that data required to validate transaction states is readily accessible. HyperGrid incorporates Celestia DA for this purpose, providing a robust layer for data storage and retrieval.

9.1 Data Retrieval for State Verification

Celestia DA serves as a decentralized repository for transaction states, enabling validators and users to access historical data for verification purposes. This integration ensures that even in the unlikely event of a Grid failure, the system can recover and maintain continuity.

9.2 Integration Benefits

The benefits of Celestia DA integration extend to the entire Solana ecosystem, offering a scalable solution to data storage challenges and reinforcing the reliability of the HyperGrid platform.

10 Conclusion

HyperGrid heralds a pivotal evolution in blockchain scalability, leveraging cutting-edge technology to enhance the Solana ecosystem's throughput and interoperability. The Firedancer execution engine, with its capacity for processing millions of transactions per second, and the integration of the Neon EVM runtime represent significant strides towards accommodating a diverse developer base and broadening the network's appeal.
The implementation of Zero-Knowledge Proofs within HyperGrid underscores a commitment to privacy and security, facilitating a trustless environment where

transaction integrity is paramount. Concurrently, the Transaction Processing Lifecycle and Celestia DA integration ensure seamless operation and data availability, essential for the network's resilience and user trust.
In sum, HyperGrid is a transformative step for the Solana blockchain, not only addressing current scalability challenges but setting the stage for future advancements. It stands as a testament to the innovative spirit of the blockchain community, promising to drive the industry towards broader adoption and more sophisticated applications.

11 Glossary

[1] HyperGrid: A Rollups horizontal scaling solution proposed in the Solana ecosystem.
[2] Sonic Shared Sequencer Network (SSSN): A multi-node distributed system within the HyperGrid architecture for access management, intelligent allocation, program invocation, and serialized transaction execution.
[3] Sonic SVM: An execution environment for smart contracts and dApps within the Sonic ZK-Coprocessor of each Grid.
[4] Grid: The division of the HyperGrid system into multiple, smaller units for better management and scalability. Each Grid has its own independent Sonic ZK-Coprocessor component.
[5] State Compression Technology: A technology used in HyperGrid to reduce the size of the state for more efficient processing.
[6] Byzantine Fault Tolerance (BFT): Byzantine Fault Tolerance (BFT) is a property of systems that allows them to resist Byzantine faults, meaning that the system can continue to function correctly even if some nodes fail or act maliciously. Leslie Lamport, Robert Shostak, and Marshall Pease, "The Byzantine Generals Problem," ACM Transactions on Programming Languages and Systems, Vol. 4, No. 3, July 1982, pp. 382-401.
[7] Zero-Knowledge Proofs(ZKPs)ZKPs are a cryptographic method that allows one party (the prover) to prove to another party (the verifier) that a certain statement is true, without revealing any information beyond the validity of the statement itself. The beauty of ZKPs lies in their ability to maintain privacy and security: the verifier learns nothing except that the statement is true, and no specific details about the statement are disclosed.
[8] ZK-Coprocessor: A core component of the Grid, ensuring that each Grid has its independent transaction execution environment, program, and data management.
[9] Celestia DA: A data availability layer integrated with HyperGrid. Celestia is a modular data availability (DA) network that securely scales with the number of users, making it easy for anyone to launch their own blockchain. https://celestia.org/what-is-celestia/
[10] FOCG (Fully on-Chain game): Games where all data and states are stored and accessed via the blockchain.

[11] Throughput: The number of transactions a blockchain network can process per second.
[12] Latency: The time it takes for a transaction to be confirmed on the blockchain.
[13] Scalability: The ability of a blockchain network to handle a growing amount of transactions or processing.
[14] State Management: The management of the state, or the information about all the accounts, smart contracts, and balances on a blockchain.
[15] NFT (Non-Fungible Token): Digital assets that represent real-world objects like art, music, in-game items, and videos, which can be bought, sold, and traded online.
[16] EVM (Ethereum Virtual Machine): The runtime environment for smart contracts on Ethereum.
[17] SVM: Stands for Solana Virtual Machine. Solana is a high-performance blockchain platform designed to support decentralized applications (dApps) with high throughput and low latency. In this context, SVM refers to the smart contract execution environment on Solana, allowing developers to deploy and execute smart contracts. Compared to other blockchain platforms, Solana achieves its performance goals through innovative consensus mechanisms (like Proof of History) and other technological optimizations.
[18] Smart Contract: A self-executing contract with the terms of the agreement directly written into lines of code.
[19] Validator: Participants in a blockchain network who validate and add transaction blocks to the blockchain.
[20] Consensus Mechanism: A mechanism used in computer and blockchain systems to achieve agreement on a single data value among distributed processes or systems. Examples include Proof of Work (PoW) and Proof of Stake (PoS).
[21] Proof of History (PoH): An innovative consensus mechanism first proposed and used by the Solana blockchain. Unlike traditional blockchain consensus mechanisms like Proof of Work (PoW) or Proof of Stake (PoS), PoH aims to solve the problem of achieving agreement on the sequence of time among different nodes without relying on complex consensus algorithms to synchronize the system state. By doing so, PoH significantly improves the throughput and efficiency of blockchain systems. Anatoly Yakovenko"Solana: A new architecture for a high performance blockchain"2018
[22] Merkle Tree (also known as Hash Tree): A data structure widely used in various areas of computer science, especially in cryptocurrencies and blockchain technology. It's a binary tree where each leaf node represents a hash of a block of data, and non-leaf nodes are hashes of their children's hashes. This structure allows for efficient and secure verification of data content integrity and accuracy. Merkle, R. C. (1979). Secrecy, Authentication, and Public Key Systems. Stanford University. Merkle, R. C. (1980). Protocols for Public Key Cryptosystems. In Proc. of the Symposium on Security and Privacy, IEEE Computer Society, pp. 122-134.
[23] Concurrent Merkle Tree: A Merkle tree structure capable of supporting concurrent operations, i.e., handling multiple data insertions, updates, or vali-

dations simultaneously without sacrificing the ability to verify data integrity.
[24] FireDancer: A new validator client for Solana developed by Jump Crypto to increase its networking throughput, resilience, and efficiency.