Layerium: The Next Gen Layer2

André Gybson (*Former Loop Developers*)
Robert Artem (*Head of Loop Developers*)
Ericc Weiss (*Blockchain Developers*)
admin@layerium.com

Abstract

We introduce Layerium, a cryptocurrency system that overcomes the scalability and privacy limitations of previous systems like Ethereum, offering enhanced smart contract support. Similar to Ethereum, Layerium enables parties to create smart contracts by utilizing code to define the behavior of a virtual machine (VM) that executes the contract's functionalities. Layerium employs mechanism design to incentivize off-chain consensus among parties regarding the expected behavior of a VM. As a result, Layerium miners only need to verify digital signatures to confirm that parties have reached an agreement on the VM's behavior. In cases where unanimous agreement cannot be reached off-chain, Layerium still allows honest parties to progress the VM state on-chain.
Should a party attempt to deceive others about the behavior of a VM, the verifier (or miners) will identify and penalize the dishonest party using a highly efficient challenge-based protocol that leverages the unique features of the Layerium virtual machine architecture. By moving the verification of VM behavior off-chain, Layerium achieves significant improvements in scalability and privacy. We provide a detailed explanation of Layerium's protocol and virtual machine architecture, along with a functional prototype implementation.

1. Introduction

Universality plays a pivotal role in revolutionizing Layer 2 solutions, as it enables the accessibility of any Ethereum Virtual Machine (EVM) to replace the current infrastructure. The convergence of digital currencies and smart contracts presents a natural alliance. Cryptocurrencies facilitate direct digital currency transfers through distributed protocols, cryptography, and incentivization to uphold fundamental rules. On the other hand, smart contracts empower parties to establish virtual trusted intermediaries that adhere to mutually agreed-upon rules. This capability allows for the creation of intricate multi-party protocols with minimal counterparty risk.
By leveraging a cryptocurrency foundation, executing smart contracts on top of it empowers the integration of monetary conditions and penalties within the contract itself. These contractual provisions are effectively enforced by the underlying consensus mechanism. The introduction of universality within Layer 2 solutions ensures that any EVM can be readily accessed, thereby enabling the seamless implementation of complex protocols and the utilization of monetary features embedded within smart contracts.

1.1 Layerium

We introduce Layerium, a groundbreaking solution that addresses the limitations mentioned above, offering a new perspective on smart contracts. Verifiers in Layerium can manage contracts at a significantly low cost. (Here, the term "verifiers" is used generally to refer to the underlying consensus mechanism, similar to how Bitcoin miners function as verifiers in the Bitcoin protocol.) When parties act in accordance with their incentives, Layerium verifiers only need to verify a few digital signatures for each contract. Even in scenarios where parties act against their incentives, Layerium verifiers possess the ability to efficiently settle disputes regarding contract behavior without needing to analyze more than

Archived copy · cryp2nova.com

one instruction executed by the contract. Furthermore, Layerium enables contracts to execute privately by publishing only saltable hashes of contract states.
In Layerium, parties can implement a smart contract as a Virtual Machine (VM) that encompasses the contractual rules. The creator of a VM assigns a group of managers responsible for overseeing the VM. The Layerium protocol guarantees trust in any scenario: if at least one honest manager exists, they can ensure that the VM adheres to its specified code. Parties with a vested interest in the VM's outcome can either serve as managers themselves or designate a trusted representative to manage the VM on their behalf. In many cases, the set of managers required for contracts will be relatively small in practice.

Layerium introduces a novel approach where verifiers only track the hash of the VM's state, rather than the complete state. To foster agreement among managers, Layerium incentivizes them to reach consensus on the VM's behavior through external communication. Verifiers accept any state change endorsed by all managers, as long as it doesn't exceed the VM's funds. In the event that two managers, against their incentives, disagree on the VM's actions, verifiers employ a bisection protocol to narrow down the disagreement to the execution of a single instruction. One manager then submits a simple proof of that one-instruction execution, which verifiers can efficiently verify. The manager who provided incorrect information incurs a substantial financial penalty, acting as a deterrent against disagreements.

Parties have the ability to send messages and currency to a VM, and the VM itself can send messages and currency to other VMs or parties. VMs can take actions based on the received messages, while verifiers track the hash of the VM's inbox.

The architecture of the Layerium VM and protocol prioritizes fast and straightforward dispute resolution for verifiers. Further details about the design are elaborated later in the paper.

Layerium significantly reduces the cost associated with smart contracts. If participants act in accordance with their incentives, verifiers will never need to emulate or verify the behavior of any VM. Verifiers' primary responsibility in such cases is simple bookkeeping, tracking currency holdings, the hash of the message inbox, and a single hashed state value for each VM. In situations where a participant acts irrationally, it may require verifiers to perform a modest amount of additional work, but the verifiers will be appropriately compensated at the expense of the irrational party.

As a corollary, Layerium VMs can operate with privacy. This means that a VM can be created and executed without revealing its code or execution details, except for the content and timing of the messages and payments it sends, as well as hashed versions of its state. While any manager of a VM possesses the ability to disclose information about the VM, managers can choose to maintain its privacy if desired.

Layerium is designed to be consensus-agnostic, assuming the existence of a consensus mechanism that publishes transactions. The Layerium design seamlessly integrates with any consensus mechanism, including a single centralized publisher, a quorum-based consensus system, or Nakamoto consensus as seen in Bitcoin. Moreover, an existing smart contract system can serve as the consensus mechanism by encoding Layerium's rules as a smart contract. In this paper, we refer to the consensus entity or system as the Verifier, and the participants within the consensus system as the verifiers.

1.2 Paper Structure

The subsequent sections of this paper are organized as follows, focusing on Layerium as the central theme. In section 2, we address the challenges associated with the efficient implementation of smart contracts. Additionally, we introduce the Participation Dilemma, a novel

Archived copy · cryp2nova.com

theoretical finding in participation games that highlights potential issues with incentivizing smart contract verification. Section 3 delves into Layerium's approach, outlining its key principles. Expanding further, section 4 provides in-depth insights into Layerium's protocol and virtual machine architecture, which collectively enable highly efficient and privacy-oriented verification of smart contract-executing virtual machines. The implementation of Layerium and performance benchmarks, as well as the sizes of proofs and blockchain transactions, are presented in section 5. To provide context, section 6 presents an overview of related work in the field, while section 7 concludes the paper, summarizing the key findings and contributions of Layerium.

2 Scaling Smartcontract is real problem

Addressing the real problem of efficiently supporting smart contracts in a comprehensive manner poses significant challenges. This section focuses on analyzing the limitations and drawbacks associated with various existing approaches.

2.1 The challenge faced by verifiers

The conventional approach to implementing smart contract Virtual Machines (VMs) involves each miner in a cryptocurrency system emulating every step of VM execution. While this method offers simplicity, it severely hampers scalability.

The challenge faced by verifiers, as a consequence of the exorbitant cost associated with verifying VM execution, becomes apparent. When transactions involving code execution by a VM are costly to verify, the verifying party has an incentive to shirk their responsibility by accepting transactions without proper verification. They rely on the assumption that either (1) misbehavior will be deterred by others who do perform verification or (2) any discrepancies will go undetected by potential verifiers who also avoid verification. This equilibrium can lead to the acceptance of transactions with minimal or no verification.

Conversely, in a scenario where all miners diligently perform verification, a miner can exploit this situation by including a computationally intensive task that takes other miners a significant amount of time to verify. While the other miners engage in verification, the miner who included this demanding transaction gains a head start in mining the next block, granting them an unfair advantage in collecting the subsequent block reward. This dilemma arises due to the steep cost associated with verifying VM execution.

2.2 The challenge faced by participation

One strategy for scaling verification, exemplified by systems like TrueBit [30], involves leveraging participation games as a mechanism design approach to incentivize a select number of parties to verify the execution of each VM. However, these systems encounter a challenge known as the The challenge faced by participation. This challenge revolves around finding effective measures to prevent Sybil attacks, wherein a single verifier, whether honest or not, pretends to be multiple verifiers, potentially forcing other verifiers out of the system.

2.3 Participation Games

In this section, we present novel formal barriers that challenge approaches based on participation games. The concept revolves around players engaging in a costly process. Consider the following game scenario:

- There are n players, each having the option to pay a cost of 1 to participate.
- Participating player i selects a number of Sybils, represented by $s_i \geq 1$. Non-participating players set $s_i = 0$.
- Player i receives a reward of $s_i \cdot f(\sum_j s_j)$, where $f: N \rightarrow R+$ is a reward function.

In the context of this paper, participation can be viewed as "verifying a computation." It incurs a cost, but once a computation is verified, one can claim to have verified it from any number of additional Sybils without additional cost. These additional Sybils are indistinguishable from "real" verifiers. The objective is to design a participation game, characterized by the reward function $f(\cdot)$, such that in an equilibrium state, no

Archived copy · cryp2nova.com

player has an incentive to create Sybils, and the desired number of players participate, aligning the apparent number of verifiers with the actual number of separate player verifiers.
The authors of TrueBit have correctly identified that functions in the form of fc(m) = c · 2^(-m) are excellent candidates for participation games. Specifically, for a target number k of participating players, the participation game with a reward function f(m) = (2^k + 0.5) · 2^(-m) has a unique (up to symmetry) pure Nash equilibrium where every player sets si ∈ {0, 1}, and exactly k players participate. Moreover, an even stronger property holds: it is always the best response for any player to set si ≤ 1. We refer to such reward functions as One-Shot Sybil-Proof (formally defined in Appendix A). Initially, this suggests that participation games hold promise for verifiable smart contracts, as One-Shot Sybil-Proof reward functions exist.
However, a challenge that previous work fails to address is that smart contract verification is a repeated game.

In the context of repeated games, there exist multiple equilibria that do not align with the Nash equilibria of their corresponding one-shot versions. To illustrate this, let's consider the classic prisoner's dilemma: if the game is played only once, the unique Nash equilibrium is for both players to defect (which is a strictly dominant strategy). However, in the repeated prisoner's dilemma, there are various other equilibria, such as the well-known Tit-For-Tat and Grim Trigger strategies.
In Appendix A, we provide the formal model for repeated games (which is standard but not the primary focus of this paper). The significance of repeated games lies in the fact that players can make sacrifices in the present to gain benefits in the future. For instance, consider the following equilibrium in the repeated participation game with a reward function f(m) = (4.5) · 2^(-m):
- Player one adopts the strategy of setting s1 = 2 in every round.
- Players i > 2 set si = 0 in all rounds.
- Player two follows this strategy: if in either of the previous two rounds, the sum of sj (excluding j = 2) is less than or equal to 1, set s2 = 1. Otherwise, set s2 = 0.
It is worth noting that all players, except player one, are best responding. They currently receive a utility of zero (since player one sets s1 = 2 every round, causing all others to set si = 0). If the other players were to participate in any round, their utility would become negative. On the other hand, player one is also best responding. This is because if they were to decrease the number of Sybils in any round, it would lead to player two participating in the next two rounds (a formal proof is available in the appendix).
This equilibrium is not unnatural; it simply reflects how players beyond the first player react to the market conditions observed in previous rounds. Player one stays one step ahead, recognizing that regardless of the scenario, there will be two participants in equilibrium. Therefore, player one might as well be all of them to fully capture the reward. This property holds for any reward function, not just the specific form c · 2^i.

3 Layerium System Overview
In this section, we provide a high-level overview of the design of Layerium.

3.1 Roles
There are four distinct roles within the Layerium protocol and system.

The Verifier serves as the overarching entity or distributed protocol responsible for validating the legitimacy of transactions and publishing accepted transactions. The Verifier can take the form of a centralized entity, a distributed consensus system like a quorum system, a global network of miners akin to the Nakamoto consensus protocol, or even a smart contract operating on an existing cryptocurrency. Since the design of Layerium remains agnostic to the specific consensus system employed, we refer to it simply as the Verifier.

A Key represents a participant in the protocol who possesses currency and can propose transactions. Each Key is uniquely identified by

Archived copy · cryp2nova.com

the hash of its public key and can initiate transactions by signing them with the corresponding private key.

A VM (Virtual Machine) functions as a virtual participant in the protocol. Each VM is equipped with code and data that define its behavior in accordance with the Layerium Virtual Machine (LVM) Specification, which is provided in the extended version of this document. Similar to Keys, VMs possess currency, and they can send and receive both currency and messages. Creation of a VM is accomplished through a specialized transaction type.

A Manager of a VM assumes the responsibility of overseeing the activities of a specific VM and ensuring its correct behavior. When a VM is created, the corresponding transaction designates a set of Managers who will oversee its operations. Managers are identified by the hash of their public key.

3.2 Lifecycle of VM

The lifecycle of a VM (Virtual Machine) in the Layerium protocol can be described in the following stages:

1. Creation: A VM is created through a dedicated transaction, specifying its code and initial data. The transaction also designates a set of managers responsible for overseeing the VM's behavior.
2. Activation: Once created, the VM becomes active and ready to execute smart contracts. It can receive incoming messages and currency from other participants.
3. Execution: The VM executes the encoded rules of the smart contracts it hosts. It processes incoming messages, performs computations, and updates its internal state accordingly.
4. Verification: The actions and outcomes of the VM's execution are verified by the Verifier, which ensures the validity of transactions and publishes accepted ones. This verification process may involve consensus mechanisms or a consensus entity depending on the specific implementation of the Layerium protocol.
5. State Updates: As the VM processes transactions and smart contract execution, its state may change. Updates to the VM's state, such as currency holdings and message inboxes, are tracked and recorded by the Verifier.
6. Manager Oversight: Throughout the VM's lifecycle, the designated managers continuously monitor its behavior to ensure correct execution and compliance with the predefined rules. They may take actions to resolve disputes or address any issues that arise during the VM's operation.
7. Termination: At some point, a VM may reach the end of its lifecycle. This can occur when a specific condition is met, such as the completion of all smart contracts hosted by the VM, or when it is explicitly terminated by a manager or participant.

3.3 The Verifier Role

The Verifier plays a crucial role in the Layer2 blockchain powered by Layerium. As a global entity or distributed protocol, the Verifier is responsible for verifying the validity of transactions and ensuring the integrity of the blockchain. It serves as the authoritative entity that publishes accepted transactions and maintains the consensus mechanism.
In the Layerium ecosystem, the Verifier can take different forms depending on the chosen consensus mechanism. It could be a centralized entity, a distributed multiparty consensus system such as a distributed quorum system, or even a smart contract deployed on an existing cryptocurrency network.
Regardless of its specific implementation, the Verifier's primary function is to validate transactions and ensure the correctness of the blockchain state. It verifies the execution of smart contracts, confirms the accuracy of transactional data, and validates the actions performed by participants within the Layer2 ecosystem.

The Verifier also plays a critical role in maintaining the security and trustworthiness of the Layer2 blockchain. By verifying transactions and ensuring their compliance with the protocol's

Archived copy · cryp2nova.com

rules, it helps prevent fraudulent activities, double spending, and other malicious behaviors.

Furthermore, the Verifier contributes to the overall scalability and efficiency of the Layer2 blockchain. By offloading transaction verification from the main Layer1 blockchain, it enables faster transaction processing and reduces the computational burden on the network.

In summary, the Verifier on the Layer2 blockchain powered by Layerium acts as the trusted entity responsible for verifying transactions, maintaining consensus, and upholding the security and efficiency of the network.

3.4 Benefits

1. **Universality**: Layerium addresses the problem by creating a Layer 2 solution that is universally compatible with any EVM (Ethereum Virtual Machine) blockchain. This means that Layerium can seamlessly integrate and operate on various EVM-based blockchains without requiring significant modifications or adaptations.

2. **Scalability**: Layerium serves as a scalability solution by operating as a Layer 2 protocol. By utilizing Layer 2 technology, Layerium can significantly enhance the scalability of the underlying blockchain network. It achieves this by offloading transactions and computational processes to the Layer 2 layer, reducing congestion on the main blockchain and enabling faster and more efficient transaction processing.

3. **Security**: Layerium addresses scalability challenges through the implementation of Optimistic Rollup. This approach allows Layerium to provide scalability benefits while maintaining a high level of security comparable to the existing Layer 2 solutions. Optimistic Rollup ensures the integrity of transactions by leveraging fraud proofs, allowing for efficient and secure validation of transactions on the Layer 2 network.

4. **Modularity**: Layer 2 solutions, including Layerium, benefit from being modular in nature. Modularity allows for the separation of different components and functionalities within the Layer 2 protocol. This modular design facilitates flexibility, ease of maintenance, and the ability to upgrade or replace specific modules without affecting the entire Layer 2 system. Modularity also enables interoperability with other Layer 2 solutions, as different modules can be designed to interact and communicate seamlessly.

5. **Interoperability**: Layerium excels in inter-blockchain communication capabilities, enabling it to interact and communicate with any EVM-based blockchain. This interoperability enables Layerium to leverage the functionalities and resources of different blockchains, enhancing its versatility and expanding its range of applications. By facilitating inter-blockchain communication, Layerium promotes collaboration and synergy among various blockchain networks, fostering a more interconnected and efficient ecosystem.

4 The Layerium Protocol

Layerium's optimistic rollup Layer 2 solution incorporates a well-defined state machine that governs the status of each Virtual Machine (VM) within the system. This state machine plays a crucial role in the efficient and secure operation of Layerium as a layer 2 scalability solution. Here is an overview of the state machine and its functionalities:

1. Initialization:
   - The state machine initializes each VM with its initial state, including code, data, and storage.
   - VMs are created and configured with the necessary parameters and properties to ensure proper execution.
2. Computation and Execution:
   - VMs execute smart contracts and process transactions within the layer 2 environment.
   - The state machine facilitates the execution of EVM bytecode and performs the required computations and state updates.
   - Transactions are processed, and the VMs' internal states are modified accordingly.
3. Validation and Aggregation:

Archived copy · cryp2nova.com

- Once the computations are completed, the state machine validates the generated state updates and transactions.

- It checks for adherence to the Layerium protocol rules and verifies the validity of transactions.

- Validated state updates and transactions are aggregated, ready for further processing.

4. Optimistic State Transition:

- Layerium's state machine employs an optimistic approach, assuming the validity of state updates and transactions by default.

- This allows for efficient and fast processing without immediate on-chain verification.

- The state machine proceeds with state transitions based on these optimistic assumptions.

5. Dispute Resolution:

- In the event of a dispute or disagreement regarding the validity of a state update or transaction, the state machine triggers a dispute resolution mechanism.

- Parties involved can provide proofs and initiate the dispute resolution process to challenge the assumed validity.

- The state machine ensures a fair and transparent resolution process to address disputes effectively.

6. Finalization:

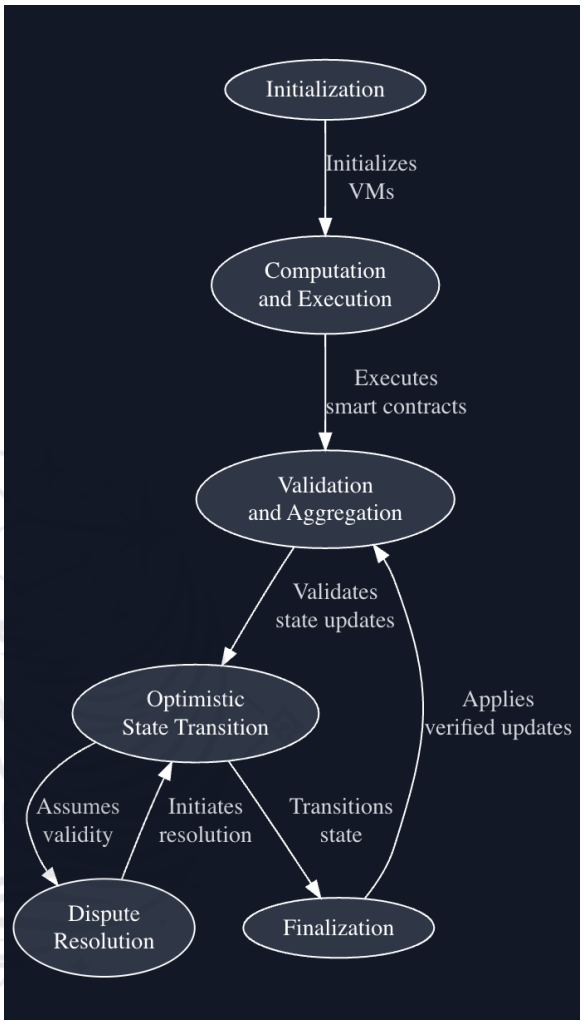
- Once a dispute is resolved and consensus is reached, the state machine finalizes the state of each VM.

- Verified state updates are applied to the Layer 1 blockchain, ensuring the integrity and security of the layer 2 solution.

- The finalization process ensures that the state of each VM accurately reflects the executed transactions and their outcomes.

The Layerium state machine is designed to optimize scalability by minimizing on-chain operations and leveraging the advantages of optimistic rollup technology. It provides a structured framework for managing the states of VMs, facilitating efficient computation, validation, dispute resolution, and finalization processes within the layer 2 ecosystem.

For more detailed and specific information about Layerium's implementation and the intricacies of its state machine, it is recommended to refer to official documentation, technical resources, or consult the Layerium development team.



4.1 The Layerium VM Architecture

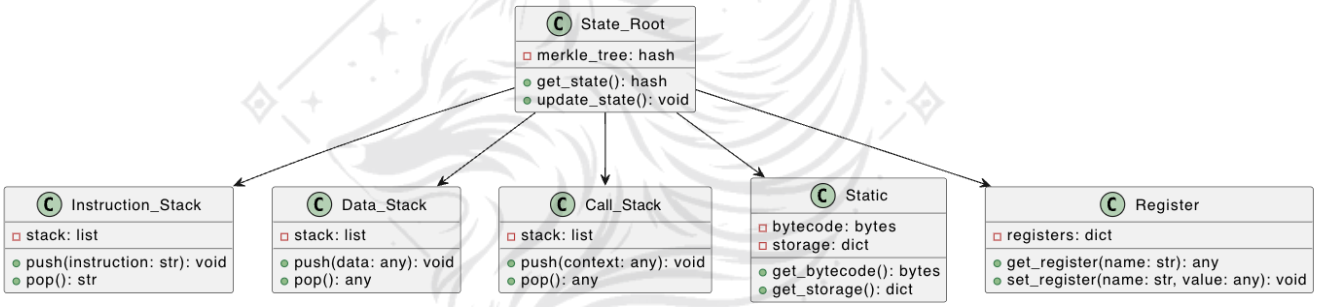
The Layerium VM is specifically designed to streamline the Verifier's task of checking one-step proofs, ensuring efficiency and simplicity. The VM architecture guarantees that the space required for representing a one-step proof and the time taken to generate and verify such a proof are bounded by small constants. These constants remain independent of the size and contents of the program's code and data.

To support constant-bounded proofs, Layerium takes an architectural approach that differs from offering a large, flat memory space. In the Layerium VM, a large, efficiently updatable hash

Archived copy · cryp2nova.com

of a flat memory space is not provided. This is because hashing such a space in a Merkle Tree style would require logarithmic proof space and time to prove and verify the memory state using Merkle proofs. Instead, Layerium employs a tuple data type that can store up to eight values, including nested tuples recursively. This allows for the construction of a tree representation similar to Merkle trees, but it is managed by Layerium code running within the VM. As a result, reading or writing a memory location only requires a logarithmic number of constant-time-provable Layerium instructions, rather than a single logarithmic-time provable instruction. Additionally, the Layerium standard library offers a convenient abstraction of a large, flat memory for programmers to use.

The VM architecture overview highlights the optimized operation of the Layerium VM, with a focus on its type system. The prototype implementation includes various types such as a special null value (None), booleans, characters (UTF-8 code points), 64-bit signed integers, 64-bit IEEE floating-point numbers, byte arrays with a maximum length of 32, and tuples. Tuples can hold any value, including other tuples recursively, allowing for the representation of complex tree data structures within a single tuple. All values are immutable, and the VM computes the hash of each tuple upon creation, enabling the (re-)computation of the hash for any value in constant time.



4.2 Offchain Virtual Machine

Layerium Layer2 utilizes the Optimistic Rollup approach to handle offchain Virtual Machines (VMs) in a secure and efficient manner, making it one of the safest Layer2 solutions available. Here's a description of how Layerium achieves this:

1. Optimistic Rollup: Layerium Layer2 leverages the Optimistic Rollup technology, which is a Layer2 scaling solution built on top of an existing blockchain, typically an EVM-compatible chain. Optimistic Rollup allows for offloading most of the computational load and storage requirements to the Layer2 solution while ensuring the security and finality of transactions through periodic verification on the main chain.

2. Offchain Virtual Machine (VM): Layerium implements an offchain Virtual Machine that executes smart contracts and processes transactions within the Layer2 environment. The offchain VM mirrors the functionalities of the underlying main chain's VM (e.g., Ethereum's EVM), providing compatibility and seamless execution of smart contracts.

3. Safety through Optimistic Execution: Layerium's safety is derived from the optimistic execution model of Optimistic Rollup. In this model, transactions are initially assumed to be valid and executed offchain within the Layer2 environment. The execution results, including state transitions and outputs, are recorded and aggregated.

Archived copy · cryp2nova.com

4. Fraud Proof Verification: To ensure the security and integrity of transactions, Layerium employs a fraud-proof verification mechanism. Periodically, these recorded execution results are batched and submitted as fraud proofs to the main chain for verification. The main chain validators check the validity of these proofs against the Layer2 state, ensuring that no fraudulent or invalid transactions have occurred.

5. Economic Incentives and Dispute Resolution: Layerium incentivizes honest behavior and discourages malicious activities through economic mechanisms. Participants who detect and provide valid fraud proofs are rewarded, while malicious actors attempting to submit fraudulent transactions are penalized. Dispute resolution mechanisms are also in place to resolve any conflicts or discrepancies during the verification process.

6. High Security and Finality: By relying on the underlying security of the main chain, Layerium achieves a high level of security for Layer2 transactions. The finality of transactions is guaranteed by the main chain's consensus mechanism, providing users with confidence in the immutability and integrity of their interactions within Layerium Layer2.

By combining the benefits of Optimistic Rollup, offchain VM execution, fraud-proof verification, economic incentives, and the security of the main chain, Layerium Layer2 offers a robust and safe solution for scaling blockchain applications. It allows for efficient execution of smart contracts offchain while maintaining the highest level of security and ensuring compatibility with existing EVM-based chains.

5. Summary

Layerium is a groundbreaking Layer 2 solution that addresses the scalability and privacy limitations of previous systems like Ethereum. It offers enhanced support for smart contracts by introducing a novel approach to consensus and execution called the Layerium Virtual Machine (VM). Here's a summary of how Layerium becomes the best Layer 2 solution:

1. Universality: Layerium enables the creation of smart contracts by utilizing code to define the behavior of a VM. It embraces universality, allowing any Ethereum Virtual Machine (EVM) to replace the existing infrastructure. This empowers the implementation of complex protocols and monetary features embedded within smart contracts.

2. Incentivized Off-chain Consensus: Layerium employs mechanism design to incentivize off-chain consensus among parties regarding the expected behavior of a VM. Verifiers in Layerium only need to verify digital signatures, significantly reducing the cost of managing contracts. Honest parties can progress the VM state on-chain even in cases where unanimous agreement cannot be reached off-chain.

3. Efficient Dispute Resolution: Layerium utilizes a challenge-based protocol to efficiently settle disputes about the behavior of a VM. Verifiers analyze a single instruction executed by the contract, penalizing any party attempting to deceive others. This ensures the integrity and trustworthiness of the VM's behavior.

4. Scalability and Privacy: By moving the verification of VM behavior off-chain, Layerium achieves significant improvements in scalability and privacy. Verifiers track only the hash of the VM's state, reducing computational and storage requirements. Contracts can execute privately by publishing saltable hashes of contract states, maintaining confidentiality.

5. Consensus Agnostic: Layerium is designed to be consensus-agnostic, seamlessly integrating with any consensus mechanism. It can work with a centralized publisher, quorum-based consensus, or even existing smart contract systems. This flexibility allows for easy adoption and compatibility with various blockchain ecosystems.

6. Trust and Security: Layerium ensures trust in contract execution by assigning managers responsible for overseeing the VM's behavior. If

Archived copy · cryp2nova.com

at least one honest manager exists, the VM adheres to its specified code. Verifiers enforce agreement among managers and penalize dishonest behavior, guaranteeing the security and reliability of the system.

Layerium's protocol and virtual machine architecture are explained in detail, along with a functional prototype implementation. By overcoming scalability and privacy limitations, incentivizing off-chain consensus, and prioritizing efficient dispute resolution, Layerium emerges as the best Layer 2 solution, offering enhanced smart contract support for a wide range of applications.



Archived copy · cryp2nova.com