Movement Network: High-Throughput Fast Finality Move-based Chains Secured by Ethereum Version 0.2.7

Movement Labs

January 23, 2025

Abstract. We introduce Movement Network, a secure and scalable network of Move-based chains secured by Ethereum, addressing the need for safer execution environments. At its core is the Move Stack, a modular framework for creating highly customizable Move-based chains.

The Movement Mainnet, our flagship general-purpose Move-based chain, spearheads the capabilities of the Move Stack that includes:

1. Move Executor: A *high-throughput* execution layer with the MoveVM, parallel execution and EVM compatibility for seamless integration with existing applications.
2. Modularity: Move Stack provides support for multiple DA services and sequencers e.g., Decentralized Shared Sequencer (DSS) to configure Move-based chains. A Move-based chain can be configured to achieve traditional Ethereum security guarantees with fraud proof/ZK-proofs or use our fast finality settlement mechanism (FFS).
3. Fast Finality Settlement (FFS): using a 2/3 super-majority mechanism, similar to Ethereum PoS and Polygon PoS, achieving confirmation times in seconds, by leveraging economic security from a network of validators while also profiting from Ethereum's security.

Movement Network integrates with our set of in-house services, which enables an ecosystem of next-generation interoperable chains. Chains benefit from DSS, which enables seamless cross-chain *interoperability*, enhances censorship resistance, and eliminates single points of failure.

DSS is secured through our multi-staking mechanism, which pools economic security across Movement Network and beyond, minimizes the infrastructure requirements and maximizes the sovereignty of each Move-based chain.

Table of Contents

Glossary

MoveVM	The virtual machine used for execution.
Move	The programming language for smart contracts on the MoveVM
Move Stack	The stack of tools, components and adapters for building and deploying custom Move-based chains.
Move Stack Chain	Blueprint for Move-based chains.
Movement Network	A network of Move-based chains.
Movement Mainnet	General-Purpose Move-based chain.
Move Executor	The module that enables the execution of both MoveVM and EVM bytecode.
DSS	Decentralized Shared Sequencer.
FFS	Fast Finality Settlement mechanism.

1 Objectives & Motivation

Blockchain technology provides a decentralized ledger where participants can transact without relying on a central authority. The Ethereum Network [5] was the first to propose a versatile *world computer* [12], with *programmable* transactions called *smart contracts*, and the ability to implement arbitrary business logic that go beyond simple currency or assets' transfers (pioneered by the Bitcoin network [11]).

Wide adoption of the Ethereum-based technology is still hindered[1] by several limitations, such as high latency to transaction finality, low throughput (expressed in *transactions per second, TPS*), and widespread security vulnerabilities in *decentralized Applications* (dApps).

Ethereum mainnet, with its unmatched level of *Total Value Locked* (TVL), offers the highest level of crypto-economic security, which creates an unrivaled incentive to capitalise on its best-in-class security guarantees.

Several solutions have been proposed to address the above limitations of the Ethereum network. The most popular ones being *rollups*, which are solutions that bundle multiple rollup transactions into a single Layer 1 (L1) transaction. Note that we use L1 and Ethereum interchangeably in this paper. Rollups *settle* transactions on the Ethereum mainnet, thereby inheriting its high level security. Rollups have been successful in addressing some of the scalability limitations of Ethereum, but they have not been able to fully address the security vulnerabilities of dApps, nor the latency issues.

Some of the original design choices of Ethereum, inherited by Ethereum rollups, have made it a very complex infrastructure, making it difficult to address the current limitations. For example, the EVM is not designed to prevent security vulnerabilities[2], unintended assets's duplications or re-entrancy attacks [7, 8]. The *global storage* model of the EVM itself makes it hard to parallelize the execution of transactions, which severely limits the scalability of the network. However, the design choices and limitations of the Ethereum network offer a good opportunity to reflect on the current technology and see how to improve it.

Recently new paradigms have emerged for the execution layer, offering new execution environment and programming languages. An example for the latter is Move, originally developed at Facebook (Diem/Libra project), a next generation highly secure and efficient Web3 development platform, providing principled solutions to security vulnerabilities and scalability. It empowers Web3 developers with modern tools to tackle the challenges of deploying reliable, cost-effective and efficient dApps. Move and the MoveVM are used in L1 chains, such as Aptos [2], Sui [4, 13], and 0L [1, 10] and has demonstrated very promising results in terms of security, low latency (sub second finality) and throughput (sustained reported throughput of 30k TPS and 160k theoretical, compared to a typical 20 TPS for Ethereum).

The Move language [3] proposes a new approach to Web3 development and was designed to address the current blockchain technology limitations. Move introduces a novel programming paradigm known as *resource-oriented programming*, enabling parallel execution of transactions in the MoveVM, together with strong security guarantees using *formal verification*.

[1] This is also true for many networks like Solana.

[2] According to DefiLlama, hacks have cost more than $680m since the beginning of 2024.

One of the main challenges for the Move community is to build an ecosystem that is crypto-economically secure, but for the time being, the L1 chains Aptos, Sui and 0L have not attained the TVL[3], liquidity and developer activity levels[4] of Ethereum yet. This is a compelling opportunity for our Move community to bring together the highly crypto-economically secure Ethereum platform and Move/MoveVM, the most technologically advanced Web3 development platform.

Our proposal is to build a network of interoperable chains to bridge the gap between two ecosystems, Move and Ethereum, where the most advanced Web3 technology meets the most crypto-economically secure L1 chain.

Our contribution. In Section 2, we introduce Movement Mainnet, a general purpose Move-based chain. The Movement Mainnet architecture is extracted from the more general Move Stack Chain blueprint framework, described in Section 3, which is shared by all Move-based chains in our network. Section 4 describes FFS, our Fast Finality Settlement mechanism. In Section 5.1, we introduce Movement Network– the network of Move-based chains – and also, DSS, the decentralized shared sequencer that enables cross-chain interoperability.

2 The Movement Mainnet

Movement Mainnet is Movement Labs' general-purpose chain (Figure 1). It is the first sidechain atop Ethereum that will integrate Celestia for data availability, decentralized shared sequencing, optimistic rollup with option for dual-settlement with FFS, and the Move Virtual Machine (MoveVM) for execution, which offers unparalleled transaction throughput.

This instantiation of the Move Stack Chain framework will allow developers to create high-performance, consumer-focused applications with minimal resource expenditure.

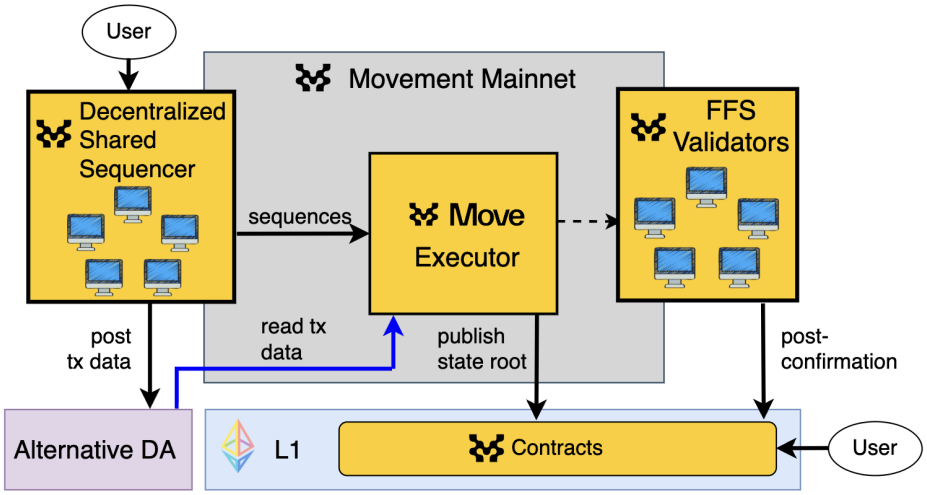

Fig. 1. Movement Mainnet architecture. Transactions are sequenced by the Decentralized Shared Sequencer (DSS). The Fast Finality Settlement (FFS) with Validators provides fast settlement.

[3] $60b for Ethereum vs $550m for Sui and $360m for Aptos according to DefiLlama.
[4] More than 13K Solidity devs vs 400 Move devs according to Electrical Capital.

2.1 Original components

We develop three core components, that we capitalize on in the Movement Mainnet.

1. The Move Executor (Section 3.2), which aims to support MoveVM and EVM transactions, enabling Web3 developers to deploy smart contracts in Move and EVM bytecode on a single network.
2. A fast finality settlement module, FFS (Section 4) which connects to a *validator network*, facilitating fast settlement when compared to optimistic and validity settlement mechanisms.
3. A *decentralized shared sequencer* module, DSS (Section 5.1) which ensures customizable transactions ordering, with templates from a set of approaches, such as fair transaction ordering for mitigation of front-running attacks and enhancement of censorship resistance.

First, Movement Mainnet supports both MoveVM and EVM transactions. This is a unique feature of our architecture, as most rollups only support one type of transactions. This feature is critical to allow Web3 developers to onboard the Movement Mainnet quickly. It is also a significant advantage for the Movement Network (Section 5.1) as it allows developers to leverage the existing EVM dApps and extend them benefiting from the advanced features of the Move platform. For instance, standard EVM contracts like ERC-20 can be deployed on Movement Mainnet and new and secure Move dApps can be developed to *interoperate* with them.

The Move Executor supports both MoveVM and EVM transactions, allowing Web3 developers to deploy smart contracts in both Move and EVM bytecode on the same network. It provides a unique infrastructure where Web3 developers can migrate or extend their existing EVM dApps with the more secure and efficient Move framework.

Second, we introduce a *fast settlement* mechanism FFS (Section 4), an alternative settlement mechanism to validity and optimistic rollups. FFS relies on a set of validators who stake native tokens. The validators have to confirm the correctness of the new Move-chain state by forming a super-majority (e.g., 2/3 of the total stakes) to validate the new state.

The fast settlement mechanism offers fast finality and also contributes to increasing the utility of the Movement Mainnet native token.

Third, by utilizing the DSS (sequencer), Movement Mainnet builds on an alternative to sequencing marketplaces, such as Espresso, Astria, or L1-based sequencing. This is a deliberate choice to ensure the sovereignty of Movement Mainnet (and the Move-based chain network more generally) and to provide a fast, customizable and verifiable ordering of transactions.

Another consideration is the complexity of (decentralized) shared sequencing marketplaces especially when it comes to distributing rewards and penalties, which are hard problems currently lacking good solutions. A sovereign sequencer module offers a solution where fees can be collected by the chain rather than by an external component (marketplace), thus positively impacting the utility of the native token of the chain. Shared sequencing aims to provide some level of *interoperability* between different chains and it is discussed in Section 5.2.

The DSS provides a sovereign, fast, customizable and censorship resistant ordering of transactions, enabling interoperability and increasing the utility of the native token of the Movement Mainnet.

2.2 Original framework

We develop am original framework, Move Stack, that comprises several components:

1. The Move Stack Core (Section 3.3) that enables to create customizable chains, with the Move *Executor* at the heart, fast finality settlement (FFS), and decentralized shared sequencing (DSS) that enables interoperability.
2. The Move Stack Binder (Section 5.1) which provides a framework to deploy and join the Movement Network.

3 The Move Stack Chain framework and the MoveVM

We introduce the Move Stack Chain framework (Figure 2), our general-purpose Move-based chain schema for Ethereum-secured chains. Move Stack Chain is a modular architecture where components can be configured to meet customers' needs with the most suitable, cost-efficient and performant components.

In the Move Stack Chain framework, we offer fraud proof, ZK-proof, and a *fast finality settlement* (Section 4), where a network of *validators* who have staked native tokens, validate the correctness of the new chain state and the availability of the data, and provide ultra-fast reliable finality with high-economic security.

We show a categorization of Move Stack Chain configurations in Figure 5 and provide examples in Table 1.

3.1 Architecture of the Move Stack Chain framework

Move Stack Chain is a generic architecture for creating *Move-based chains* that use the Move Executor (Section 3.2, Figure 2, page 7).

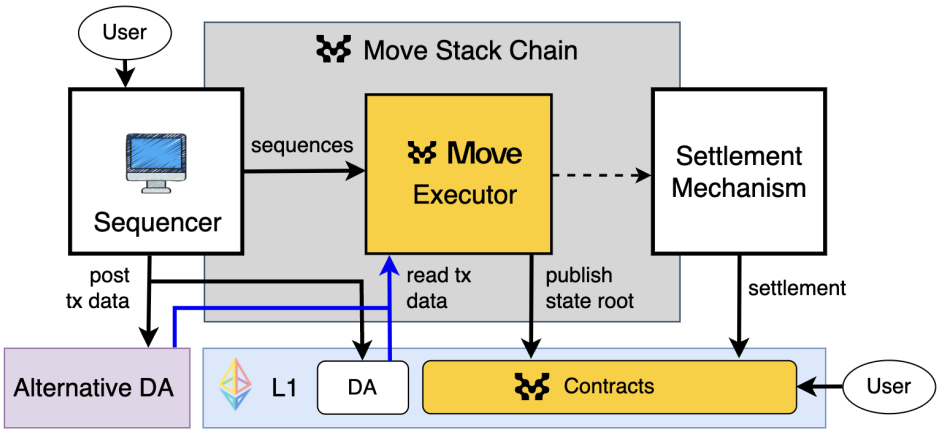

Fig. 2. The generic Move Stack Chain architecture with the Move Executor. Components in yellow are fixed in the architecture whereas components in white are customizable.

The Move Stack Chain generic architecture has a set of core components:

- *Executor* to process transactions and generate new blocks.
- Connection to a *Sequencer* to order transactions.
- Connection to a *Data Availability* (DA) service to ensure transaction data accessibility to the settlement mechanism.
- Connection to a *Settlement Mechanism*: to verify transaction execution correctness.
- *Bridge contracts* on L1 and a Move-based chain for asset deposits and withdrawals between L1 and the chain.

Move Stack Chain can be used to create Move-based rollups or chains, for instance the Movement Mainnet (Section 2) which targets a specific instantiation of this generic architecture.

The lifecycle of a transaction within a Move-based chain is as follows:

1. A transaction *tx* is submitted to the mempool (client, top of Figure 2).
2. The *sequencer* extracts a *batch* *b* of transactions from the mempool, including *tx*, and orders them. The sequencer publishes the transactions data of *b* to the DA service (L1 or an alternative DA).
3. The *executor* processes the transactions. This results in a new chain state (and a short commitment of it, known as a *state root* *s*) that is also published to L1 in the bridge contract.
4. The *settlement* of the transaction *tx* happens when the L1 validating contract verifies or approves the new state. This can be done with ZK-proofs, by passing the challenge period successfully in the optimistic setting, or when the quorum certificate (2/3 supermajority) is validated in FFS.

3.2 The Move Executor

The execution layer of all Move-based chains is the MoveVM. The Move Stack provides an execution module, Move Executor, that can execute MoveVM bytecode and EVM bytecode. This module is at the heart of our architecture and not configurable.

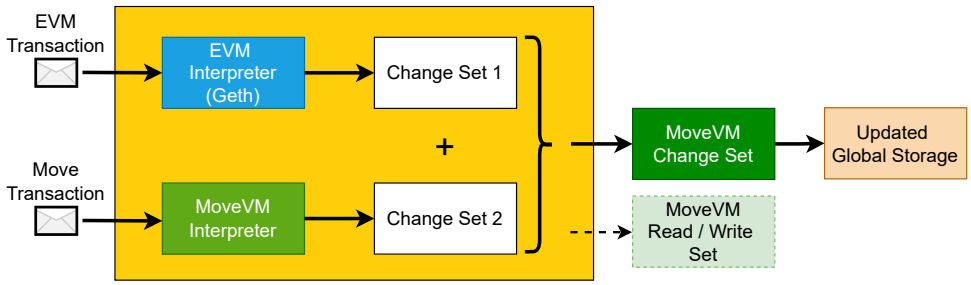

Fig. 3. The Move Executor. Transactions are routed to an interpreter depending on their types, and the effects on global storage are safely merged following the MoveVM rules using change sets.

The Move Executor module supports the execution of both MoveVM and EVM bytecode on the same chain.

Figure 3 gives a high-level view of the Move Executor module. Transactions are triaged from the mempool according to their types, Move or EVM. A corresponding VM (MoveVM

or Geth) executes a transaction. In MoveVM this results in a *change set* that is later applied to the global storage. In Geth a transaction can be executed and modify the global storage, but it can also be *traced* instead to produce a change set that can be applied to the global storage. This provides a way to get a common format for the update of the global storage by both Move and Ethereum transactions. Another nice feature is that read/writes sets can also be extracted with Geth and can be used seamlessly in *BlockSTM*, the MoveVM's built-in parallel execution engine.

The Move Executor re-uses existing EVM interpreters and integrates seamlessly with MoveVM to benefit from its parallel execution engine, thereby providing a *parallel* EVM. Moreover, using an existing EVM interpreter under the hood ensures that Move-based chains are EVM-equivalent and that executed EVM bytecode executed has the same behaviour as on L1.

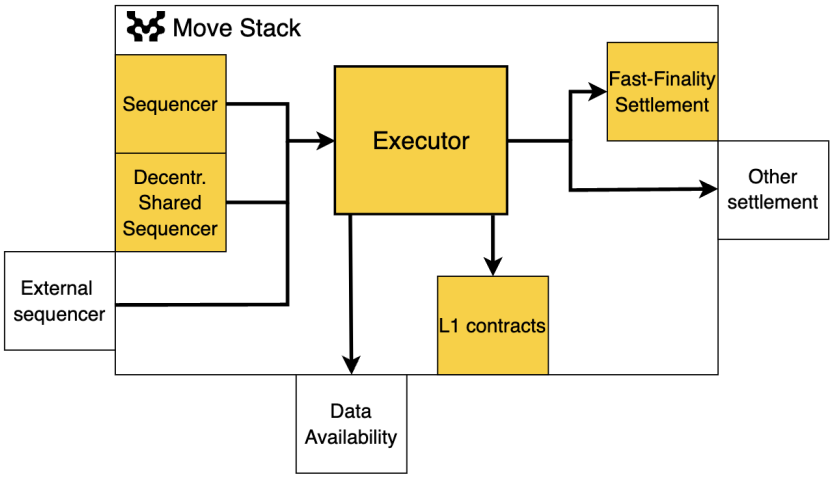

Fig. 4. The Move Stack provides a set of components (yellow boxes) along with adaptors (white boxes). To create a chain instance from the Move Stack Chain blueprint a component is selected (i.e. configured) from the available options.

3.3 Move Stack Core

The Move Stack Core provides support to create and deploy Move-based chains. Developers are empowered to quickly spin up new chains in the network, by selecting suitable components from the provided options in the Move Stack, see Figure 4.

The (configurable) components of the Move Stack include:

- *Sequencer:* A chain can opt-in for the default DSS, decentralized shared sequencing service, but is provided with a default self-reliant sequencing mechanism.
- *Data Availability:* We plan to support Ethereum EIP-4844 blobs, and major DA solutions (e.g., 0G, Avail, Celestia, EigenDA, Near).
- *Settlement mechanisms:* Optimistic (fraud proof), ZK (validity proof), FFS (attestations).

Chain name	Type	Sequencer	DA	Settlement
Movement Mainnet	General Purpose	DSS	Celestia	FFS chain
Gamechain	Gaming	Centralized	EigenDA	ZK rollup
Finchain	DeFi	DSS	Ethereum	FFS chain
Duckchain	Art	Centralized	EigenDA	Optimistic rollup

Table 1. Example of Move-based chain configurations

Table 1 illustrates a diverse range of Move-based chains within the Movement Network (Section 5.1), showcasing the extensive customization possibilities based on specific requirements. Moreover, using Move Stack to deploy a Move-based chain promotes standardization across crucial infrastructure components, including wallet software, developer APIs, and block explorers. This standardization enhances interoperability and significantly improves the developer and user experience across the Movement Network ecosystem.

4 Fast Finality Settlement (FFS)

The Move Stack Chain framework is modular and we can customize the settlement mechanism. As a result the chain can be secured through a *staking* mechanism. This staking mechanism provides *fast finality*[5] with high crypto-economic security, and in this configuration, a Move Stack Chain is technically a sidechain.

Since our Fast Finality Settlement (FFS) approach is similar to how Ethereum (or Polygon PoS) works in terms of security, we will recall basic concepts relevant to this context, specifically Ethereum's security model (Section 4.1) and the security of ZK and optimistic rollups (Section 4.2). We then introduce the concepts of Postconfirmations and validator-confirmations[6], a simple mechanism for implementing FFS (Section 4.3). We compare the level of security provided by FFS to optimistic and validity rollups, see also Figure 5. Finally, we propose a combination of optimistic- or ZK-based rollup settlements with the FFS, called dual-settlement, to provide both Ethereum security and economically protected FFS guarantees (Section 4.6).

4.1 Ethereum settlement and security

Ethereum's consensus is a *Proof of Stake* (PoS) protocol, and validators have to stake some assets (32 ETH) to be incentivized to attest honestly about the status of a state transition. A validator that would be Byzantine (malicious) bears the risk[7] of being slashed of their stake. On Ethereum mainnet (L1), a state transition (creation of a new block) is *final* once it has received *enough attestations* from the validators. Enough stake is usually understood as 2/3 of the total stake – *a super-majority* – of all the validators. As a result, under the assumption that less than 1/3 of the validators are malicious,[8] if more than 2/3 of the validators have attested for a state transition, it must be correct as at least one the validators in this 2/3 is not Byzantine (it is honest).

[5] *Finality* is the time for a transaction to become practically irreversible, subject to the preservation of security of the underlying system.

[6] This is a 2/3 super-majority mechanism similar to Polygon PoS.

[7] There are two slashing conditions on Ethereum, and if a validator is caught violating one of them, they can be slashed.

[8] And each validator stakes the same amount.

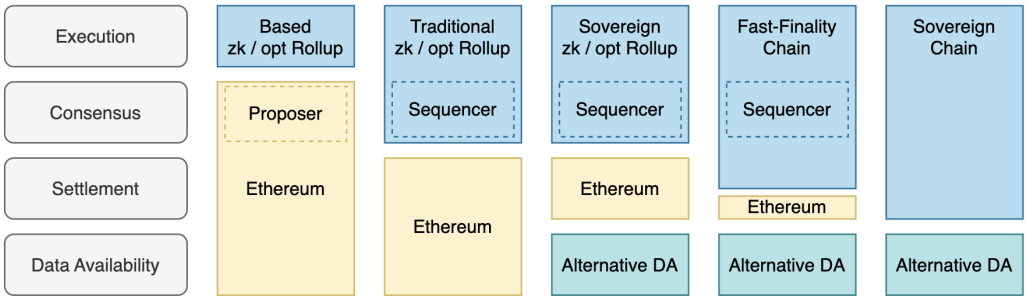

Fig. 5. Categorization of rollups and chains. Move-based chains are configurable and can in principle take on any of the above forms. A Move-based chain can be configured with an FFS settlement or dual settlement mechanisms including optimistic approaches.

The *security* provided by the PoS mechanism is two-fold:

- *liveness*: in order to prevent a super-majority to attest a correct state transition, an adversary would have to control more than 1/3 of validators. This is considered *infeasible* when the total stake in the system is large (the probability that this happens is negligible.)
- *safety*: in order to force an incorrect state transition (e.g., double spending) an attacker would need to control 2/3 of the validators. Similarly to the previous point, this is considered infeasible given a large enough stake.

Ethereum security: The *level of security*, i.e., the liveness of the Ethereum network and the safety (correctness of a state transition), increases with the total stake in the system. The higher the total stake, the more secure the network is. The level of security provided by the Ethereum network is commonly referred to as *Ethereum security*.

4.2 Security of validity and optimistic rollups

There are two main types of rollups, *validity (ZK) rollups* and *optimistic rollups*. Both settle on a Layer 1 (e.g., Ethereum mainnet) but use different settlement mechanisms.

In a ZK-rollup, settlement happens when the ZK-proof of the state transition is *accepted*. This is done by submitting a *verification* transaction to the L1 *verifier* contract. Since the verifier is implemented as a contract on L1, the security level of the verification phase is Ethereum's security. Under the assumption that the ZK-proof system (proof generation and verifier contract) is correct, the ZK-proof is accepted *if and only if* the state transition is correct, hence

ZK-rollup security: The level of security of a ZK-rollup is the same as Ethereum's security: a ZK-rollup *inherits* Ethereum's security.

In an optimistic rollup, finality of transactions – after submitting the data and state commitments to Layer 1 – is achieved at the end of a time window, called the *challenge period*. It follows that security is *conditional*: the settlement happens if at the end of the challenge period (usually 7 days), no *disputes* have been *successful*. A dispute is a way of challenging a state transition. Validators can *raise* a dispute against a state transition if

they think that it has been computed incorrectly. A *trusted dispute resolution* mechanism resolves the challenge: if the challenge is successful, the submitter of the incorrect state transition is slashed. Otherwise the challenging validator is slashed. Assuming at least one honest validator (e.g., *watchtower*) re-executes each rollup state transition, it is impractical for the rollup to submit an incorrect new state. The level of security depends on *where* the dispute is settled. If it settles on Ethereum mainnet via a contract (e.g., as in [14]), and the contract that resolves the dispute is trusted (no bugs) then the security of the dispute resolution is Ethereum's security.

Optimistic rollup security: The security of the dispute resolution mechanism of an optimistic rollup can inherit Ethereum's security. But if no validator checks a state transition before the end of the challenge period, then the level of security is zero.

Finality of a state transition (i.e., finality of a transaction) on Ethereum mainnet is in the order of 12 minutes. On average the time to generate a ZK-proof is in the order of 10-15 minutes, and hence the finality of a transaction on a ZK-rollup is expected to be in the order of 20-25 minutes. For optimistic rollups, the standard challenging period is 1 week. In both cases, the time it takes to finalize a transaction can be prohibitively large for some, if not many applications.

4.3 Security of Fast Finality Settlement (FFS)

As discussed in the previous sections, validity (zero-knowledge proof, ZKP) and optimistic (fraud-proof, FP) rollups can finalize transactions with Ethereum security within approximately 30 minutes and 1 week, respectively. However, until a transaction is finalized, assurance about its validity and result (success or failure) is limited. This can be a limiting factor for many types of DeFi applications. An intermediate level of economic security but with fast confirmation, can be provided via FFS.

FFS provides security through a Proof of Stake (PoS) protocol, similar to Polygon PoS. In a PoS protocol, validators stake some assets (e.g., in native chain tokens) to be incentivized to attest honestly about the status of a state transition. If they are dishonest (they accept incorrect state transitions or reject correct state transitions) their stakes can be slashed. If they are honest validators, they are rewarded for their activity. A network of *validators* can then provide fast and economically backed confirmations of correctly executed blocks. More precisely, the role of a validator is to confirm that the execution of a state transition is correct.[9]

A state transition (that corresponds to the execution of a set of transactions) is *finalized* (irreversible) on a Move-based chain that utilizes *only* FFS when *enough* validators have confirmed the correctness of the state transition. For the sake of simplicity, we assume all the validators stake the same amount and *enough* means more than 2/3 of the validators. The entire stake value is called *chain-stake*.

If the chain is complemented through a ZKP or FP, finality is achieved when this type of settlement is completed. The user can then decide whether it is good enough to assume irreversibility of the transaction through FFS or to wait for the ZKP or FP to complete to get the L1 (Ethereum) level of security.

Figure 6 illustrates the process of FFS, and the time to finality of a transaction.

[9] w.r.t. the semantics of the execution layer i.e. MoveVM.

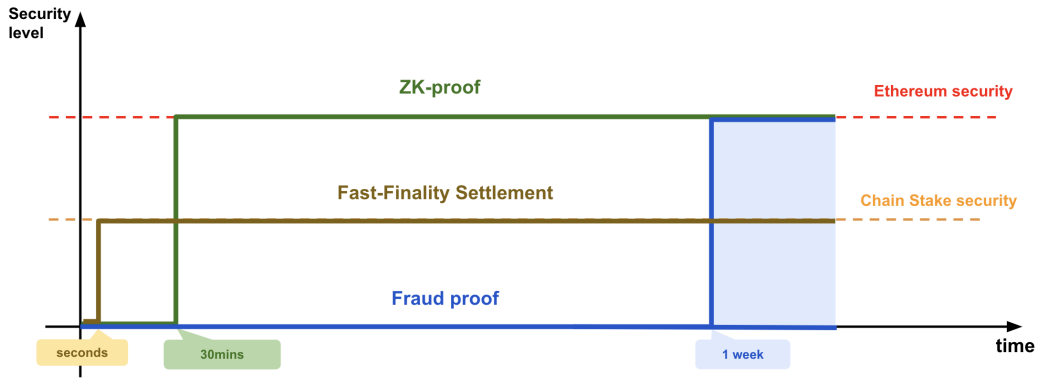

Fig. 6. Security levels and time to finality. The time to finality does not include L1 (Ethereum) average finalization time which is 13 minutes. Times displayed are indicative and may vary.

Comparison with optimistic and ZK-rollups. The security of an optimistic rollup inherits L1 (Ethereum) security *under the condition* that an honest validator raises a dispute for each incorrect state transition. However, at present, optimistic rollups limit the list of challengers to reduce the risk of delay attacks in which an adversary could open as many disputes as they are willing to forfeit bonds for.[10] All of the above impose significant trust assumptions onto the user. Moreover, slashing penalties may not be economically large, compared to total stake protection as is the case in the validator network.

In contrast to ZK-rollups, a Move-based chain that employs only FFS does not require expensive proof generation equipment. However, the most significant improvement delivered by a mechanism like FFS is the reduction in latency compared to both optimistic and ZK-rollups. Since attestations can be delivered in the order of seconds, we can provide fast finality guarantees and substantially improve user experience. This compares to order of minutes in the ZK setting and days in the optimistic setting.

FFS is instrumental to interoperability and atomic cross-chain transactions where a fast settlement time is required. Both optimistic and ZK-rollups lack in that respect. Hopefully, ZK-proof technology that permits real-time proving with specialized hardware, will be widely available in the near future, however, it is not clear at what point in time this is the case. Regarding optimistic rollups, they have an inherent requirement of extensive challenge periods (up to a week) to account for social engineering and attack vectors. In contrast, FFS-based chains can provide finality guarantees within seconds.

An FFS-based chain can be more secure than an optimistic rollup (e.g. if watchtowers are not sufficiently funded or live) and has faster finality than a ZK-rollup. In principle, if the total stake of the validator network is greater or equal to Ethereum validators' total stake, then the FFS-based chain even reaches L1 (Ethereum) economic security level. The overall security of the FFS approach depends on the total stake of validators. The staking, rewarding and supermajority verification steps inherit L1 (Ethereum) security.

We discuss a generalization of the staking mechanism, multi-asset staking, in Section 5.3.

[10] https://docs.arbitrum.io/how-arbitrum-works/bold/gentle-introduction. Accessed on 2024-07-10.

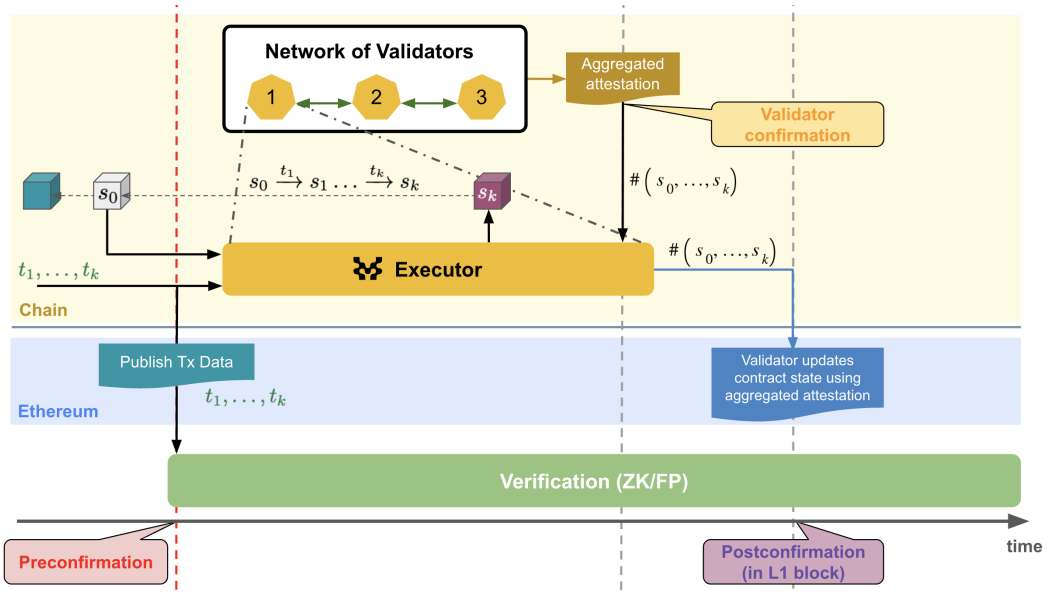

Fig. 7. Confirmation stages of a transaction with Postconfirmation and validator-confirmation.

4.4 Postconfirmation

Postconfirmation is an implementation of FFS.

First we provide a definition of *confirmation* on L1, that differs to that of L1-finalization. (We do not deem it useful to define confirmation the same as finality).

> We say a transaction is *confirmed on L1*, or *L1-confirmed*, when it is included and successfully executed through a valid L1 block.

We note that an L1-confirmed transaction can still be reverted if the block that contains the transaction is orphaned. Hence it is important that the L1-block that contains the Postconfirmation is also finalized on L1, before considering the Postconfirmation as final.

Postconfirmations are different to preconfirmations. Preconfirmations are promises that a transaction will be included (or successfully executed) in the next block(s). Typically a preconfirmation is also protected by reputation or single-actor stake. In contrast Postconfirmations provide a guarantee that a new block is correct, with high crypto-economic protection through an entire validator set. It is also not a replacement for the complex execution tickets mechanism, as it does not provide a way to influence the creation of a block, but rather to report on the correct execution of a block. In Figure 7 we depict the confirmation stages of a transaction.
In the first iteration the mechanism is as follows:

- A set of validators stake assets in a L1 contract `StakingK`.
- For a given state transition, (this may include several blocks from the chain), the validators attest by committing on a contract `PostconfirmationK`.
- When enough validators –more than 2/3 of the total stake– has attested the contract `PostconfirmationK` accepts the new state root.

Due to cryptographically protected signatures, a Byzantine validator cannot forge/tamper with the signed attestations. Assuming less than 1/3 of the validators are Byzantine, and due to the 2/3 majority requirement, there cannot be a malicious actor who could submit

enough attestations supporting an erroneous state transition. Moreover, during synchronous periods liveness is preserved as 2/3 of stake are also sufficient to complete the attestation process.

The verification that the threshold of 2/3 of the validators have confirmed a state transition is performed by contract `PostconfirmationK`. As a result the verification step inherits L1 (Ethereum) security. The staking/slashing/rewarding functions are also executed on L1 with the same security level.

Once a 2/3 super-majority is reached, the `StakingK` contract can slash the validators that attested for an incorrect state, as they are dishonest (under the assumption that at most 1/3 of validators are Byzantine).

Efficiency improvements. To make the attestation process more efficient, we can employ validator-confirmations, see Section 4.5. The mechanism could then be adapted on L1 to

- A set of validators stake assets in a L1 contract `StakingK`.
- For a given state transition (this may include several blocks from the chain), the validators collect the validator-confirmation.
- When an validator has collected the validator-confirmation they can submit it to contract `PostconfirmationK`. This reduces the number of L1 transactions needed to record the attestations.
- The contract `PostconfirmationK` verifies that the attestation signatures are valid, unique, and account for more than 2/3 of the total stake. The state transition becomes postconfirmed.

4.5 Validator-confirmation: Aggregating Attestations

Validator-confirmation is part of the implementation of FFS and complements the mechanism of Postconfirmation.

Postconfirmation provides increased security in FFS since it utilizes the L1 (Ethereum) security for the super-majority verification and staking process. However, the frequency and latency of Postconfirmations are limited by the block time, latency and cost of the L1. To make the attestation process more efficient and provide confirmations off-L1, we can require the validators to run an L1 light client, to learn about the state of `StakingK` and determine themselves how many validators are *active*. An active validator is a validator that has an active stake. The validators can then broadcast their votes for a new block to the entire validators' network. The validators can record and aggregate signed attestations. When one of the validators has determined that the 2/3 super-majority is reached, they can provide an aggregated (signed) attestation.

Since this approach provides a confirmation of the state transition by the validators on the chain level, this is a type of confirmation.

We say a transaction is *confirmed by the validators*, or *validator-confirmed*, when 2/3 of the total stake has been reached through aggregated attestations.

Furthermore, since the approach is not limited by the block time, latency and cost of the L1, confirmations can be obtained within seconds and at high frequency.

As validators rely on a recent state of `StakingK` contract (to compute what the 2/3 majority threshold is), we have to prevent validators from withdrawing their stakes too quickly. This is to ensure that a dishonest validator cannot wrongly/dishonestly attest and then withdraw their stakes before being slashed. We can lock the stakes for a pre-defined amount of time (a few epochs).

4.6 Dual-settlement

While FFS provides a rapid and economically robust form of transaction finality, a Move-based chain using FFS provides finality for a sidechain. However it can be further strengthened by integrating it with the proven security guarantees of optimistic and ZK-rollups to satisfy the requirements of a Layer 2. By layering these approaches, we can offer a dual-layer security model (i.e. *dual-settlement*) that leverages the strengths of both systems.

In the combined approach, FFS delivers rapid confirmation that is backed by the economic security of staked validators. It also profits from L1 (Ethereum) security if increased security is required, albeit with the typical latency associated with these methods.

This dual-layered finality model operates as follows:

- **FFS Layer**: Validators within the FFS framework rapidly confirm the correctness of state transitions, providing an initial layer of assurance of successful transaction execution that is economically secure and swift, thus enhancing user experience by reducing waiting times.
- **Optimistic/ZK-Finality Layer**: The transaction data is also processed through the second settlement mechanism — either FP or ZKP - on the L1 (Ethereum). This ensures that even if FFS is compromised (e.g., due to a significant, albeit improbable, collusion of validators), the transaction still benefits from Ethereum's robust security guarantees. This enhanced security comes at the cost of increased latency, as the finality of the transaction is subject to the FP or ZK verification process.

By combining these mechanisms, the system offers a highly secure and efficient transaction confirmation process:

- **Improved User Experience**: Users benefit from the fast and economically secure finality provided by FFS, without sacrificing the long-term security provided by Ethereum's settlement layer.
- **Flexibility and Resilience**: This approach allows the system to adapt to various security needs, offering a balanced trade-off between speed and security based on the specific requirements of different applications.
- **Enhanced Security**: The integration of two independent finality mechanisms significantly reduces the probability of a successful attack, as an adversary would need to compromise both the validator network and the primary settlement process.
- **Improved Slashing**: The primary settlement process ensures that in the case of a successful attack on FFS even a dishonest super-majority of validators can be slashed, thus further disincentivizing malicious behavior.

In conclusion, this dual-layered finality approach provides the best of both worlds: the swift and economically backed assurances of FFS, combined with the well-established security of L2s, such as optimistic or ZK-rollups. This hybrid model is particularly advantageous for applications requiring both immediate transaction confirmation and the highest possible security standards.

5 The Movement Network: a network of application-specific chains

Application-specific chains are becoming the norm in the blockchain world. This is driven by the fact that applications like DeFi, gaming or supply chain applications have different requirements for latency and throughput. Privacy or proprietary requirements may also need to isolate a chain and its dApps from others. As a result, app-specific chains are proliferating across L1 networks like Avalanche, Cosmos, and Polkadot.

We can take advantage of the modularity of our architecture (Section 3) to cater for specific needs, while at the same time providing cross-chain interoperability and shared liquidity. This is achieved by creating a network of Move-based chains, called Movement Network. By sharing the same modular architecture, the chains in Movement Network are equipped with increased interoperability, can share the same bridge and Data Availability layer, and can profit from the fast settlement provided by the Move Stack, see Figure 8.

This design choice is consistent with the other Layer 2 ecosystems, including Optimism Superchain, Arbitrum Orbit, Polygon Supernets, zkSync Elastic Chain or Starknet appchains (layer 3).

The Move Stack offers a cost-efficient and secure way of deploying new application-specific Move-based chains in the Movement Network. Moreover, by being part of the Movement Network, these chains are provisioned with cross-chain interoperability, as well as shared liquidity between them.

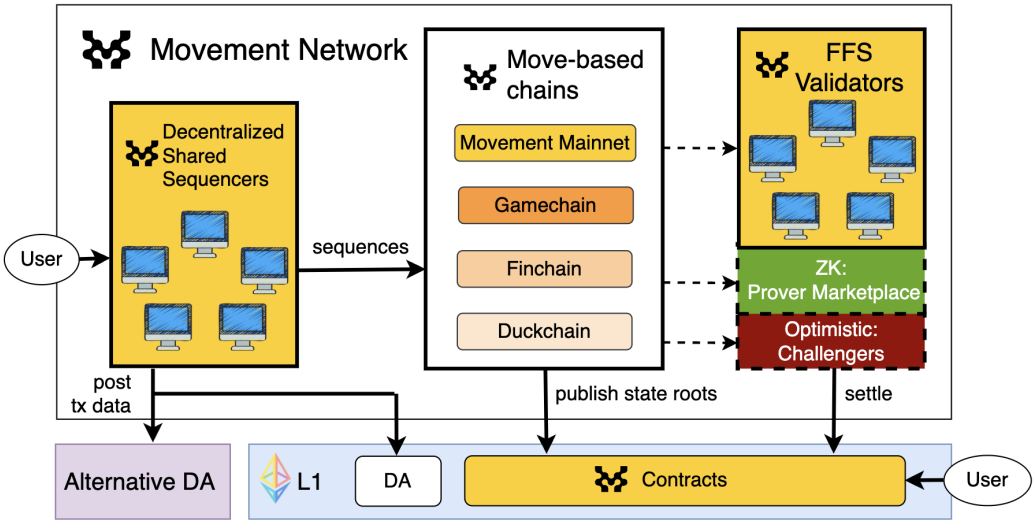

Fig. 8. The Movement Network: a network of interoperable Move-based chains.

5.1 The Move Stack Binder

The Move Stack Binder is part of the Move Stack and is designed to seamlessly integrate with our suite of in-house services and support a network of interoperable chains. Move Stack Binder facilitates the creation of a dynamic ecosystem where various chains can operate efficiently and interact with one another. Its designed to meet the diverse needs of modern blockchain applications, offering enhanced interoperability, security, and resource efficiency.

The Move Stack Binder provides a framework to deploy and join the Movement Network (see Figure 9):

- DSS: A decentralized shared sequencer network that ensures seamless cross-rollup interoperability and enhances network security.
- **Validator Network:** A Proof-of-Stake based attestation system to ensure secure and fast operation of FFS, see Section 4.

- **Multi-Asset Staking:** Allows stakers to use multiple assets for staking, increasing flexibility and economic security.

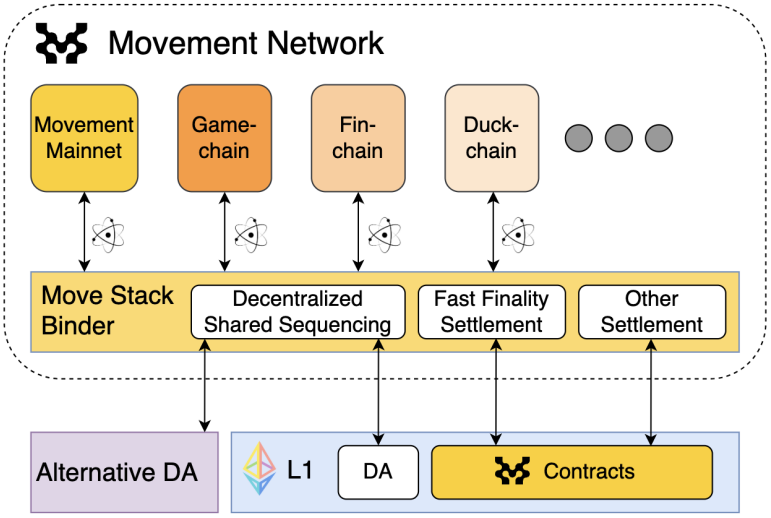

Fig. 9. The Movement Network. This figure is equivalent to Figure 8 but highlights the layered approach and which components the Move Stack Binder connects between the Move-based chains and the L1.

5.2 DSS: Decentralized Shared Sequencer

DSS serves as a decentralized and shared sequencer for the Movement Network, diverging from the centralized sequencers commonly used in most rollups. This decentralized design enhances network robustness by eliminating single points of failure, promotes fairness and censorship resistance in transaction ordering, and allows permissionless participation [9].

To achieve consensus on transactions ordering, we employ a highly scalable, performant Byzantine Fault Tolerant (BFT) protocol.

Centralized sequencers may offer faster preconfirmations than decentralized sequencers due to their centralized nature. However, being built with a highly scalable BFT consensus mechanisms and an efficient mempool mechanism, DSS can provide fast preconfirmations with only marginal increase in times, with the additional benefit of being economically backed, rather than based on trust. In regards to throughput a centralised sequencer could have less throughput limitations, however significant advances have been made on modern BFT protocols and which enable throughput levels that more than satisfy requirements. Finally, users may value interoperability features over latency questions.

A distinguishing feature of DSS is its *shared* architecture across all Move-based chains. This shared sequencer approach is pivotal in enabling seamless interoperability within the Movement Network ecosystem. By utilizing a common sequencing layer, DSS facilitates cross-chain atomic swaps and pooled liquidity across Move-based chains, significantly enhancing the network's overall security, functionality and efficiency.

The sequencers are responsible for posting transaction data to the DA service chosen by each rollup. To mitigate data withholding attacks, we implement slashing mechanisms for non-compliant sequencers. While DSS manages transaction ordering consensus, the Move

Executor, powered by MoveVM, handles transaction execution. This separation of concerns optimizes network efficiency and security, laying the foundations for future innovations such as privacy enhancements or opt-in censorship capabilities.

Unlike other shared sequencer solutions, our stewardship of both the DSS and the Move Stack Chain framework allows for deeper integration and optimization.

Our shared infrastructure approach not only reduces infrastructure burden for individual rollups but also creates a unified ecosystem where assets and liquidity can flow freely between Move-based chains, enhancing overall user experience and network utility. The result is a highly interoperable and scalable solution that combines the benefits of MoveVM.

5.3 Multi-asset staking

The DSS uses a Proof of Stake (PoS) mechanism. FFS also uses PoS to incentivize validators to be honest when attesting for new blocks. PoS, proven effective in ecosystems like Ethereum, requires candidates to stake native tokens, demonstrating commitment and adding capability to resist attacks. *Single-asset* staking requires stakers to stake in a fixed

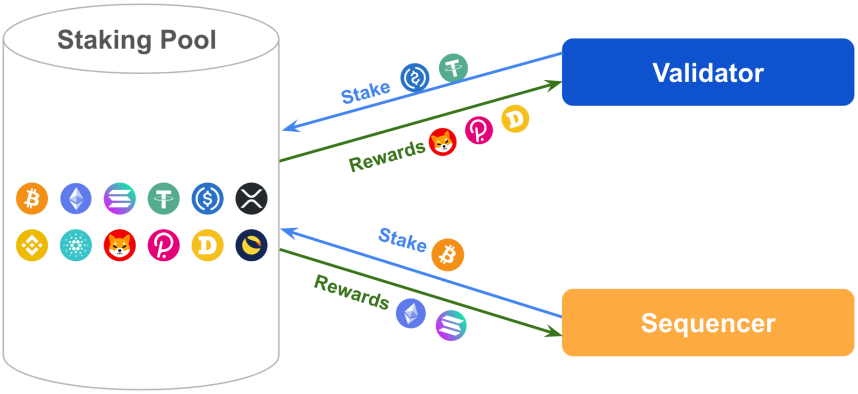

Fig. 10. Multi-asset staking.

crypto currency which means that they may have to swap assets before staking if they don't hold the token used in the staking protocol. This can be hurdle for stakers. This is why we will enable *multi-asset* staking, which is PoS that allows stakers to stake and get rewards in multiple assets (Figure 10, Image by myriammir on Freepik).

This is why we will enable *multi-asset* staking, which is PoS that allows stakers to stake and get rewards in multiple assets.

Multi-asset staking is convenient for stakers but comes with some challenges for the operator of the network:

- the staking pool is composed of several assets the prices of which may fluctuate,
- a PoS protocol usually relies on a *super-majority* of 2/3 of the total stake to finalize a decision (an ordering in the sequencer, a confirmation of a new block through FFS).

- as a result of the previous two points, some stakers may obtain unreasonable power and a small fraction of them may control the 2/3 super-majority, which negatively impact crypto security.

One solution to mitigate the problem is to use a (staking) *pool token*. Stakers stake arbitrary assets and are awarded pool tokens. When new stakers stake (resp. unstake) some assets, pool tokens can be minted (resp. burnt) and some re-balancing strategies [6] and liquidity curves choices have to be applied to manage the staking pool.

The implementation of secure strategies that protect our stakers (e.g. from impermanent loss) is an active research topic.

A critical feature in our multi-staking approach is the ability to stake without operating a node. This mechanism, called *Delegation*, maximizes the amount of staked value and, therefore, boosts the economic security substantially.

References

1. 0L: 0l network. https://0l.network/, accessed: 2024-01-12
2. Aptos: https://aptosfoundation.org/, accessed: 2024-01-12
3. Blackshear, S., Cheng, E., Dill, D.L., Gao, V., Maurer, B., Nowacki, T., Pott, A., Qadeer, S., Rain, D.R., Sezer, S., et al.: Move: A language with programmable resources. Libra Assoc p. 1 (2019)
4. Blackshear, S., Chursin, A., Danezis, G., Kichidis, A., Kokoris-Kogias, L., Li, X., Logan, M., Menon, A., Nowacki, T., Sonnino, A., Williams, B., Zhang, L.: Sui lutris: A blockchain combining broadcast and consensus (2023)
5. Buterin, V., et al.: A next-generation smart contract and decentralized application platform. white paper **3**(37), 2–1 (2014)
6. Forgy, E., Lau, L.: A family of multi-asset automated market makers (2022), https://arxiv.org/abs/2111.08115
7. Ivanov, N., Li, C., Sun, Z., Cao, Z., Luo, X., Yan, Q.: Security threat mitigation for smart contracts: A survey. arXiv preprint arXiv:2302.07347 (2023)
8. Liu, C., Liu, H., Cao, Z., Chen, Z., Chen, B., Roscoe, B.: Reguard: finding reentrancy bugs in smart contracts. In: Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings. pp. 65–68 (2018)
9. Motepalli, S., Freitas, L., Livshits, B.: Sok: Decentralized sequencers for rollups. arXiv preprint arXiv:2310.03616 (2023)
10. Motepalli, S., Jacobsen, H.A.: Decentralizing permissioned blockchain with delay towers. arXiv preprint arXiv:2203.09714 (2022)
11. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. Cryptography Mailing List (2008)
12. Ogundeji, O.: Vitalik buterin sets milestones on ethereum's route to be the 'world computer'. https://cointelegraph.com/news/vitalik-buterin-sets-milestones-on-ethereums-route-to-be-the-world-computer (2016), accessed: 2024-01-11
13. Sui: https://sui.io/, accessed: 2024-01-12
14. Ye, Z., Misra, U., Song, D.: Specular: Towards Trust-minimized Blockchain Execution Scalability with EVM-native Fraud Proofs. CoRR **abs/2212.05219** (2022), https://doi.org/10.48550/arXiv.2212.05219