Mango

Mango Network

Transactional Omni-Chain Infrastructure Network

v1.2

Abstract

In response to the multiple pain points of Web3 application and DeFi protocol, such as fragmented user experience and fragmented liquidity, Mango Network proposes a transaction-based omni-chain infrastructure network solution, aiming to build a one-stop liquidity service network through main-chain contracts and modular contracts, and to bring users a more secure and trustworthy, asset-diversified, and convenient and autonomous transaction experience. At the same time, Mango, as Layer1 and supported by Move, provides developers and users with a secure, modular and high-performance Web3 infrastructure, and the Devnet boasts a high TPS of 297.45k, which is both standardized, scalable and interoperable, and provides a solid underlay for building non-custodial, decentralized, omni-chain mobility pledges, lending and borrowing, POS pledges, GameFi, and other core applications, so that participants can experience an unprecedented level of freedom in Web3.

I Web3.0 Public Chain Leader - Move

1.1 Move emergence background

In 2019, Meta (formerly Facebook) introduced the global circulation of the sovereign digital currency project Libra. Throughout this process, Libra and the Diem team left behind a valuable legacy - the Move language that addresses the shortcomings of Solidity and EVM, as well as the derived public chain team. Move language is a resource-oriented programming language that borrows syntax and semantics from languages such as Rust, Cyclone, and Ada. The Move language aims to provide a high-performance, secure, and reliable programming environment for blockchain applications. It is a programming language designed specifically for digital assets.

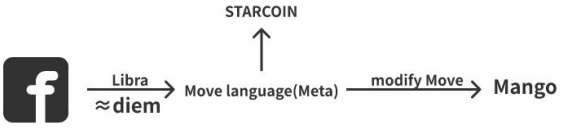
1.2 Advantages of Move series public chain

Move has redefined digital assets, believing that token assets are a unique and important form of data that should not be defined and represented using ordinary numerical types. Therefore, Move has created Resource to define on-chain assets separately.

Move language lowers the security threshold for developers, allowing contract developers to focus on business logic and write highly secure code, avoiding security vulnerabilities caused by low-level bugs. This is what makes DeFi (Decentralized Finance) safer and gives peace of mind to node users.

1.3 Characteristics of Move language

Move aims to provide a secure, efficient, and modular development solution for distributed applications. The core idea of the Move language is to break down complex programming tasks into multiple modules, effectively reducing complexity and lowering the probability of errors in programming.



1.3.1 Programming assets as first-class resources (First-class Resources)

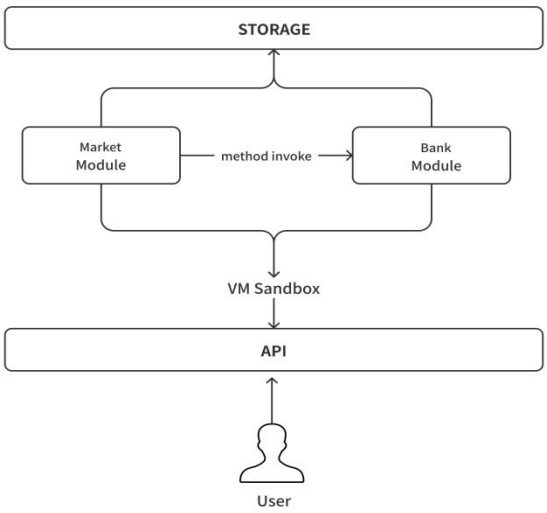
The Move language abstracts four attributes of resources: copyable, indexable, droppable, and storable. With different combinations of these attributes, users can easily define any type of resource.

The specified resources will be stored in a module controlled by the owner's account, represented by a verified owner/signer. The owner of these resources has the ultimate decision-making power and only the owner can determine the storage and transfer of the resources.

The design of resources allows for the transfer of digital assets to be a movement between storage locations rather than a simple addition or subtraction of balance values between accounts, thus avoiding re-entry and double-spending attacks.

1.3.2 Ensure its security through static call, virtual machine sandbox, etc

The Move language uses static calls, which improves runtime security, solves the problems of dynamic calls, and enhances network stability.



Move VM is the virtual machine sandbox for the Move programming language, where contract invocations are placed within the same sandbox. During this process, the security of the contract's state is primarily isolated through the internal security of the programming language, rather than relying on the virtual machine for isolation.

1.3.3 Verifiability

The Move programming language adopts formal verification. Formal verification is a means of using digital tools to analyze and prove the security of a program. The previously mentioned static calls and virtual machine sandbox can enhance the security of verification.

1.3.4 Flexibility

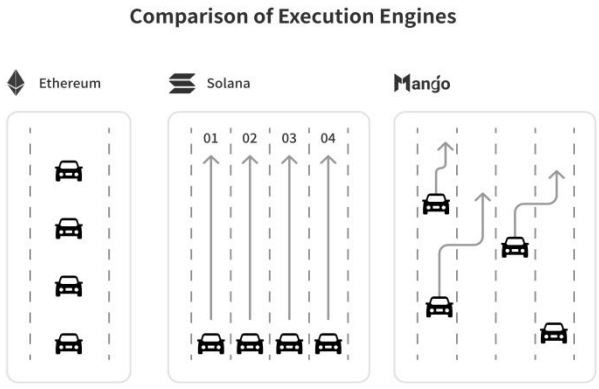
The flexibility of Move is reflected in the ability to freely combine various transactions through transaction scripts to achieve different functionalities, where one script can invoke multiple transactions. By using generic programming, Move ensures the scalability of contracts and increases code reusability.

1.3.5 Contract composability

The composability of the Move language comes from the design of its Modules. Upgrading and optimizing Modules enables all other contracts that have used this Module to automatically use the latest version, accelerating the speed of upgrading and optimizing smart contracts using the Move language.

1.4 Status quo of Move series public chains

The Move language truly achieves the digitization of assets and is hailed as one of the most suitable languages for writing blockchain smart contracts. It has been compared to the current mainstream language Solidity multiple times and has surpassed it in many aspects. The main reason is that Move provides a more fitting handling of the core token assets in blockchain, compensating for Solidity's shortcomings in intuitiveness and security.



Currently, the public chain based on the Move language is in the stage of ecological expansion, attracting more developers and users to join. Once an ecosystem is formed, users can bring sustainable income. The more prosperous the ecosystem, the stronger the innovation ability, and the more marginal benefits there will be. This is also the reason why public chains can become the core carrier of Web3.

II Technical Differences of Solidity VS Move

Transitioning from Bitcoin's Script to Ethereum's Solidity, undergoing the baptism of DeFi, Solidity has undoubtedly been successful. However, the security vulnerabilities of DeFi are also worth paying attention to. How to ensure the security of on-chain assets has become a major issue that the entire industry urgently needs to face.

Behind these vulnerabilities, VM is also constantly patching up, such as fixing the original overflow issues, but it is difficult to make substantial improvements in the underlying design flaws. Unlike previous development experiences, DeFi applications are essentially open-source and transparent financial systems that have extremely high security requirements. So, for financial scenarios, designing a more secure, reliable, and user-friendly smart contract language may be a better choice.

Characteristic	Solidity	Move
Turing completeness	Yes	Yes
Safety	Poor	Very good
Developer friendly	Yes	Yes
Dispersed storage	No	Yes
Engineering ability	Poor	Good
Flexibility	Dynamic mobilization	Generic programming
Financial scene enhancement	No	Yes

Solidity still has three major technical

challenges that need to be overcome urgently.

Solidity's syntax and expression methods are complex.

Traditional object-oriented programming languages like Java have become the best choice for novice developers to create Ethereum contracts. However, due to the complexity of Solidity, developers are unable to effectively understand its syntax.

There are code security vulnerabilities in Solidity.

This is also a major weakness of its competitiveness. There are many security vulnerabilities in Solidity due to improper usage, so developers need to be well-prepared for security testing before writing contracts in Solidity.

Solidity still has challenges in terms of performance.

The performance of Solidity compared to relatively simple programming languages like Python still needs improvement. With the growth of Ethereum, there is also a continuous increase in Ethereum-based applications. The performance issues of Solidity will affect the overall operational efficiency of Ethereum, thereby increasing network load and reducing the overall security of the Ethereum network.

From the above perspective, Solidity, as the most popular smart contract development language on Ethereum, is powerful enough but not perfect. Move language emerged in this context.

III Mango Programming Model

3.1 Resource-oriented programming

Mango Move has been specifically enhanced for financial, social, and gaming scenarios, introducing resource-oriented programming. For common scenarios such as FT and NFT, Mango Move defines data as resources, ensuring data security at the virtual machine level.

3.2 Pure static language

Dynamic dispatch is the cornerstone of Solidity, and all cross-contract invocations are implemented through dynamic dispatch. It is also the entry point for most security vulnerabilities, such as TheDAO attack, PolyNetwork cross-chain attack, and so on. Given the real experience of Solidity, Mango Move adopts a completely pure static implementation to better ensure the security of on-chain assets.

3.3 Formal verification

Formal verification, also known as FV (formal verification), refers to the use of mathematical tools to analyze the space of possible behaviors of a design, rather than computing specific values as results. Mango Move comes with a formal verification tool that can be used to test and prove the reliability of contracts using mathematical methods.

3.4 Distributed Storage

In the era of Web3.0, users have ownership of data. Solidity stores contract data in a centralized manner through Maps. When a contract has a vulnerability and the contract Owner's permissions are obtained, all user data will be compromised. Mango Move cleverly uses Resource to store data scattered under each user's Account, ensuring both data security and true ownership of data by all network nodes.

3.5 Generic programming

For security reasons, Mango Move has been designed as a purely static language, but this has not reduced its flexibility. Through generic programming, Mango Move ensures the scalability of contracts and increases code reusability.

Blockchain is a very important system, with a large amount of assets stored on the blockchain. Moreover, once these transactions on the blockchain are executed, they cannot be revoked or tampered with. It is necessary to ensure the normal operation of functions such as transaction storage on the chain. Mango Move is a smart contract language that can be compiled and run in a blockchain environment that has implemented Move VM. Mango Move and the MoveProver tool have emerged as a result.

IV Mango Move core design concept

Ensuring the security of smart contracts from the ground up, according to SlowMist's previous report, blockchain security incidents in 2021 resulted in losses exceeding 9.8 billion US dollars. As an emerging programming language, the Move language has also made breakthroughs and innovations in terms of security at different levels.

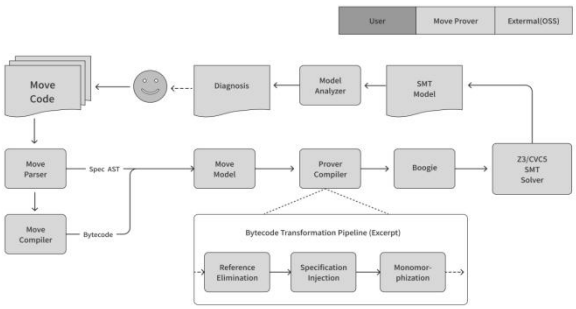
Just as what First-classResources really means as "digital assets are first-class citizens", Move is a smart contract language designed for manipulating digital assets. Compared to other languages, Move has a more native and low-level handling of token assets.

The Move language specifically defines assets as a type of resource, distinct from other data. In the context of blockchain, tokens are a type of resource, and the data of the resource must be stored under an account. During the transaction process, assets must flow to a destination, either transferred to another address or destroyed. Tokens cannot be duplicated or "double used".

Solidity and most programming languages treat tokens as numerical variables, where assets are simply numbers that can be added or subtracted. When the balance of one address decreases, the balance of another address increases. By using code, the decrease and increase in numbers can be made consistent, thus ensuring the security of assets through logical programming. Resource encapsulates the concept of assets at the language level, avoiding the arbitrary generation and access of assets, greatly improving security.

Byte verification goes before contract execution. Move differs from Solidity in terms of compiler. As an executable bytecode language, Move has built-in security algorithms and bytecode verifiers, which can prevent many common errors.

The Move contract code must be verified before it can be executed, which allows the contract to be protected from potential compiler errors and possible attacks. Move has been committed to establishing a standardized culture from the very beginning, with a standard library for each Move module. The Move development team has initially developed a dedicated validator called Move Prover for contract verification.



Move Prover Overall Frame

Based on modular composability: The modularity of smart contracts, which is more efficient and flexible, is another important feature for building a programming language ecosystem. The combination of smart contracts in the Solidity and other language ecosystems is primarily based on message passing between interfaces.

And in Mango Move, it is based on the combination of modules, interacting through the transfer of resources. Through the combination of modules, Mango Move saves block space occupied by contracts on one hand, and makes upgrades easier on the other hand. At the same time, due to the adoption of linear logic in the Module system, it can effectively package and encapsulate the concept of digital assets, clearly separating the definition of resources from the behaviors related to resources.

This resource-oriented programming brings about expressiveness and scalability that other languages cannot provide.

Mango Move Partial Program Code Display

Current status of the Move Prover

- 100% open source and actively maintained on GitHub.
- Fully automatic verification that implementations meetspecifications.
- Runs only a little slower than a linter (or other static analysistools) — important for developer experience.
  - 8,800 lines of code.
  - 6,500 lines of specifications.
  - Verification completes in a few minutes.
- Enforced in continuous integration, every change of Move codeneeds to pass the verification.

Look and feel : programming in Move

```
module 0x1 : : Account {
    // resources that can be stored in global storage
    struct Account has key {
        balance : u64
    }

    fun withdraw(account : address, amount : u64) acquires Account {
        // creates a mutable reference to an item in the global storage
        let ptr : &mut Account =
            borrow _ global_mut < Account > (account);
        ptr.balance = ptr.balance - amount;
    }

    // the "main" function for a transaction
    public entry fun transfer(
        from : address, into : address, amount : u64
    ) acquires Account {
        withdraw(from, amount);
        deposit(into, amount);
    }
}
```

V Type System of Mango Move

Due to a large amount of assets stored on the blockchain, the programming language of blockchain smart contracts must provide security guarantees, and Mango Move is a programming language designed for the blockchain that can provide better security.

The type system of Mango Move provides many protections, and we emphasize three points: first, type safety; second, resource safety; and finally, reference safety. The formal verification tool Move Prover mainly provides more advanced and expressive methods of representation. Whereas - structinvariants is a type, a structure should have some kind of state. -unit specification (per single function) Each program and each function must adhere to the corresponding specifications.

Type safety

Summary : Move is a strongly/statically typed language with strictly no type conversions

In more details :

- Every variable and expression in Move has one and only one type
- The type is known at compile time.
- The type can never be changed whatsoever.

NOTE: these are not type conversions as new variables are created

- freeze: &mut T --> & T
- as: u8/u64/u128 --> u8/u64/u128

5.1 Type safety

Mango Move is a strong type system and strictly guarantees that no type conversion can be done in any situation. Types cannot be converted between each other, and there is only one type for each. This is a guarantee provided by the Mango Move type system.

5.2 Resource safety

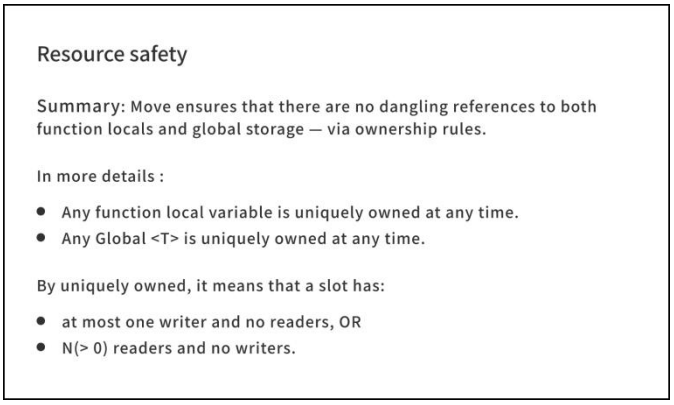
One is to ensure the scarcity of resources. The

abilities of copy and drop are depended to ensure that a resource is not duplicated, which is determined by these two abilities. The other two abilities (key and store) determine the possible use cases for this resource, whether it can appear in a function and disappear after the smart contract ends, and whether it can be written on the blockchain. These two abilities also determine this.

Mango Move also encapsulates resources, and all code related to resources should be encapsulated in one module, ensuring that developers cannot package it into multiple modules for resource handling. Therefore, Mango Move actually guarantees the security of resources in this aspect.

5.3 Quote safety

In the Mango Move language, it is ensured through "quote safety" and follows the ownership rules of Rust. Any local variable in a function can be modified by only one process at any given time. At the same time, for a global variable, within the entire blockchain, there can only be one owner who can make changes for any given type. This is ensured through Mango Move's reference safety.



VI Mango formalized verification system.

As a new generation of smart contract programming language, Move has made security its top design goal in order to support Libra's vision as a financial infrastructure and empower billions of people worldwide. The security features of the Move language can be divided into three levels: language level, virtual machine level, and tool level.

The formalization verification tool for Move is called Move Prover. The basic idea is to use an automated theorem prover in the formal verification domain to verify whether a program complies with a certain specification. This method requires users to have a detailed understanding of the program's operational logic, representing the program logic as specifications and conveying it to the verification system along with the program. "Move defines a specification language called Move specification language." Translate the following text into en-US without explanation: Convert Move programs and specifications into Boogie programs using the Move to Boogie compiler - an intermediate verification language with formal semantics.

6.1 Struct invariant

The invariants of a struct allow for a complex struct with multiple fields, where there exists a relationship between different fields within the struct. The invariants of the struct allow for specifying the relationship between different fields, which is currently not achievable by the type system.

Outline

Struct invariants allows you to specify complicated relations among the fields of a struct type which have to hold at runtime.

This set of specifications for struct plus struct actually serves as an "enhanced type" that can ensure the relationships between fields within the struct conform to a certain relationship.

```
module 0x1 : : Account {
    struct SumIsConst {
        a:u64,
        b:u64,
    }
    spec SumIsConst {
        invariant a >= b;
        invariant a+ b == 100;
    }

    fun create_valid(x : u64) : SumIsConst {
        assert! (x >= 50, 1);
        SumIsConst { a : x, b : 100 - x }
    }
}
```

```
[INFO] transdorming bytecode
[INFO] generating verification conditions
[INFO] 1 verification conditions
[INFO] running solver
[INFO] 0.011s build,0.005s trafo,0.052s gen, 0.625s verify, total 0.693s
error : data invariant does not hold
      test _ sum _ is _ const. move : 7 : 9
            invariant a >= b :

      at test _ sum _ is _ const. move : 11 : create _ valid
            x = 47
      at test _ sum _ is _ const. move : 12 : create _ valid
      at test _ sum _ is _ const. move : 13 : create _ valid
      at test _ sum _ is _ const. move : 7

exiting with verification errors
make : *** [ Makefile : 26 : demo _ sum _ is _ const] Error 1
```

6.2 Function specification

One focus of Mango Move is to do the specification of functions, specifying how a function should behave in certain situations, including how it should present itself and when it should throw an error. Once a warning condition is triggered, the execution will be terminated, which is the goal of function specification.

Function specification

For people not familiar with formal verification, function specification can be loosely considered as exhaustive unit testing.

The unit testing joke

A software engineer walks up to an ATM and ...

```
- withdraw(@0x1, 10);
- withdraw(@0x1, 1000);
- withdraw(@0x1, 99999999999999999999999999);
- withdraw(@0x1, 0);
- withdraw(@0x1, 20);
- withdraw(@0xdeadbeef, -123);
- withdraw(@0xbaadf00d, 666 * 2);
```

6.2.1 Abort Conditions

Move Prover provides an opportunity to express function transactions in a more specialized manner, allowing developers to write specifications for functions through the function's specification.

Abort conditions in the function specification

```
module 0x1 : : Account {
    struct Account has key { balance : u64 }

    fun withdraw(account : address, amount  : u64) acquires Account {
        let ptr : &mut Account = borrow _ global _ mut < Account > (account);
        ptr.balance = ptr.balance - amount;
    }
    spec withdraw {
        aborts _ if !exists < Account > (account);
        aborts _ if global < Account > (account).balance < amount;
    }
}
```

6.2.2 Post-conditions

Post-condition is the running state of a program function.

Post conditions in the function specification

```
module 0x1 : : Account {
    struct Account has key { balance : u64 }

    fun withdraw(account : address, amount: u64) acquires Account {
        let ptr : &mut Account = borrow _ global _ mut < Account > (account);
        ptr.balance = ptr.balance - amount;
    }
    spec withdraw {
        aborts _ if !exists < Account > (account);
        aborts _ if global < Account > (account).balance < amount;

        // on successful function return
        ensures global < Account > (account).balance ==
            old(global < Account > (account).balance) - amount;
    }
}
```

6.2.3 Pre-conditions

Function specification: Precondition, which means that in addition to being able to write exceptions in certain situations, there will be changes in the absence of exceptions. When calling this function, the precondition must exist. Having a precondition can simplify exception conditions.

6.2.4 Global Invariants

Invariants are a type of property that always remains the same during program execution. Immutability is one of them: once an object of an immutable type is created, it represents an unchanging value.

```
script {

  use 0x1::Debug;

  const RECEIVER : address = 0x999;

  fun main(account: &signer) {
    Debug::print<address>(&RECEIVER);

    // they can also be assigned to a variable

    let _ = RECEIVER;

    // but this code leads to compile error
    // RECEIVER = 0x800;
  }
}
```

VII Mango modular blockchain

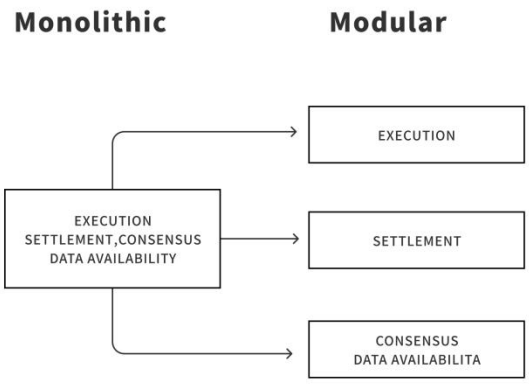
The Mango modular blockchain underlying code and contract layer code are programmed using the Move language, decomposing blockchain functionality into different levels of network architecture. Traditional blockchain systems typically integrate consensus, settlement, data availability, and execution functionalities into a single architecture. However, with the increasing complexity and demands of blockchain applications, a single architecture may not be able to meet the requirements of different scenarios. Therefore, Mango modular blockchain separates these core functions, allowing each functional module to operate independently while maintaining collaboration between them. This architecture makes the blockchain system more flexible and scalable, allowing for customization and optimization based on different needs.

7.1 Modular blockchain and monolithic blockchain

The concept of Mango modular blockchain corresponds to a single-piece blockchain, and the two have significant differences in terms of functionality processing. First, it is necessary to clarify the four core functions of blockchain:

1. Execution - transaction processing and computation.
2. Resolve - Dispute resolution and bridging.
3. Consensus - Transaction ordering.
4. Data availability - Ensure data is available.

Mango modular blockchain is any chain that is part of the modular blockchain stack, by modularizing the blockchain and splitting the four functionalities into multiple specialized layers. The modular blockchain stack consists of multiple layers of modular blockchains that depend on each other to create a system with all the above components.



7.2 Advantages of Mango Modular Blockchain

7.2.1 Sovereign protection

Although other layers are used, the Mango modular blockchain can have sovereignty like Layer1. This allows the blockchain to respond to hacker attacks and push upgrades without any underlying permission. Essentially, sovereign blockchain retains the ability to make critical decisions based on social consensus, which is one of the most important aspects of blockchain as a social coordination mechanism.

7.2.2 Efficient and low-cost start-up of new blockchains

Because the Mango modular blockchain does not need to handle all functions, new blockchains can simply use existing modular blockchains for components they wish to unload. This allows for the efficient bootstrapping of new blockchains, reducing deployment time and minimizing costs.

7.2.3 Scalability

Returning to the most fundamental issue, solving scalability is a major reason for promoting the modular development of blockchain, without sacrificing security and decentralization. Modular blockchain is not limited by the requirement to handle all functionalities. By dividing them into multiple layers, scalability can be achieved without sacrificing security and decentralization. This makes sustainable blockchain scalable and compatible with a decentralized multi-chain environment.

VIII Mango's new paradigm for omni-chain applications.

8.1 From single chain to multi-chain, and then to omni-chain.

As the first smart contract platform, most of all decentralized applications could only be deployed on Ether at the beginning of its birth. However, with the flourishing of public chain ecosystems and the development of Layer 2 solutions, applications now have more choices.

8.1.1 Single chain deployment status

The application can choose to deploy on a specific chain that suits its own business. Each chain or L2 has its own unique mechanism design, with different characteristics in terms of decentralization, privacy, and data availability, etc., to meet the needs of different applications. At the same time, various one-click chain creation stacks are becoming more and more mature, and application developers can completely create their own chains and customize related features. A typical example is the DYDX protocol, which is an early decentralized derivatives trading market created on Ethereum and has now migrated to a separate Cosmos Zone to better meet its throughput requirements.

8.1.2 Multi-chain deployment status

Applications can choose to deploy on multiple chains, allowing users on different chains to access the services provided by the application. From the point of view of the application itself, it is possible to scale up the business and increase the revenue of the protocol. For example, AAVE, a well-known lending protocol, has deployed its application on six chains.

8.2 Multi-chain application VS Mango omni-chain application

The deployment of contracts on multiple chains in an application cannot share liquidity, which also leads to the fragmentation of the Web3 ecosystem, including the fragmentation of user experience and liquidity.

8.2.1 Fragmented user experience in multi-chain cross-chain

In order to use different on-chain services, users need to transfer assets across multiple chains, register addresses on different chains, and learn operations on different chains. In many cases, users need to take many steps and prepare many kinds of Gas in order to achieve the desired results.

8.2.2 Liquidity fragmentation across multiple chains

In many types of DeFi protocols, liquidity depth is a part of their core experience. The same liquidity cannot exist simultaneously on multiple chains. Every DeFi protocol, when deployed on a new chain, needs to rebuild liquidity, which reduces the overall efficiency of liquidity.

8.2.3 Mango Omni-dApp (Mango omni-chain application)

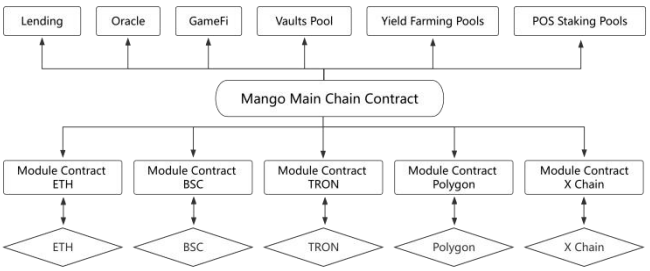
Mango's omni-chain application is based on the Mango Move underlying modular deployment contract protocol to carry the program's global state, which is a new way of building applications where developers design the application as a whole, taking omni-chain interoperability as a prerequisite, rather than simply replicating a single-chain application on multiple chains. The Mango omni-chain application consists of different modules on different chains, which have interoperability and together form a complete application. Compared to multi-chain deployment, Mango's omni-chain application can extend its tentacles to more chains, allowing users on more chains to use program services without the problem of fragmented user experience and fragmented liquidity.

Features	Multi-Chain	Omni-Chain
Status record	Distributed across multiple chains.	Unified on one chain
Liquidity	Distributed across multiple chains.	Unified on one chain
User experience	Users need to be familiar with multi-chain operations, prepare multi-chain gas, and frequently transfer assets.	Users can access programs on any chain just like accessing local programs.
Integratability	Only friendly to integrated chains, but more complex for cross-chain integration.	Better cross-chain interoperability

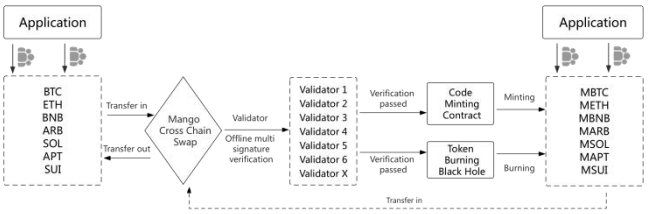
In this table, the analogy between omni-chain architecture and simple multi-chain deployment is shown:

8.3 Technical architecture of Mango Omni-chain Application

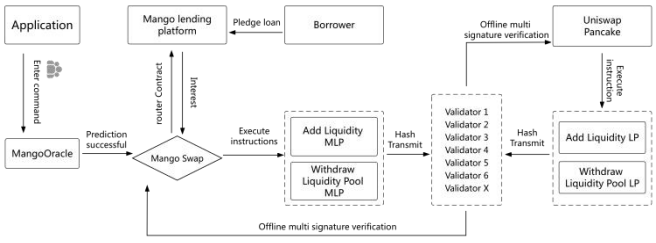
The technical architecture of the omni-chain application consists of the Mango main chain contract and module contract pattern. The main logic of the application is placed on the Mango main chain, like a "master control", and then other chains provide a remote access module to achieve interaction with end users, obtain user input, and output the desired results for users.



Mango Omni-chain Architecture Principle Logic Diagram



Mango's omni-chain cross-chain bridge technology architecture diagram



Mango's Omni-chain application technology architecture diagram

After the new chain obtains the user's input, it passes that input across the chain to the Mango master chain for processing and outputting the result, which is then passed across the chain to the new chain for outputting to the user. In some cases, different modules of the main chain may be split into different chains, which together form a virtual main chain.

8.4 Technical architecture advantages of Mango's Omni-chain Application

1. The entire chain is easy to expand. The main logic of the program is processed on the Mango main chain, and the application has a unified state record. After deploying the contract on the new chain, users can inherit all the state records and liquidity from the main chain without reinventing the wheel.

2. The overall user experience is improved, as users do not need to worry about which chain the program is deployed on. They can access the program from the main chain of the entire network, just like accessing a local program. There is no need to transfer assets back and forth, and there is no need to learn operations on multiple blockchains or prepare various types of gas.

3. The entire chain is easy to integrate across chains. When other applications integrate this program, they only need to connect on the main chain of the entire network to access all its features and liquidity, instead of having to connect separately to all other chains.

Mango Network's omni-chain application, as a new paradigm, has provided many new possibilities for us, supporting for arbitrary chain forging, redemption, and exchange. The application allows users to complete operations without perceiving any cross-chain processes.

When the omni-chain application becomes the mainstream paradigm of decentralized applications,

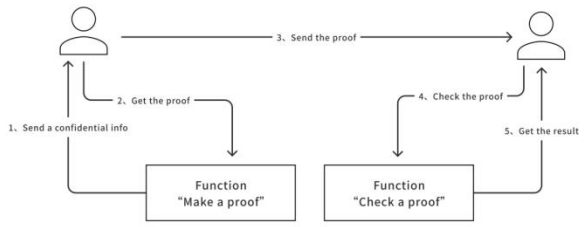
both application developers and users can experience unprecedented freedom in the blockchain world.

IX Mango public chain infrastructure protocol

9.1 ZK zero-knowledge proof

The Mango public chain combines zero-knowledge proofs, allowing one party to disclose information knowledge to another party without revealing the information itself. Extend the universal blockchain. Through proof of validity, blockchain nodes can verify transactions without storing detailed information or replaying computations. This reduces confirmation time and improves network throughput.

Due to its scalability advantages, zero-knowledge proof has become the core infrastructure of blockchain scaling projects, especially zero-knowledge rollups, in the Mango public chain. ZK-SNARKs and ZK-STARKs are the main types of zero-knowledge proofs.



9.1.1 Anonymous trading

Mango public chain uses ZK proof technology, allowing users to maintain anonymity during transactions. To verify the validity and legality of a transaction without exposing any information about the transaction participants or the assets involved, generate ZK proofs.

9.1.2 Privacy protection

Mango's public chain uses ZK proof technology to perform data validation without revealing sensitive information. When verifying that a certain node on the chain owns specific assets, ZK proofs can be used without revealing the identity of the participant or any other sensitive information.

9.1.3 Data integrity

The Mango public chain ensures that important data such as block data and smart contract states have not been tampered with and have been correctly calculated through the zk-SNARKs (Zero-Knowledge Scalable Non-Interactive Argument of Knowledge) protocol, providing highly reliable data proof for the Mango public chain.

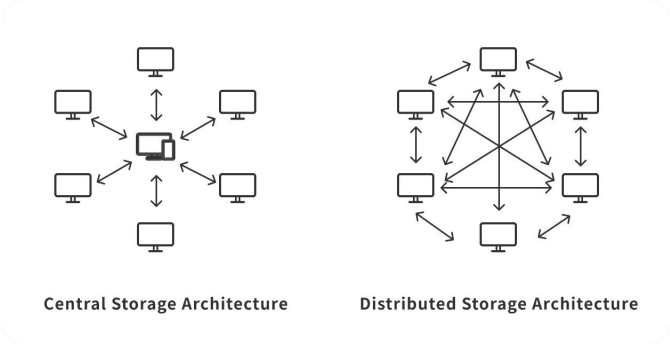
9.1.4 Cross-chain interaction

Mango public chain and ZK proof technology can be used for secure and private cross-chain interactions between different blockchains. To verify the validity of specific cross-chain transactions and protect the privacy of participants, it is done by generating ZK proofs.

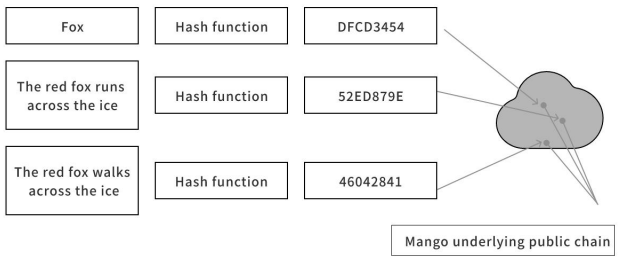
9.2 Distributed Storage

One way is to store data in a decentralized manner on multiple independent devices, which overall achieves distributed architecture. In the field of blockchain applications, it represents a new generation of distributed storage technology

represented by IPFS. Unlike traditional storage technology, the new generation of distributed storage not only changes the way storage is done, but also changes the system architecture and network transmission protocol, allowing distributed storage to truly be stored among different parties, while also achieving privacy protection and security for the data.



When the user extracts data, the same algorithm is used to calculate the hash of the data and obtain the corresponding data storage location from the hash table (as shown in the figure below).



In the Mango public chain infrastructure protocol, distributed storage is a technology used for storing and managing data. It disperses data storage across multiple nodes in the network to enhance data reliability, security, and scalability.

9.2.1 Data redundancy

The Mango public chain adopts distributed storage technology to achieve data redundancy. Each data block will be replicated on multiple different nodes to ensure that even if some nodes fail or go offline, the complete data can still be obtained from other nodes.

9.2.2 Data encryption

In order to protect user privacy and data security, the Mango public chain infrastructure protocol uses encryption algorithms to encrypt the data uploaded by users to the distributed storage network, which ensures that only authorized users can access and decrypt this data.

9.2.3 Data reliability

By replicating data to multiple nodes and using fault-tolerant mechanisms to detect and repair damaged or lost copies, Mango public chain ensures that its distributed storage system has high reliability. Even if certain nodes fail or are attacked, it is still possible to recover lost or damaged files from other replicas.

9.2.4 Scalability

The distributed storage system of Mango public chain can be horizontally scaled according to demand. When more data storage is needed, simply adding more storage nodes can increase the storage capacity of the entire system without compromising performance or reliability.

9.2.5 Data permission control

The Mango public chain infrastructure protocol provides a powerful data permission control mechanism. Users can define the permissions and policies for accessing their data, ensuring that only authorized users can access and modify this data.

9.3 MgoDNS domain name service

MgoDNS is a solution for distributed domain names based on cross-chain protocols. It provides domain and domain data analysis services for non-intermediated networks. This platform can help businesses and individual users manage valuable on-chain data information, and now they can participate in digital asset transactions in a more efficient, secure, and convenient way. MgoDNS aims to make the internet a transparent, secure, and free space through its innovative decentralized measures.

MgoDNS maximizes compatibility and achieves decentralization of the system. The underlying layer of MgoDNS is blockchain, and the upper layer is traditional internet. MgoDNS can also connect various types of public chains and consortium chains, collectively forming a super hub for connecting various blockchains.

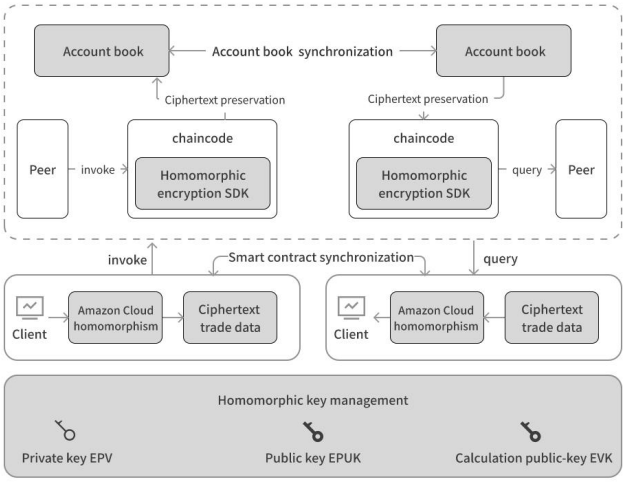
Traditional Internet has mature business models in project operation, but its drawbacks are also obvious, such as data leakage, server downtime, and other incidents. On the other hand, although blockchain technology fundamentally solves the problem of data trust, there is a lack of a certain foundation in terms of the scale and operation of applicable scenarios. So, the connection function of MgoDNS will enable the effective integration and application of both, complementing each other and achieving decentralization of the system.

9.3.1 MgoDNS domain name system, connecting the information internet and the value internet.

Unlike the infrastructure of traditional domain name services, MgoDNS's smart contract can replace the roles and processes of most registration agencies. Any institution or individual can create subdomains based on the rules of the registering institution. The parser plays the role of a translator, converting names into hash addresses and some mainstream public chain addresses. Based on cross-chain design, MgoDNS can serve existing public chain ecosystems such as Ethereum, IPFS, etc., making file access, address transfers, and smart contract calls more convenient and faster.

9.3.2 MgoDNS domain name parsing system, super hub connecting the information Internet and the value blockchain.

MgoDNS domain name system focuses on providing blockchain domain names and domain name data resolution services for blockchain networks, helping enterprises and individual users to participate in digital asset transactions and manage on-chain value data information more conveniently and securely. Meanwhile, based on cross-chain design, MgoDNS can serve existing public chain ecosystems such as Ethereum, IPFS, etc., making file access, address transfers, and smart contract calls more convenient and faster.



MgoDNS domain name resolution technology:

· MgoDNS achieves distributed domain name resolution by using domain name resolution smart contracts and cross-chain data exchange on multiple

mainstream chains.

· Domain name registration, resolution, and trading are not affected by any centralized factors, achieving complete decentralization. The security of domain names is guaranteed by blockchain technology, and the resolution of domain names is supported by a high-speed blockchain network.

·MgoDNS is fully compatible with the management and resolution of existing domain names.

·Support dynamic domain name resolution. MgoDNS automatically updates IP mapping and achieves blockchain confirmation in seconds through the wallet, enabling dynamic domain name services like traditional Peanut Shell and Dandelion without additional costs.

·Support for user-defined domain names. From the perspective of the domain name product, there are currently Ethereum Name Service (ENS) and Unstoppable Domains. However, ENS only supports the resolution of cryptocurrency addresses, making it difficult to connect to the traditional information internet and the underlying blockchain. On the other hand, the application scenarios of Unstoppable Domains are also relatively limited.

In addition to fulfilling the functions of traditional DNS domains, MgoDNS also combines the functions of information networks and value networks, supporting traditional domain name resolution, user identity authentication, DApp, social relationship networks, video conferences, and distributed information flow and other scenario-based applications.

	ENS	Unstoppable Domains	MgoDNS
Digital currency address resolution	Support	Support	Support
IP address parsing	Nonsupport	Support	Support
Traditional domain name analysis	Nonsupport	Nonsupport	Support
Underlying blockchain	Ethereum (ETH)	Ethereum (ETH)	ManGO
User identity authentication	Nonsupport	Nonsupport	Support
Domain name resolution client	Nonsupport	Nonsupport	Support
Decentralized network	Nonsupport	Support	Support
Decentralized encrypted mailbox	Nonsupport	Nonsupport	Support
Decentralized edge-cloud computing	Nonsupport	Nonsupport	Support
Decentralized edge DAppstore	Nonsupport	Nonsupport	Support

9.4 Mango client end

A copy (also known as: full client) is a consistent copy that maintains the effective state of the system, used for auditing and building transactions or operational services. Lightweight clients are based on object references and transactional authentication information, which can authenticate transactions that cause their creation or execution. Copies perform high-integrity reads of the system state without the need to maintain complete replica nodes. By providing concise

evidence packages and effect certificates, the lightweight client can undergo transformation within Mango and perform transaction operations or observe results. Regular checkpoint mechanism can be used to create collective checkpoints for final transactions, effectively verifying the recent state of objects and the events issued.

Customer-driven is a mechanism for handling authorization failures or client malfunctions. The client can solve these problems by updating to a level of honest authority that can handle correct transactions. Once there are no more certificates to synchronize, the list will be submitted to the new authority for execution. The relay is the client that performs this operation, and multiple relays can run simultaneously to update and coordinate the replica operation service with each other. Blocks are tools used by followers to receive updates and maintain the latest view state when processing authorizations. In addition, the authorities can use a push-pull gossip network to update transactions with each other and reduce the number of times the relay needs to perform this function. The authorities may use periodic status commitments to ensure that they have dealt with the complete set of certificates until a certain checkpoint.

The Mango system expands its capabilities by allocating more resources to handle transaction permissions. These resources include CPU, memory, network, and storage within or across multiple machines. Increasing resources can improve the ability to process transactions, thereby increasing revenue. At the same time, more resources will also reduce latency because operations can be executed without waiting for necessary resources to become available.

In order to ensure that more resources can increase capacity in a quasi-linear manner, the Mango system actively reduces bottlenecks and synchronizes points that require global locking within permissions. Transaction processing is explicitly divided into two phases: the first phase is to ensure exclusive access to a specific version of an object or shared access to an object; the second phase is to execute the transaction and commit its effects.

For transactions involving shared objects, it is necessary to use consensus protocols for ordering, which may become a bottleneck. "But the latest high-throughput consensus protocol for engineering indicates that sorting is only the bottleneck in state machine replication, not the execution order." In the Mango system, sorting is only used to determine the version of the input shared object, that is, incrementing the object version number and associating it with the transaction summary, without involving sequential execution.

The second phase occurs when the versions of all input objects are known to the permissions and involves executing a move transaction and committing its effects. Once the version of the input object is known, it can be executed in a fully parallel manner. On multi-core or multi-physical machines, virtual machines read versioned input objects and write the generated objects to storage. For object and transaction storage, the consistency requirements are very loose (except for sequential lock mapping), allowing each authority to use scalable distributed key-value storage. In addition, due to the idempotent nature of the execution, it is easy to recover even in the event of component crashes or hardware failures.

X References

1. https://github.com/MangoNet-Labs/mango
2. D. Matsuoka, C. Dixon, E. Lazzarin, and R. Hackett.(2022) Introducing the 2022 state of crypto report. https://a16z.com/tag/state-of-crypto-2022/
3. S. Blackshear, E. Cheng, D. L. Dill, V. Gao, B. Maurer, T. Nowacki, A. Pott, S. Qadeer, D. R. Rain, S. Sezer, T. Zakian, and R. Zhou, "Move: A language with programmable resources," 2019. https://developers.diem.com/papers/diem-move-a-language-with-programmableresources/2019-06-18.pdf
4. Sam Blackshear, David L. Dill, Shaz Qadeer,

Clark W. Barrett, John C. Mitchell, Oded Padon, and Yoni Zohar. Submitted on July 23, 2020; v1 submitted on April 10, 2020; originally announced in April 2020. Resources: A Safe Language Abstraction for Money. CoRR abs/2004.05106 (2020). arXiv:2004.05106 https://arxiv.org/abs/2004.05106
5. Marco Patrignani and Sam Blackshear. 2021. Robust Safety for Move. CoRR abs/218.05043(2021).arXiv:218.05043 https://arxiv.org/abs/218.05043
6. Daniel Collins, Rachid Guerraoui, Jovan Komatovic, Petr Kuznetsov, Matteo Monti, Matej Pavlovic, Yvonne-Anne Pignolet, Dragos-Adrian Seredinschi, Andrei Tonkikh, and Athanasios Xygkis. 2020. Online Payments by Merely Broadcasting Messages. In the 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2020, Valencia, Spain, June 29 - July 2, 2020. IEEE
7. David Dill, Wolfgang Grieskamp, Junkil Park, Shaz Qadeer, Meng Xu, Emma Zhong. Fast and Reliable Formal Verification of Smart Contracts with the Move Prover. Submitted 12 February, 2022; v1 submitted 15 October, 2021; originally announced October.2021. arXiv:218.08362 https://arxiv.org/abs/218.08362