Axiomesh: Modular Blockchain Built for Mass Commercial Adoption

AXIOMESH Foundation

April 18, 2024

Abstract

The mass commercial adoption of blockchain represents a new economic form based on crypto economics. It refers to the integration of the Web3.0 with traditional industries through blockchain technology, aiming to revolutionize production relations, optimize production factors, and reconstruct business models to achieve economic transformation and upgrading. Similar to the transformative impact brought by the internet and mobile internet on traditional businesses, the mass commercial adoption of blockchain will lead to significant economic leaps. Emerging applications such as real-world assets, crypto payments, and decentralized finance demonstrate tremendous market potential. However, compared to the market space of Web 2.0, there is still significant room for improvement, which requires further refinement of blockchain infrastructure.

Currently, the blockchain infrastructure suffers from issues like high barriers to entry, lack of privacy protection, and inadequate scalability. Therefore, we propose Axiomesh, a modular blockchain designed specifically for mass commercial adotption. Axiomesh innovatively addresses these challenges from both technical architecture and incentive mechanism perspectives to support mass commercial adoption.

In terms of technological innovation, Axiomesh proposes a modular blockchain architecture with unified liquidity based on the Trusted Inter-Module Communication Protocol (TIMC). This protocol facilitates dynamic expansion of execution and storage, enhancing system processing efficiency while maximizing the value of native blockchain tokens. Building upon this foundation, Axiomesh reduces the user barriers through Native Smart Accounts (NSA) and ensures privacy and security for user assets and identities through a Decentralized Confidential Protocol with Auditability (DCPA). These innovative technologies provide a solid foundation for increased participation of users and businesses in blockchain commercial adoption.

In terms of incentive mechanisms, Axiomesh innovatively introduces referral incentives while also retaining traditional mining incentives. These incentives ensure the stability of business activities and services for the blockchain underlying network.

Keywords: Mass Commercial Adoption, Smart Accounts, Privacy, Modularization, Referral Incentives

Contents

Chapter 1

INTRODUCTION

As cutting-edge technologies such as 5G communication and artificial intelligence continue to shape new paradigms of the internet, blockchain technology is similarly driving the development of the Web3 and new economic models. However, before this vision can be realized, the primary goal is to promote the mass adoption of blockchain in the commerce domain. The mass commercial adoption of blockchain symbolizes a new economic paradigm built on the principles of cryptoeconomics. Its aim is to reshape the essence of production relations, optimize resource allocation efficiency, innovate business process systems, and reconstruct business models through deep integration of blockchain technology with traditional industries, thus effectively driving the transformation and upgrade of the socio-economic structure.

Specifically, the mass commercial adoption of blockchain will impact the current Web3 ecosystem and traditional Web2 domain in two aspects. On one hand, mass means billions of users and trillions of capital, in other words, Web3 needs to absorb resources such as traditional Web2 enterprises, individual users, and diversified assets from the traditional Web2 world to continually expand the Web3 ecosystem. This will drive the widespread adoption of Web3, ultimately fostering the creation of a more open, equitable, and efficient new digital socio-economic structure. On the other hand, by integrating core Web3 technologies such as blockchain with Web2 industries, traditional internet and real-world economies can be empowered. This empowerment can lead to the improvement of traditional business models, optimization of business process efficiency, and catalyze industry upgrades and transformations.

What revolutionary impacts might the mass commercial adoption of blockchain bring? We will share several potential application scenarios of blockchain in commerce, so that everyone can more intuitively understand and imagine its profound impact and unlimited potential in commercial applications.

1.1. Application Scenarios

1.1.1. *Clearing and payment*

Currently, the Web2 financial clearing system exhibits evident centralization. A representative example is the reliance of most countries on the SWIFT system for international payment clearing. However, to date, several countries have been barred from using the SWIFT system, leading to significant disruptions in their international trade. Furthermore, the existing cross-border remittance mechanisms commonly suffer from prolonged processing cycles, typically requiring 3-5 business days or even longer to complete transfers.

In contrast, the cryptocurrency payment system, leveraging blockchain technology, effectively reduces trust costs and intermediaries' involvement, achieving borderless and real-time settlement for cross-border payments, thereby significantly enhancing payment efficiency and optimizing user experience. According to Brevan Howard's research[1], the on-chain settlement volume of stablecoins reached $11.1 trillion in

2022, surpassing PayPal's $1.4 trillion and comparable to Visa's $11.6 trillion. This highlights the immense potential of stablecoins in the payment application, particularly in providing efficient on-chain settlement. The application of stablecoins can assist projects in hedging revenue risks arising from the decline in US bond yields. Additionally, in developing countries with inadequate payment and banking systems, stablecoins can meet their demands for efficient and low-cost payment solutions. Therefore, stablecoins play an increasingly vital role in the global financial ecosystem, especially in promoting financial inclusivity and economic growth.

1.1.2. *Real-world assets*

Real-world Assets(RWA) tokenization refers to transforming tangible or intangible assets from the real world, such as gold, wines, artworks, stocks, and bonds, into digital tokens. These tokens represent the ownership and value of these assets on the blockchain. This process constructs a bridge between real-world assets and crypto finance, not only enhancing the liquidity of real assets but also utilizing fractional ownership investment models to lower the financial barriers for investors to access high-quality real-world assets.

Currently, RWA encompasses a vast scale of real-world asset markets. For example, global debt market has reached $307 trillion[2] in 2023, and the market value of gold is around $14.5 trillion[3]. In comparison, the total market value of native crypto assets is only $2.8 trillion[4]. If a portion of these RWA assets' shares can be introduced into the DeFi space, the total size of DeFi will experience a substantial increase. A report[5] released by global consulting firm BCG and private market digital trading platform ADDX predicts that by 2030, the tokenization of global illiquid assets will create a market worth $16 trillion.

1.1.3. *Decentralized physical infrastructure networks*

The core concept of Decentralized Physical Infrastructure Networks(DePIN) revolves around leveraging blockchain technology and token incentive mechanisms to encourage users and enterprises to share their physical infrastructure based resources, such as storage space, data, computing power, communication bandwidth, and wireless networks.

Taking the application of DePIN in AI as an example. Despite large Internet enterprises controlling a vast amount of high-end graphics card resources, integrating consumer-grade GPU and CPU resources in a decentralized manner can also build a substantial computing power network. According to an academic study conducted by UCLA[6], under equivalent cost conditions, decentralized computing architecture achieved up to 2.75 times the cost-effectiveness compared to traditional GPU cluster solutions. This translates to a 1.22 times speed improvement while reducing costs by 4.83 times. According to a recent report by Messari[7], the total market value of DePIN track and liquid tokens exceeds $20 billion, generating approximately $15 million in annualized on-chain revenue. Messari estimates that by 2028, the market size of the DePIN track is expected to grow to $35 trillion. This not only reflects the industry's interest in the DePIN model but also demonstrates people's expectations for the development of new network infrastructure.

1.1.4. *Trade finance*

Trade finance is a set of techniques or financial instruments used to mitigate the risks inherent in international trade, ensuring both payment to exporters and the delivery of goods and services to importers.

The traditional trade finance process typically involves as many as 20 different paper documents, leading to potential errors and fraud due to numerous handlers, human errors, and incompatible systems. With the UK legal system beginning to accept electronic transferable records, trade finance is undergoing a digital transformation. In this process, the introduction of blockchain technology will create and maintain a single version of the document accessible to all relevant parties for collaboration and editing, thereby eliminating redundant manual processes and the possibility of forgery. According to a report by UK Government[8], the digital transformation of trade finance is expected to reduce the printing and transportation of nearly 30 billion paper documents daily, consequently reducing the global cost of international trade. The Commonwealth estimated in a report[9] that, by 2026, the paperless trade across the Commonwealth countries could bring benefits about up to $1.2 trillion, and similar benefits are also expected in Asia.

Blockchain has shown its potential in numerous real-world applications due to its inherent transparency and immutability. However, to fully unleash its capabilities in broader and more complex large-scale commercial scenarios, blockchain needs to integrate and strengthen core capacities such as privacy protection mechanisms, high-performance processing capabilities, and user-friendliness.

In the following discussion, we will examine the various technical and strategic optimizations undertaken by mainstream blockchain systems to drive the process of mass commercial adoption, as well as the challenges they face in this process.

Chapter 2

RELATED WORKS

Since Bitcoin[10] and Ethereum[11] kickstarted the blockchain revolution, researchers have been experimenting to make blockchain work for large-scale commercial use. But so far, no perfect blockchain system has emerged that can smoothly handle the demands of mass commercial adoption. In this analysis, we'll look at how popular blockchain systems are trying to adapt for mass commercial use, what challenges they face, and how our system aims to learn from their experiences and innovate to create a blockchain platform that can support mass commercial adoption.

One of the goals of achieving mass commercial adoption of blockchain is to attract billions of users to migrate to the Web3 ecosystem. Just as in the traditional internet environment where personal identities are constructed through account registration, the first step for users entering the Web3 world is to obtain a blockchain address prefixed with 0x. Currently, the digital identity system on the blockchain has developed a series of innovative technologies such as decentralized identity (DID)[12], hardware wallets[13], and multi-party computation (MPC) wallets[14]. Although wallets like MetaMask[15] have accumulated millions of users, they still fall short compared to the hundreds of millions of users on the traditional internet. It is widely believed in the industry that improving the user experience in interacting with blockchain applications is a crucial path to achieving this goal. In terms of usability, the Ethereum name service (ENS) [16] can map Ethereum addresses with 0x prefix to custom and recognizable domain names, such as vitalik.eth. But this is just the first step in improving the experience. To further reduce the complexity of Web3 usage, researchers are actively exploring the application of account abstraction(AA)[17], which encapsulates the technical details of underlying Ethereum account interactions. So account abstraction can greatly simplify user operations, such as eliminating cumbersome steps like mnemonic phrases. However, the current Ethereum abstract accounts are built on smart contracts and still face issues such as expensive gas fees and high developer entry barriers.

The public and transparent nature of blockchain technology provides security and verifiability for transactions. However, this transparency also comes with the risks of privacy breaches, especially in scenarios where blockchain integrates with Web2 industries, potentially leaking sensitive information such as user identities and assets. Researches on blockchain privacy protection have led to various technical strategies. For example, Dash[18] adopts a decentralized mixing mechanism to enhance privacy by obfuscating the correlation between multiple users' transaction inputs and outputs. Zcash[19] employs zero-knowledge proof to effectively mask the identity of transaction recipients. Monero[20] uses ring signature technology to achieve high privacy by hiding transaction amounts and the addresses of transaction parties. However, the occurrence of sanction incidents involving Tornado Cash[21] has sparked widespread discussions on privacy rights and regulatory compliance issues, thereby giving rise to technical approaches that balance privacy with regulatory compliance, such as the Privacy Pool[22] mechanism. However, the current industry often overlooks the composability between protocols when implementing such technologies.

Scalability is a key indicator for measuring whether a blockchain platform can support a large-scale business ecosystem. Currently, the industry has explored several effective technical approaches to scalability. One approach is to enhance the performance of a single chain, such as the Aptos[23] and Sui[24] based on the Move programming language, as well as Solana[25], which adopts a novel consensus algorithm. These projects to some extent sacrifice composability in exchange for higher single-chain processing capabilities. The second approach is to use a multi-chain and side-chain architecture to compensate for the shortcomings of a single chain's performance. For example, concepts like parallel chains proposed by Polkadot[26] and the Zone mechanism of Cosmos[27]. However, while this approach can enhance the overall capacity of blockchain infrastructure, the high operating and maintenance costs of parallel chains restrict the robust development of ecosystems. The third approach is the Rollup[28], represented by Ethereum Layer2. Rollup reduces decentralization to some extent by inheriting the security of Layer1. However, due to the loosely coupled relationship between current Layer2 protocols and Layer1, as well as the complete dependence of Layer2 transaction verification and settlement on Ethereum smart contracts, there are still issues with transaction latency.

Despite extensive and in-depth research conducted by the industry at various technical levels, there is still no comprehensive technical solution that fully supports mass commercial adoption of blockchain. This is due to the fragmentation and independent development of various solutions. To meet the demands of mass commercial adoption, blockchain still needs to achieve breakthroughs in core aspects such as usability, privacy protection, and scalability. Therefore, Axiomesh will build upon the wisdom of predecessors and strive to construct a blockchain system that integrates composability, privacy, scalability, usability, and sustainability, aiming to support and drive mass commercial adoption of blockchain.

Chapter 3

VISION

Axiomesh's vision is to empower the real-world economy through cryptoeconomics, fostering a business revolution in the third generation of the Internet while supporting mass adoption of blockchain technology for commercial applications. Just like the industrial revolution brought about by the Internet and mobile Internet, breakthroughs and improvements in infrastructure are essential prerequisites for achieving industry transformation. Currently, the core technology of the third generation of the Internet, blockchain, still faces crucial technological barriers that need to be overcome. Our mission is to provide a modular blockchain architecture with unified liquidity to solve the last challenge of the third-generation Internet business revolution caused by high user entry barriers, lack of privacy protection, and poor scalability due to current blockchain technology. When designing our novel blockchain architecture, we have incorporated the advantages and research findings from existing outstanding underlying blockchain infrastructures. These include efficient and secure consensus algorithms, widely adopted smart contract execution engines like EVM, and protocol-rich frameworks like LibP2P. Building upon this foundation, we aim to offer low-entry capabilities, robust privacy protection mechanisms, and scalable technical architecture for mass commercial adoptions. Additionally, we seek to design a more sustainable tokenomic model for Axiomesh.

Chapter 4

DESIGN PRINCIPLE

- **Composability:** Currently, Ethereum serves as the most widely adopted platform for smart contracts globally. Its built-in Ethereum Virtual Machine (EVM) and the extensive ecosystem of surrounding development tools have achieved a high level of maturity and refinement. To maximize the utilization of this advantage, Axiomesh should prioritize the implementation of EVM compatibility, enabling seamless migration of the majority of EVM ecosystem dApps to Axiomesh at zero cost.
- **Privacy:** The full transparency of blockchain transactions leads to the disclosure of user transaction behaviors and asset status. Merely relying on address-level anonymity is insufficient for fully protecting user privacy, especially when it comes to commercial secrets and enterprise information. With the maturity of zero-knowledge proof technology, Axiomesh can provide an asset protection protocol that balances verifiability and privacy protection, helping businesses and individuals better build Web3 applications.
- **Scalability:** To meet the performance demands of mass commercial adoption, Axiomesh's core architectural design should have extremely high scalability. We will support native component extension capabilities. This differs from the current architecture of Layer2 or sidechains, as Axiomesh will maintain unified token liquidity while providing scalability.
- **Ease of use:** The high entry barriers to blockchain usage have always been one of the factors hindering the mass commercial adoption of blockchain. Axiomesh is dedicated to adopting a native support for smart accounts to provide developers and end users with a user experience on par with Web2.0.
- **Sustainability:** The realization of mass commercial adoption for blockchain still has a long way to go, which is why Axiomesh's tokenomic model should consider providing incentives for ecosystem and community development during this transitional period. Additionally, given the widespread adoption of mass commercial adoption, the network usage fees for Axiomesh should be sufficiently affordable.

Chapter 5

ORGANIZATION

The organization of this article is as follows: Chapter 6 will explain Axiomesh's basic logical data model, including the account model, transaction structure, and block structure. Chapter 7 provides an overview of the protocol, describing Axiomesh's topology, node roles, transaction process, and also introducing its core components, consensus algorithm and smart contracts. Chapters 8, 9, and 10 are modules innovatively designed by Axiomesh for mass commercial adoption, including the **Native Smart Account (NSA)**, the **Decentralized Confidential Protocol with Auditability (DCPA)**, and the **Trusted Inter-Module Communication protocol (TIMC)**. Chapter 11 expounds on Axiomesh's tokenomic, including token functions, mining rewards, and referral reward models.

Chapter 6

DATA STRUCTURE

This chapter primarily introduces the core data model utilized in Axiomesh. The core data structures of Axiomesh are designed around its core capabilities such as NSA, DCPA, TIMC, etc. Based on this, compatibility with Ethereum's EVM is achieved to reuse existing EVM ecosystem tools, with reserved fields for subsequent upgrade optimizations.

6.1. Account

Accounts in Axiomesh refer to the basic units maintaining the on-chain state, which can be interacted with by users/other accounts. Axiomesh internally supports three types of accounts:

- **Externally Owned Accounts (EOA)**: Controlled by users through private keys, incapable of supporting complex logic on their own.
- **Contract Accounts (CA)**: Controlled by the code of smart contracts, supporting complex logic.
- **Smart Accounts (SA)**: Controlled by users through various means (such as using traditional Web2 accounts or custom methods), while also supporting complex logic through smart account code, and programmable logic.

These three types of accounts are represented and stored using a unified account model, which includes the following fields:

Name	Description
addr	Account Address.
type	Account Type (EOA, CA, SA).
version	For SA accounts, used to indicate the corresponding smart account implementation version (based on system contract implementation rather than EVM contract).
code_hash	For CA accounts, records the hash of the bytecode (bytecode stored separately).
nonce	For EOA and SA accounts, records the cumulative transaction count to prevent double spending attacks.
balance	Balance of native coin(AXC).
state_root	Root hash of the internal state tree for CA and SA (internal state stored separately).

Table 6.1. *The account structure*

Accounts are stored in the chain according to the logic shown in Figure 6.1:

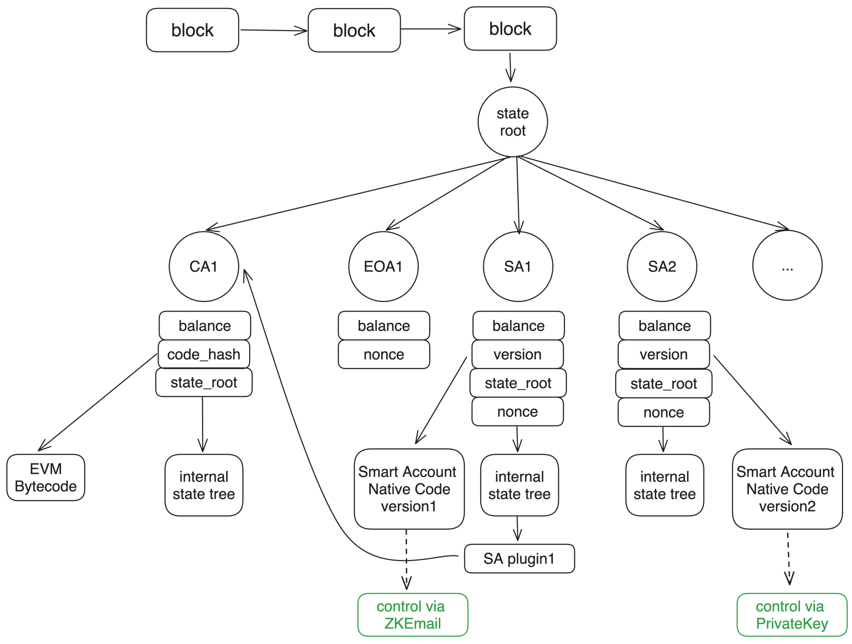

Figure 6.1. *Account model*

6.2. Transaction

In Axiomesh, transactions refer to the primary means by which users interact with the Axiomesh system. Changes in the Axiomesh ledger state are also driven by transactions. Axiomesh primarily supports the following five types of transactions:

1. **Normal Transaction**: This is the most basic transaction format, primarily used by regular users to send transactions using EOA through EVM ecosystem tools. Its transaction format is compatible with Ethereum's EIP1559 transaction [29] format, with the priority_fee field being ignored. For specific transaction fields, refer to Ethereum's EIP1559 transaction.
2. **EOA Referral Transaction**: Axiomesh supports referral rewards, rewarding the address of the source node in transactions. This transaction extends the basic transaction and is mainly sent by regular users using EOA through Axiomesh peripheral tools.

   In addition to the specific fields of a regular transaction, the additional fields included in the transaction are as follows:

Name	Description
referral_address	The address to receive referral rewards (typically the EOA account address of the receiving node of this transaction).

Table 6.2. *The account structure*

3. **Smart Account Transaction:** Axiomesh natively supports smart accounts, and the use of smart accounts requires invocation through smart account transactions, eliminating the need for the bundler component for secondary packaging and sending in AA.

The specific fields of the transaction are as follows:

Name	Description
nonce	Transaction nonce.
gas_price	Maximum acceptable gas price.
gas_limit	Maximum gas limit.
to	Address of the SA account to be called.
value	Amount of tokens to transfer.
input	Data for SA account invocation.
chainid	Chain ID.
referral_address	The address to receive referral rewards (typically the EOA account address of the receiving node of this transaction).
proof_data	Proof data (to validate the transaction's validity by invoking SA using this data).

Table 6.3. *The smart account transaction structure*

4. **System Transaction:** Refers to some operations that settlement layer consensus, incentive, and other modules need to automatically perform after the end of certain blocks (such as incentive settlement, validator election, etc.). These operations are uniformly represented by system transactions and attached to the beginning/end of the block's transaction list. The sending of these transactions is appended by the block-producing node according to the logic of system contracts when packaging transactions. Their validity is ensured by the system contracts and consensus algorithms, and execution does not require gas consumption.

   The specific fields of the transaction are as follows:

Name	Description
nonce	Transaction nonce (shared counter for all system transactions).
chainid	Chain ID.
to	Address of the system contract to be called.
input	Data for calling the system contract.
source	Address of the system contract generating this transaction.
source_check_data	Call data used by the system contract generating this transaction to validate its validity.

Table 6.4. *The system transaction structure*

5. **TIMC Transaction:** This transaction is used to submit states from the execution layer to the settlement layer, while also supporting the transmission of messages to dApps on the settlement layer or other execution layers. This transaction is separately verified and executed by the TIMC module (only TIMC transactions issued by registered execution layers are valid). The gas fees generated by this transaction are deducted from the funds locked in the execution layer.

   The specific fields of the transaction are as follows:

Name	Description
nonce	Transaction nonce.
chainid	Chain ID of the settlement layer.
batch_number	Sequence number of the batch submitted by the execution layer.
batch_start_block	Start block number of the batch submitted by the execution layer.
batch_end_block	End block number of the batch submitted by the execution layer.
batch_hash	Hash of the batch submitted by the execution layer.
batch_exec_proof_data	Execution proof data of the batch submitted by the execution layer (validated by verification contracts registered to TIMC on the settlement layer).
batch_da_proof_data	Storage proof data of the batch already uploaded to the DC layer submitted by the execution layer (verified by TIMC on the settlement layer).
msgs	List of messages sent to the settlement layer/other execution layers.
withdraws	List of withdrawals of native tokens from the execution layer to the settlement layer.
transfers	List of transfers of native tokens from one execution layer to another.
signature	Signature of the transaction (signed by the account private key registered to TIMC on the execution layer).

Table 6.5. *The TIMC transaction structure*

6.3. Block

In Axiomesh, multiple transactions are packaged and undergo consensus to form a valid block. A block consists of two parts: the block header and the block body. This section mainly describes the block structure in the Axiomesh settlement layer (the block structure in the execution layer supports custom extensions for scalability and is therefore not described here). To maintain compatibility with the settlement layer and the EVM ecosystem, the block structure is compatible with Ethereum blocks.

The block header contains the following fields:

Name	Description
number	Block number.
state_root	Root hash of the current block's state.
tx_root	Root hash of the transaction tree.
receipt_root	Root hash of the receipt tree.
parent_hash	Hash of the parent block.
timestamp	Timestamp of block creation.
epoch	Epoch to which the block belongs (an epoch is the basic unit of operations such as settlement, validator node election, etc.).
bloom	Bloom filter of log indices from all receipts (used for quickly determining if the log to be queried exists in the block).
gas_price	Gas price for the next block (dynamically changes based on the number of transactions in the recent block).
gas_used	Gas used in the current block.
proposer_account	Account address of the proposing node for the current block.
proposer_node_id	Node ID of the proposing node for the current block.

Table 6.6. *The block header structure*

Chapter 7

PROTOCOL OVERVIEW

This chapter aims to provide a preliminary description of the overall system of Axiomesh. Firstly, it will outline the three-layer modular architecture of Axiomesh logically. Secondly, it will explain the interaction between transactions and the three core modules of Axiomesh in conjunction with the transaction lifecycle. Lastly, it will delve into the introduction of Axiomesh's core layer consensus algorithm and smart contract virtual machine. Other core parts of the protocol will be detailed in separate chapters.

7.1. Overview

The modular blockchain ecosystem of Axiomesh, as shown in Figure 7.1, consists of the Settlement Layer, Data Custodian Layer, and Execution Layer. These layers are interconnected through the Trusted Inter-Module Communication Protocol (TIMC) to facilitate the transfer of value and witness data between different modules at various levels.

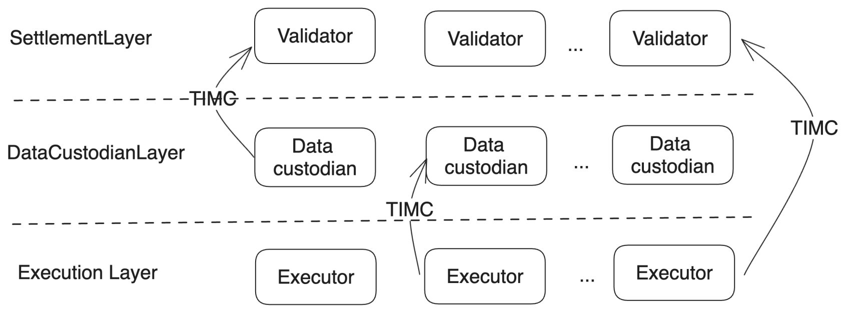

Figure 7.1. *Axiomesh modular schematic*

- **Settlement Layer:** This is the core ledger layer of Axiomesh, primarily responsible for the security of the entire modular ecosystem. It is uesd for asset issuance, transactions, and computation, as well as for proof storage and verification in the execution layer and data custodian layer. The settlement layer consists of validator nodes and non-consensus candidate nodes, which run the dBFT consensus algorithm and provide an EVM-compatible execution engine.

- **Execution Layer:** This is the extension layer for the settlement layer's execution module. With TIMC protocol, Axiomesh can extend the execution module to achieve efficient parallel processing of non-related businesses by running EVM-compatible execution modules. Additionally, the execution layer can also provide better privacy protection and business-related execution optimization capabilities.

- **Data Custodian Layer:** This is the extension layer of Axiomesh's internal storage module. With TIMC protocol, Axiomesh can extend the storage module. Data custodian nodes periodically provide proof of stored data to the settlement layer using erasure coding, data sampling, and zero-knowledge proof algorithms. The data custodian layer not only ensures data availability, but also leverages secure multi-party computation technology to provide data privacy protection for the business data of the settlement layer and execution layer.

7.2. Roles

While Axiomesh blockchain utilizes a modular architecture, it differs from the traditional Rollup + DC modular architecture in that Axiomesh blockchain avoids the liquidity fragmentation issue that occurs in traditional modular architectures by adopting the TIMC protocol and uniformly using AXC as the base token for economic operations.

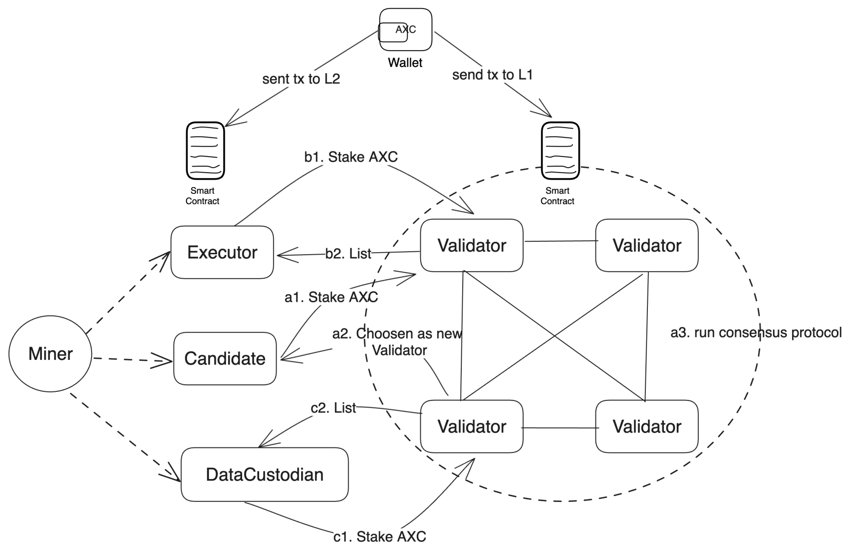

Figure 7.2. *Diagram of the main roles in Axiomesh*

As shown in Figure 7.2, blockchain miners can become one of the roles in the Axiomesh ecosystem by staking Axiomesh's native token AXC after entering the Aixomesh network. Different network roles receive corresponding benefits based on the network contributions they provide.

- **Validator nodes** earn the qualification to participate in maintaining consensus nodes and earn network incentives, referral incentives, and transaction fees in the Axiomesh ecosystem by staking Axiomesh's native token AXC.

- **Candidate nodes** refer to miners who will first go through the role of a candidate node before formally becoming a validator node. They become validator nodes and participate in network consensus after being elected at specific nodes in the consensus algorithm.
- **Executor nodes** earn the permission to provide execution services by staking Axiomesh's native token AXC to the Settlement Layer and earn transaction fees and network incentives from the Axiomesh ecosystem.
- **Data Custodian nodes** earn the operational rights by staking Axiomesh's native token AXC to the Settlement Layer and earn data storage service fees paid by users as well as network incentives from the Axiomesh ecosystem.

The native token AXC of the Aixomesh blockchain network serves as both the asset that miners need to stake to become different roles and the billing unit and incentive unit for different modules and levels within the network ecosystem. This design approach helps avoid the ecological liquidity fragmentation issue caused by the traditional architecture of Rollup + DC.

7.3. Transaction Flow

Axiomesh has adopted a modular architecture with unified liquidity. Users can directly send transactions to the settlement layer for processing or to the execution layer for processing, depending mainly on the user interaction account and the address where the dApp is deployed. Here we will introduce these separately.

7.3.1. *Settlement layer*

The settlement layer transaction flow refers to the entire process from when a user initiates a transaction on the settlement layer to transfer native tokens, interact with smart contracts deployed on the settlement layer, and transfer funds to the execution layer, until the transaction is confirmed by the settlement layer (confirmation means the transaction has passed through network consensus and cannot be tampered with). The overall process is shown in the following diagram.

1. Users send transactions to the RPC module of the validation nodes through their wallets.
2. The PreCheck module of the validation nodes reads account data from the ledger to verify the validity of transactions.
3. If the transaction is valid, it will be placed in the transaction pool of the validation nodes.
4. The transaction pool broadcasts transactions to other nodes in the settlement layer network via a P2P network. When the transactions in the pool meet the packaging conditions (reaching a certain quantity/fixed time), a batch of transactions will be packaged and sent to the consensus module.
5. The consensus module runs a consensus algorithm on the packaged transactions with other validator nodes to reach a consensus. Once the consensus is reached, a block is generated and passed to the execution module.
6. The execution module of the validation nodes executes transactions. Deposit transactions and cross-layer message transfer transactions in the block are processed by the TIMC module.

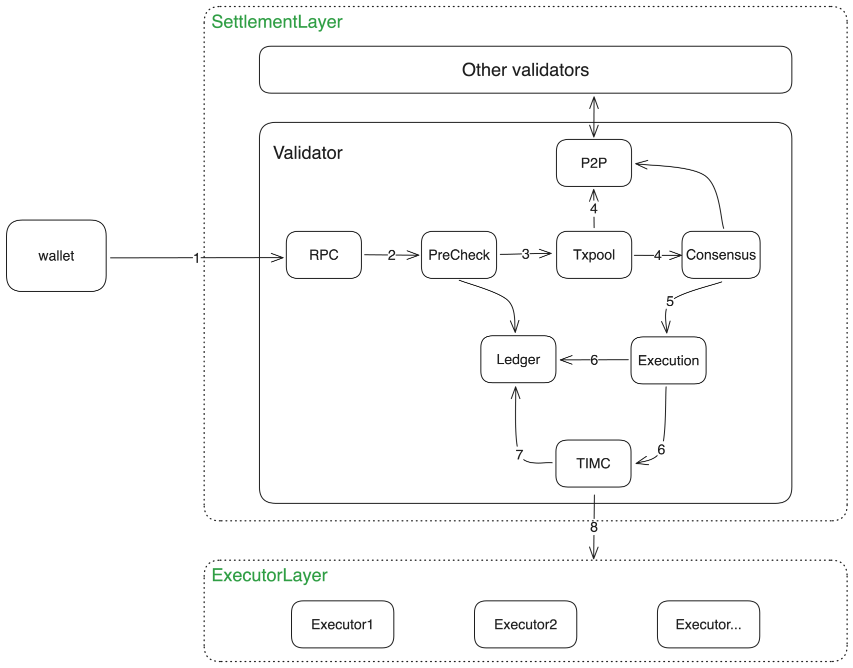

Figure 7.3. *Settlement layer transaction flow*

7. The TIMC module of the validation nodes verifies and processes transactions, generating TIMC messages attached to the block header.
8. After all transactions are processed, a series of block data is updated (such as the state tree root hash, receipt tree root hash, etc.), and the block data and transaction receipt data are written to the ledger for persistent storage, confirming the transactions in the block at this stage.
9. The executor in the execution layer listens to the TIMC messages in the settlement layer block header to find and process messages belonging to that executor.

7.3.2. *Execution layer*

The transaction flow in the execution layer refers to the entire process of user-initiated transactions on the execution layer, including native token transfers, interactions with smart contracts deployed on the execution layer, withdrawing funds to the settlement layer, transferring funds to other execution layers, and other operations. This process encompasses the entire lifecycle of a transaction, from initiation to confirmation by the execution layer and further confirmation by the settlement layer. The overall process is illustrated in Figure 7.4.

1. Users send transactions to the RPC module of a specific executor in the execution layer through their wallets.
2. The PreCheck module of the executor validates the transactions for their validity.
3. If the transaction is valid, it will be placed in the transaction pool of the executor.

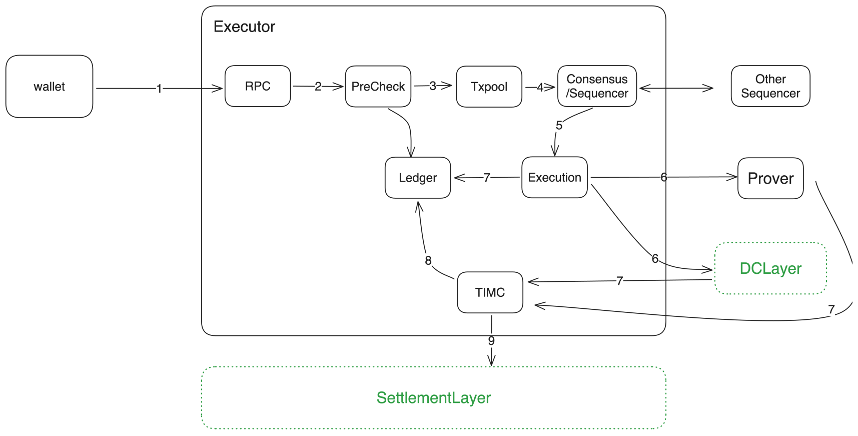

Figure 7.4. *Execution layer transaction flow*

4. When the transactions pool meet the packaging conditions (reaching a certain quantity/fixed time), a batch of transactions will be packaged and sent to the consensus/sequencing module for sequencing.
5. The consensus/sequencing module sequences the transactions and generates blocks, which are then passed to the execution module.
6. The execution module of the executor executes transactions, with TIMC transactions in the block being processed by the TIMC module.
7. The TIMC module of the executor verifies and processes transactions, generating TIMC messages attached to the block header.
8. After all transactions are processed, a series of block data is updated (such as the state tree root hash, receipt tree root hash, etc.), and the block data and transaction receipt data are written to the ledger for persistent storage, confirming the transactions in the block at this stage.
9. The block is sent to the Prover to generate execution proofs and sent to the DC layer to generate storage proofs.
10. Once the TIMC module of the executor collects a batch of execution proofs and storage proofs for blocks, it generates TIMC transactions for the settlement layer and sends them to the settlement layer.
11. Confirmation by the settlement layer of this TIMC transaction signifies the final confirmation of the transactions in that batch of blocks by the executor.

7.4. Consensus

The Axiomesh Delegated Byzantine Fault Tolerant(dBFT) consensus algorithm is the core algorithm of the Axiomesh settlement layer. It is a novel consensus algorithm that combines PoS[30] and PBFT[31] algorithms, as well as our self-developed Weighted Random Function(WRF). dBFT features Byzantine fault tolerance, fast transaction confirmation, unpredictable primary nodes, good scalability, and security stability.

The majority of the process of dBFT consensus is similar to PBFT consensus, including three main phases: pre-prepare, prepare, and commit. In addition, dBFT introduces mechanisms such as checkpoint confirmation, block producer election, validator election, enhanced punishment for abnormal nodes, and epoch change to improve the security and stability of dBFT consensus.

7.4.1. *Checkpoint confirmation*

After the executor of the settlement layer completes the execution of the transaction set within a block, it notifies the consensus module of the execution result of the current block (block hash, block height, etc.). When a block reaches a checkpoint, validator nodes reach consensus on the block information to ensure the consistency of the data custodian layer. After the consensus on the state check is passed, the consensus network can continue with a new round of block consensus. The state checkpoint mechanism ensures the final consistency of the data, and validator nodes do not need to reach consensus on the execution results of each block during the block packaging process, effectively improving the performance of the consensus network. In an Axiomesh network using the dBFT algorithm, transactions can achieve sub-second confirmation.

7.4.2. *Block producer election*

In the traditional PBFT consensus algorithm, block producers are fixed unless they misbehave and are changed. However, fixed validator nodes are more susceptible to attacks on the entire network (such as MEV attacks, transaction/account blocking, etc.), and they also fail to fairly incentivize every node in the network (the majority of incentives are obtained by fixed block producers). Therefore, the dBFT algorithm introduces the WRF algorithm to elect block producers, ensuring the random rotation of block producers and enhancing the security of the entire network.

WRF (Weighted Random Function) is an algorithm used for random selection based on weight values. When a node joins the network, it is assigned different weights. The weight is positively correlated with the staked amount. All node weights are mapped to a range, and then a node is selected based on the range in which the provided random seed falls using a random number generator. Nodes with higher weights occupy a larger proportion in the range and are correspondingly more likely to be selected. The advantages of the WRF algorithm include:

- Flexibility and Customization: Different weights can be assigned to each node based on specific requirements (such as staking ratio, governance allocation, etc.), thus flexibly controlling the probability distribution of random selection.
- Efficiency: Since the random seed is content already passed by consensus, there is no need for a new round of consensus on the random seed; only basic mathematical calculations are involved.
- Unpredictability: As the confirmation of the random seed requires the completion of block production in the previous state checkpoint, the random number cannot be predicted in advance, and the set of validator nodes cannot be confirmed. Similarly, block producer selection also requires the completion of block production in the previous state checkpoint, and the current term's block producers cannot be predicted in advance.

7.4.3. *Validator node election*

Due to the high communication complexity, the scalability of the traditional PBFT algorithm is significantly inferior to other classic BFT algorithms such as Tendermint[32] and Hotstuff [33]. In the dBFT algorithm, to address the scalability bottleneck of the consensus network and adapt to large-scale node blockchain networks, two different node roles are introduced:

- **Validator Nodes**: Nodes participating in the consensus network are called validator nodes.
- **Candidate Nodes**: Candidate nodes do not participate in consensus but maintain synchronization with the consensus network and continuously update the latest blocks to their ledgers. At each epoch change, candidate nodes are randomly elected based on the block hash of the latest checkpoint block, the current epoch information recorded in the epoch contract, and the number of validator nodes currently selected as a random seed using the WRF algorithm. At this point, candidate nodes have a certain probability of transitioning to validator nodes.

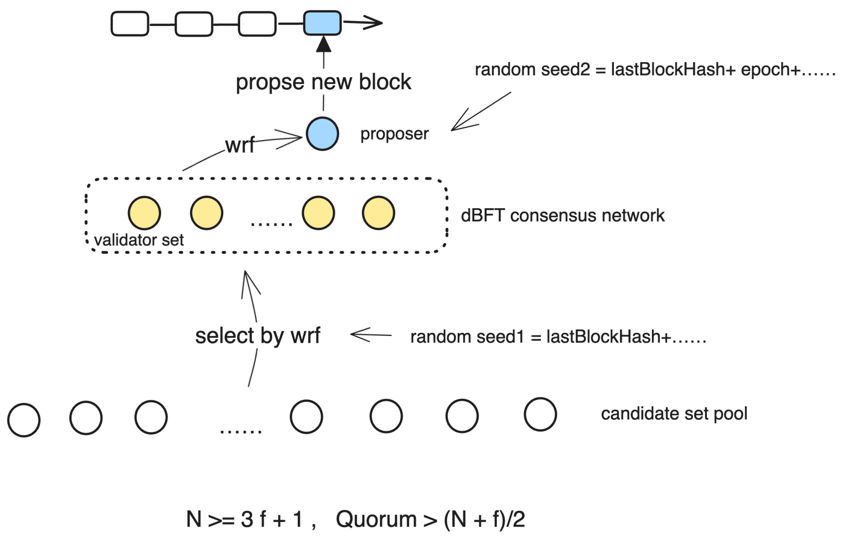
propse new block
random seed2 = lastBlockHash+ epoch+......
wrf
proposer
dBFT consensus network
validator set
......
select by wrf
random seed1 = lastBlockHash+......
......
candidate set pool
N >= 3 f + 1 ,   Quorum > (N + f)/2

Figure 7.5. *WRF election*

7.4.4. *Punishment for abnormal nodes*

7.4.4.1. *Byzantine fault tolerance (BFT)*

Inheriting the excellent properties of PBFT that can resist node misbehavior, assuming the number of malicious nodes is f and the total size of validators is N, the minimum number of nodes required for the consensus network to function normally is Quorum. The dBFT algorithm can guarantee the system's liveliness and security when the number of honest nodes is not less than Quorum (specific calculation rules refer to the following formula):

$$N >= 3f + 1, \quad Quorum > (N + f)/2$$

7.4.4.2. *Node punishment and forfeiture mechanism*

The performance bottleneck of validator nodes will directly affect the operational efficiency of the entire consensus network. To encourage honest nodes with stronger machine performance to become validator nodes, the dBFT algorithm provides node punishment measures:

- For nodes with poor network performance, their weight in being selected as the block-producing node in the next block will be reduced, decreasing their probability of being selected. The information about weight punishment is stored in the state checkpoint and is effective in the next block packaging.
- When a validator node has the right to package a block, it may want to obtain more referral rewards and thus delay the issuance of a new block proposal. In the case where the block-producing node does not package transactions, the dBFT algorithm provides a mechanism for non-block-producing nodes to actively initiate a block-producing node switch and punish the node accordingly.

After a certain number of blocks are generated, the weight of punished nodes will return to normal. In addition, for some malicious validators that disrupt the consensus network, dBFT will immediately remove the malicious nodes from the consensus network and forfeit their staked bonds. They can only rejoin the consensus network after the observation period expires.

7.4.5. *Epoch change*

Axiomesh defines the update cycle of blockchain configuration as an epoch. The blockchain configuration within each epoch is fixed, and modifications to the configuration for the next epoch can be made within the current epoch. The update of blockchain configuration is achieved through epoch changes. Epoch changes typically involve the reselection of validator nodes or relevant configuration changes, which are completed through the built-in epoch management contract in the executor. This contract contains historical epoch information, and epoch information usually consists of epoch number, epoch block interval, on-chain configuration, validator node set, etc. The introduction of epoch facilitates on-chain governance upgrades, and the dynamic rotation of validator nodes introduces decentralization and fairness to various nodes in the Axiomesh network, allowing block rewards to be distributed to more nodes.

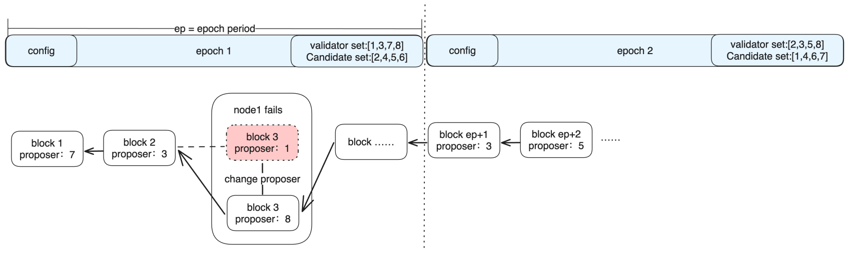

Figure 7.6. *Epoch change*

7.5. Smart Contract

In addition to supporting Solidity-based smart contracts, Axiomesh also has built-in more efficient system contracts. These system contracts are written in a native language, offering higher performance compared to Solidity. The native smart account module is primarily implemented based on system contracts. Axiomesh's system contracts are fully compatible with the EVM encoding format, making them virtually indistinguishable for users when compared to Solidity contracts, thus providing a user-friendly experience. Additionally, system contracts allow for seamless cross-contract calls with EVM contracts, enhancing the capabilities of smart contracts and delivering a superior and faster user experience. Furthermore, system contracts offer closer-to-the-metal capabilities than EVM, significantly accelerating the execution performance of system contracts, such as faster ZK circuit verification and GPU parallel acceleration.

7.6. Performance

Axiomesh employs a modular architecture, allowing for the secure enhancement of its performance through the coordinated interaction of various modules. The settlement layer of Axiomesh utilizes the dBFT consensus algorithm to achieve lower transaction confirmation times (sub-second level), ensuring minimal risk of forks. The entire throughput of Axiomesh is supported by an efficiently scalable execution layer, where multiple executors can handle transactions concurrently, with each additional executor enhancing Axiomesh's overall TPS. Additionally, the introduced data custodian layer can further reduce the data storage pressure on the settlement layer, resulting in an additional increase in the settlement layer's TPS after releasing transaction data from the execution layer.

The three layers of Axiomesh complement each other, fully extending overall performance while ensuring security. Specifically, the settlement layer of Axiomesh primarily oversees the security of funds and the entire ecosystem; the data security of executors is ensured by the data custodian layer, while execution security is guaranteed by the settlement layer, achieving high TPS without compromising security. The security of the data custodian layer also stems from the settlement layer, with staking and data availability ensuring the security of data storage.

Chapter 8

NATIVE SMART ACCOUNT

Native Smart Accounts are one of the key initiatives to lower the barrier for user adoption and realize mass commercial adoption of blockchain technology. Existing smart account solutions are mostly implemented through smart contracts and external components, which result in low performance, high gas costs, and support for only a limited number of cryptographic algorithms, making it difficult to scale up.

By supporting smart accounts natively on the chain, we can ensure higher execution efficiency, more flexible authentication methods (such as ZK authentication, aggregate signature authentication), and directly provide practical capabilities such as gas sponsorship, compatibility with Web2 login, account secret key recovery, and limited authorization within a valid period. This greatly reduces the entry barrier for developers and end-users. The core functions of smart accounts are mainly implemented through the system contracts of the chain, which are written in the native programming language. Compared to system contracts written in Solidity, there is a significant improvement in execution efficiency. At the same time, in order to embrace the Ethereum abstract account ecosystem, the system contracts of smart accounts are compatible with the Ethereum abstract account scheme, ensuring that tools such as Bundlers from the Ethereum abstract account ecosystem can seamlessly switch to Axiomesh, further enriching the ecosystem.

Overall, smart accounts are designed in a modular way, with built-in functional modules (such as email recovery, conditional authorization, aggregate signatures, zero-knowledge proofs) to enhance the functionality of smart accounts. Users can freely choose these features. In addition, users can also customize their own smart account functions to meet the personalized needs of different users.

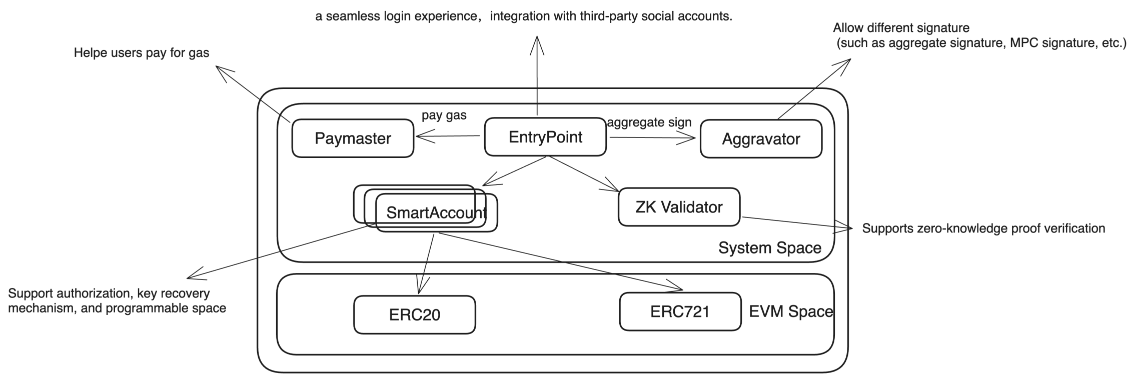
a seamless login experience，integration with third-party social accounts.
Allow different signature
(such as aggregate signature, MPC signature, etc.)
Helpe users pay for gas
pay gas
Paymaster
EntryPoint
aggregate sign
Aggravator
SmartAccount
ZK Validator
Supports zero-knowledge proof verification
System Space
Support authorization, key recovery
mechanism, and programmable space
ERC20
ERC721
EVM Space

Figure 8.1. *Native smart account*

8.1. Smart Account

Smart accounts are highly user-friendly for Web2 users, allowing account creation and login via email, social media accounts (such as Twitter, Facebook, Github, etc.). During account initialization, users log in through third-party account systems like email, deriving a private key based on the third-party account's username and password to control the smart account. The user sets a secure password to encrypt this private key and uploads and stores it on the server. The address of the user's smart account on the blockchain is generated based on the user's ID and salt, and the address corresponding to this private key is the owner of the smart account. The user controls the smart account by signing all on-chain operations with the private key. The user only needs to remember this secure password; if it is lost, it can be recovered through email or other means. Upon next login, the user downloads the encrypted private key from the server and decrypts it by entering the secure password. All on-chain operations by the user are signed with this key and sent to the blockchain, where native system contracts verify the user's signature to ensure that the identity and operations are legitimate before proceeding.

Smart accounts support various verification methods. In addition to traditional ECDSA signature verification, they also support off-chain zero-knowledge proof generation and on-chain verification, meeting users' privacy needs in specific scenarios. Smart accounts provide a zero-knowledge proof module, supporting various efficient zero-knowledge proof verification methods to meet different user requirements. Moreover, for unsupported zero-knowledge proof verification methods, users can also customize this functionality to meet complex needs.

8.2. Gas Sponsorship

Currently, the entry and usage of Web3 are highly unfriendly to Web2 users. For instance, new users wishing to use Ethereum dApps must first generate a private key and address, then deposit a certain amount of ETH into that address to cover the gas fees for using Ethereum dApp. These procedures are complex for users, particularly newcomers. Smart accounts inherently offer gas sponsorship, meaning the Paymaster of the smart account enables dApps to sponsor user operations, pay for gas in different tokens, eliminating the need for new users to purchase gas to use dApps. Consequently, users do not perceive gas fees while using dApps. This significantly reduces the user entry barriers and enhances the user experience of the product.

8.3. Recovery

If a user loses their private key, they can set a new one using social recovery. Since the private key is not saved anywhere except in the user's memory, it cannot be directly retrieved and can only be recovered by generating a new private key. Through the guardian mechanism, utilizing the DomainKeys Identified Mail(DKIM) function of email, users can set their own email domain as the guardian of their smart account. The blockchain can verify the validity of this email domain, and only after verification is passed can the owner of the smart account be modified, effectively restoring ownership of the smart account.

8.4. Session Key

Users can generate a session key that is valid for a short period, granting it certain permissions, such as limiting the number of uses, automatic expiration within a set time frame, or a specific spending limit. This prevents users from having to frequently sign authorizations when using dapps that require frequent operations, like games, greatly enhancing the user experience. The user locally generates a random private key as the session key, sends its address to the blockchain along with the authorization conditions, and then encrypts and stores the session key in the browser or local storage of the application. When user authorization is needed, the front-end/client signs the user's actions with the session key and sends it to the blockchain. The signing process is seamless for the user, significantly improving their product experience.

8.5. Multi-signature Account

In some enterprise application scenarios, it is obviously unsafe and inappropriate for the enterprise account to be managed by only one person. The person who holds the account can transfer all the assets of the enterprise, which will put the enterprise into great risks. Enterprise accounts can be controlled by multiple private keys of multiple people. Enterprise account operations require authorization from multiple people. Only after multiple people who meet the conditions are authorized to sign, the asset operation of the enterprise account can be initiated on the chain to avoid illegal transfer of enterprise assets by a single person. risks of. The smart account provides an aggregate signature function module and supports BLS aggregate signature and on-chain verification.

Chapter 9

DECENTRALIZED CONFIDENTIAL PROTOCOL WITH AUDITABILITY

The absence of privacy protection in public chains is one of the core factors hindering users, especially enterprise users, from using and building blockchain-based commercial applications. To address this issue, Axiomesh needs to provide users with powerful privacy protection capabilities. Combining design principles, Axiomesh proposes the compliant and privacy-compatible protection protocol DCPA (Decentralized Confidential Protocol with Auditability). DCPA adopts a twin-token model in its overall architecture to ensure the composability and flexibility of the protocol. It is based on cutting-edge cryptographic technology to ensure the security and privacy of transaction content and identity while ensuring regulatory compliance. In the following section, we will introduce our zkToken, zkDID and the corresponding audit flow.

9.1. zkToken

This chapter will introduce the zkToken protocol. In terms of overall architecture, zkToken utilizes a variant of Pedersen commitments[20], with its main scheme drawing inspiration from PGC[34], albeit with slight differences in detail.

9.1.1. *Twin-token protocol*

zkToken adopts twin-token protocol, which ensures the composability of the protocol with popular public chain projects while supporting flexible capital flow between plaintext and ciphertext accounts[1]. Existing privacy protocols mostly involve modifications to the underlying blockchain, which can ensure privacy but often sacrifice composability with other protocols. Additionally, many privacy protocols are relatively simplistic, treating all user data as private without distinguishing between plaintext and ciphertext, let alone enabling interaction between them.

DCPA proposes a twin-token protocol to address these two issues effectively. Users register an existing ERC20 token, such as USDT, through the twin-token protocol. This protocol then deploys a corresponding privacy contract for the ERC20, ensuring that the data in this contract is ciphertext and does not affect the original ERC20 contract.

In practice, if users opt for plaintext interaction, they directly interact with the original ERC20. For ciphertext interaction, they engage with the privacy ERC20. In cases requiring interaction between plaintext

1. For those with privacy needs, an account that encrypts transaction data is referred to as a ciphertext account, while the opposite is termed as a plaintext account

and ciphertext, such as transactions involving both types, the twin-token routing contract facilitates the transfer between the two contracts.

9.1.2. *Basic parameter*

The zkToken protocol primarily utilizes the bn254[35] curve, which is friendly to zero-knowledge proofs. Parameters related to elliptic curves such as F_p, G, and others mentioned in the following text are all based on the bn254 curve parameters.

Zero-knowledge proofs are implemented using the Groth16[36] algorithm, known for its higher verification efficiency. This choice is due to the simplicity of ERC20 logic and the controllable time it takes to generate zero-knowledge circuits, meeting production needs.

9.1.3. *Encryption*

For the information encryption component, since ciphertext calculations are required on-chain, the encryption algorithm needs to satisfy homomorphic properties. However, as on-chain ciphertext calculations only involve basic addition and subtraction operations, the encryption component of the privacy protocol mainly employs elliptic curve based encryption algorithms.

9.1.3.1. *Key pairs*

In this document, for ease of description, the user's private key will be denoted by s, and the user's public key will be represented by P. Each user randomly generates their own random number as their private key, and the public key is calculated using the following method:

$$P = s \cdot G, s \longleftarrow_R F_p$$

9.1.3.2. *Encryption flow*

For any arbitrary message m, without loss of generality, we can consider it to be of integer type. For the encryption algorithm $E(\cdot)$ of this message, we need to satisfy the following conditions:

1. Indistinguishability: For any two messages m and m', the encrypted results $E(m)$ and $E(m')$ are indistinguishable to everyone.
2. Additive homomorphism: For any two messages m_1 and m_2, $E(m_1) + E(m_2) = E(m_1 + m_2)$, where $E(\cdot)$ denotes the encryption function.

The specific encryption algorithm is as follows:

Algorithm 9.1 Encrypt message

```
Data: message $m$ to be encrypted
Result: Encrypted message $E(m)$
1 begin
2     $M \longleftarrow m \cdot G$;
3     $k \longleftarrow_R F_p$;
4     $C_1 \longleftarrow k \cdot G$;
5     $C_2 \longleftarrow M + k \cdot P$;
6     return $(C_1, C_2)$
7 end
```

Specifically, users encode the information onto an elliptic curve and generate a random number to blind the encoded result. Finally, they return the blinded result along with auxiliary data to obtain the encrypted result, which consists of two points on the elliptic curve.

The indistinguishability property is guaranteed by the elliptic curve and the random algorithm, and further elaboration is unnecessary here. Below, we will demonstrate the homomorphic property of this encryption algorithm.

For any two messages m_1 and m_2, we have the following computation:

$$\begin{aligned}
E(m_1) &= (k_1 \cdot G, m_1 \cdot G + k_1 \cdot P) \\
E(m_2) &= (k_2 \cdot G, m_2 \cdot G + k_2 \cdot P) \\
E(m_1) + E(m_2) &= (k_1 \cdot G + k_2 \cdot G, m_1 \cdot G + k_1 \cdot P + m_2 \cdot G + k_2 \cdot P) \\
&= ((k_1 + k_2) \cdot G, (m_1 + m_2) \cdot G + (k_1 + k_2) \cdot P) \\
&= E(m_1 + m_2)
\end{aligned}$$

Therefore, the encryption algorithm satisfies the homomorphic property.

9.1.3.3. *Decryption flow*

Unlike conventional encryption and decryption algorithms, our decryption algorithm consists of two parts: restoration and decoding. The restoration part is mainly responsible for computing the value of the elliptic curve point M, while the decoding algorithm is responsible for deriving the original message m from M.

The restoration part algorithm is as follows:

Algorithm 9.2 Decrypt message

```
Data: message $(C_1, C_2)$ to be decrypted
Result: decrypted message $m$
1 begin
2     $M \longleftarrow C_2 - s \cdot C_1$;
3     return $M$
4 end
```

Its correctness is derived as follows:

$$C_2 - s \cdot C_1 = M + k \cdot P - s \cdot k \cdot G = M + k \cdot P - k \cdot P = M$$

In the decoding part, due to the encryption algorithm encoding plaintext onto an elliptic curve, this transformation from the real number space to the elliptic curve space is conventionally considered one-way due to the existence of the discrete logarithm problem on elliptic curves. However, we can utilize the Baby-step Giant-step algorithm[37] to efficiently solve the above problem within a certain range.

9.1.3.4. *Validity*

Since the information received on-chain consists of encrypted data generated by on-chain users, the blockchain cannot verify the legitimacy of such transactions in the same way it verifies regular transactions, such as checking if the transferred amount exceeds the user's account balance. To ensure the validity of transactions, we need to use zero-knowledge proofs to verify the user's inputs. The specific verification content includes:

1. Whether the encrypted amount exceeds the account balance;
2. Whether the ciphertext encrypted with the public keys of both parties is generated from the same transferred amount;
3. Whether the encrypted plaintext balance of the user matches the on-chain ciphertext;
4. Whether the view key has been encrypted using the auditor's public key;
5. Whether the transaction meets audit requirements, such as not exceeding the single transaction spending limit;

There will be verification contracts to validate the specific transaction's validity on-chain. Transactions that fail verification will be rolled back.

9.1.4. *Transaction flow*

In this chapter, we will combine the previous sections to describe the complete transaction process. If Alice wants to transfer some tokens m to Bob, the overall process is as follows, where the verification part occurs on-chain, and the rest occurs off-chain:

Algorithm 9.3 Alice Transfer an ERC20 to Bob

```
Data: value $m$, Alice's public key $P_A$, Bob's public key $P_B$
Data: Alice's Balance $(Ba_1^A, Ba_2^A)$, Bob's Balance $(Ba_1^B, Ba_2^B)$
begin
    $(C_1^A, C_2^A) \longleftarrow E_{P_A}(m)$;
    $(C_1^B, C_2^B) \longleftarrow E_{P_B}(m)$;
    $Proof \longleftarrow ZK$;
    if Validate($Proof$) then
        $Ba_1^A \longleftarrow Ba_1^A - C_1^A$;
        $Ba_2^A \longleftarrow Ba_2^A - C_2^A$;
        $Ba_1^B \longleftarrow Ba_1^B + C_1^B$;
        $Ba_2^B \longleftarrow Ba_2^B + C_2^B$;
    else
        revert
    end
end
```

Overall, users will encrypt the message using their own public key before transferring, and then encrypt the message using the recipient's public key. These two encrypted contents will be used to update the on-chain balances of both the sender and the recipient.

9.2. zkDID

We are well aware that zkToken still cannot adequately protect privacy. For example, when users go to merchants for consumption, they can obtain the merchant's address. By querying all transactions at the merchant's address, although the specific transaction amounts cannot be known, the frequency of payments to the merchant can still be inferred. Based on the merchant's average price level, deductions can be made about the merchant's turnover, and insights can also be gained into the distribution of the merchant's clientele. Therefore, apart from the privacy of transaction amounts, a good privacy protocol still needs to possess a certain level of anonymity.

a

9.2.1. *Stealth address*

In this chapter, we will introduce the implementation of the stealth address technology in the BaseSAP protocol. The following symbol definitions will be used in the following text:

- (s, S): Represents the public-private key pair of the sender.
- $(r_{sc}, R_{sc}), (r_{sp}, R_{sp})$: Represent the Scanning and Spending public-private key pairs of the recipient, respectively.
- (r_{st}, R_{st}): Represents the recipient's stealth address.
- (e, E): Represents the public-private key pair of the regulator.
- G: Represents the generator point of curve bn254.
- $H(\cdot)$: Represents the hash function.

The Spending public-private key pair is the actual account used by the recipient, capable of controlling account expenditure. The Scanning public-private key pair is randomly generated by the recipient for generating stealth addresses and is only used for receiving payments but cannot spend funds. The use of the Scanning public-private key pair is necessary because it is impossible to host funds solely with the Spending public-private key pair, which would increase the difficulty for users.

The specific process is as follows:

1. The sender computes the receiver's stealth address $R_{st} = H(s \cdot R_{sc}) \cdot G + R_{sp}$ and sends the message to R_{st}.
2. The recipient listens for on-chain events and computes own stealth address $R_{st} = H(r_{sc} \cdot S) \cdot G + R_{sp}$. It then compares the computed result with the on-chain R_{st} to ensure consistency. If they match, the recipient proceeds to the next step.
3. The recipient computes the private key of the stealth address $r_{st} = H(r_{sc} \cdot S) + r_{sp}$. It is evident that r_{st} is the private key corresponding to R_{st}.

Therefore, at this point, the recipient can control the funds in the receiving account. As the sender lacks the private key of the recipient's Spending public-private key pair, they cannot obtain the private key information of the stealth address.

9.2.2. *Transaction flow*

Algorithm 9.4 Transfer to stealth address

```
Data: Sender's key pair $(s, S)$
Data: Receiver's scanning key pair $(r_{sc}, R_{SC})$
Data: Receiver's spending key pair $(r_{sp}, R_{sp})$
Data: Audit's key pair $(e, E)$
Sender:
begin
    $R_{st} \longleftarrow H(s \times R_{sc}) \times G + R_{sp}$;
    $\tau \longleftarrow R_{sp} + s \times E$;
    $Proof \longleftarrow ZK$;
    send tx to $R_st$;
end
Receiver:
begin
    $R'_{st} \longleftarrow H(r_{sc} \times S) \times G + R_{sp}$;
    if $R'_{st} = R_{st}$ then
        $r_{st} \longleftarrow H(r_{sc} \times S) + r_{sp}$;
    else
        return;
    end
end
```

In addition to the process described in the previous section, the sender needs to upload two additional components before sending:

1. τ: Used by regulators to recover the actual recipient.
2. $Proof$: Used to prove the correctness of the computation.

9.3. Auditability

Compliance has always been an area Axiomesh has been exploring and actively cooperating with. We have witnessed many privacy projects becoming hotbeds for illegal activities such as money laundering, drug trafficking, and arms trading. As a blockchain project aiming for large-scale commercial use, our goal is to safeguard privacy under the premise of compliance. To achieve this, we have made significant efforts.

Axiomesh's compliance is generally divided into two aspects: ZK-based audit rule verification and MPC-based view key technology. This two-tier audit model ensures secure verification of transaction compliance both before and after transactions occur, thereby preventing the occurrence of illegal activities.

9.3.1. *Audit rule*

Based on audit rule verification includes common audit scenarios, and embeds the verification rules in both the ZK module and the on-chain contract module. Specifically, the verification rules include:

1. Transaction limit: Users specify that the transaction limit within a certain period (such as per transaction, per day, or per week) must not exceed a specific limit.

2. Transaction frequency limit: Users must not send more than a specific number of transactions within a unit time.

9.3.2. *View key*

9.3.2.1. *zkToken view key*

In section 9.1, we outlined the specific encryption process of zkToken. We use public keys for encryption and private keys for decryption. However, in addition to the private key, the random number k generated by the user for each transaction can also be used for decryption. The specific decryption method is as follows:

$$C_2 - k \cdot P = M + k \cdot P - k \cdot P = M$$

We refer to the random number k here as the view key. The decryption scheme using the view key has several advantages:

1. Any party obtaining the view key can only decrypt a specific transaction, rather than all transactions. This provides us with more precise access control.

2. The view key can only decrypt encrypted transactions and cannot be used to initiate transactions, ensuring the security of user funds.

Users generate a view key for each transaction randomly. However, this key cannot be directly uploaded to the blockchain. It needs to be encrypted using asymmetric encryption before uploading. The public key of asymmetric encryption can be stored on the blockchain.

9.3.2.2. *zkDID view key*

In section 9.2, we outlined the overall process of zkDID. In this process, the sender needs to send additional information τ during the transaction, which we refer to as the view key of the anonymous part in this transaction. The specific decryption method is as follows:

$$\tau - e \cdot S = R_{sp} + s \cdot E - e \cdot S = R_{sp}$$

At the same time, to ensure the correct generation of the view key, it needs to be verified off-chain using zero-knowledge proofs to ensure the authenticity of on-chain data.

9.3.2.3. *Key management*

In the view key sections of zkToken and zkDID, we described the decryption method for the view key. However, the storage of private keys therein poses a significant challenge. Axiomesh employs a distributed solution combining secret sharing and Trusted Execution Environment (TEE) to address this issue.

Key management in Axiomesh is handled by Data Custodian nodes and primarily consists of three functionalities: Sharding, Recovery and Re-sharing.

9.3.2.4. *Sharding*

Due to the simplicity of the requirements, there is no need for a large number of sharing nodes, so we have adopted Shamir Secret Sharing[38] as the basic algorithm for secret sharing.

Shamir Secret Sharing has the ability to share a secret with a threshold, meaning a key will be divided into several parts, but it is not necessary for all key holders to be present to recover the key. This algorithm possesses strong robustness, which is why we chose it as our secret sharing algorithm.

The Data Custodian nodes have built-in TEE modules. During initialization, one module will be randomly elected among all Data Custodian nodes. This module will generate random public-private key pairs in the TEE and then shard the private key, sharing the shards with the other Data Custodian nodes.

9.3.2.5. *Recovery*

The process of key recovery is primarily initiated by the auditing party. After verifying the identity at the Data Custodian, the auditing party sends a recovery request to the other Data Custodian nodes. Upon receiving the key shards, the key recovery is completed within the TEE.

9.3.2.6. *Re-sharing*

To prevent excessive withdrawal of Data Custodian nodes holding key shards, key re-sharing can be employed under certain conditions to re-shard the key. To ensure that past on-chain transactions can still be recovered, key re-sharing does not involve generating a new key. Instead, it first recovers the old key and then performs re-sharding.

9.3.2.7. *Audit flow*

The auditing agency sends a data recovery request to the Data Custodian nodes, which includes the encrypted view keys corresponding to all on-chain transactions that need decryption. The Data Custodian nodes then send recovery requests to other nodes and recover the private keys within the TEE. They decrypt all the encrypted read-only keys and return the decryption results. The auditing agency decrypts these encrypted transactions using the read-only keys obtained, thus gaining access to the details of all transactions requiring auditing.

Chapter 10

TRUSTED INTER MODULE COMMUNICATION

The TIMC protocol is a protocol similar to IBC[39] of Cosmos and XCMP[40] of Polkadot, which is used to transmit data and value between modules within Axiomesh in a secure and reliable way. On this basis, it also provides some more powerful capabilities:

- TIMC offers a unified message passing mechanism and a flexible, customizable Rollup mechanism for the execution layer. It does not restrict the execution layer to using ZKRollup or OPRollup, nor does it limit the choice of ZK algorithms. The execution layer only needs to implement the verification contract interface.
- Leveraging the consensus security of the settlement layer and the Rollup security of the execution layer, TIMC enables secure transfer of data and assets from the execution layer to the settlement layer and other execution layers. Moreover, it allows for interoperability with other blockchains like Ethereum and Solana through public cross-chain bridge protocols. Due to the extended validity verification period of messages issued by the OpRollup executor, there may be a high latency in message processing. For executors with high real-time requirements, the sender executor can individually stake a bond to the recipient executor to accelerate message reception and processing. Each time a message is transmitted, a portion of the bond is locked. After the batch containing the message is confirmed, this portion of the bond is unlocked. The recipient executor can immediately receive and process the message. If a message is proven to be invalid, both the message and the subsequent locked bond amount will be compensated to the recipient executor.
- For the secure transfer of native tokens between the settlement layer and the execution layer, the TIMC module in the execution layer will maintain a separate balance tree of standard execution layer accounts. When an executor performs Rollup, the TIMC module will handle the account balance tree state transition.
- TIMC limits the number of execution layers in order to balance scalability and the pressure on the settlement layer. The permission to use an execution layer (which we call a Pipe) is rented through an auction of native tokens. The auction winner obtains the right to use the Pipe and locks up the auction funds for later consumption by the execution layer during Rollup. At the end of the Pipe lease term, the previous owner has the right to renew the lease.
- TIMC mandates that execution layers under Axiomesh use settlement layer native tokens locked on the corresponding execution layer for gas payment. This enhances the liquidity of Axiomesh settlement layer tokens and eliminates the need for users to convert additional tokens when using/withdrawing from execution layers. The specific gas price is determined by each execution layer. During Rollup, the execution layer generates and submits balance transition ZK proofs for the execution

layer's Miner account through TIMC (to ensure proof validity, the Miner is a special address on the executor that cannot receive or transfer tokens).

- To prevent settlement layer data from expanding due to the continuous submission of execution layer transactions, Axiomesh supports execution layers uploading transactions to the DC layer for secure/privacy-preserving storage. During Rollup, the execution layer submits additional storage proofs, and TIMC also operates a light client of the DC layer to assist with verification.

The overall overview of the TIMC protocol is depicted in the following figure:

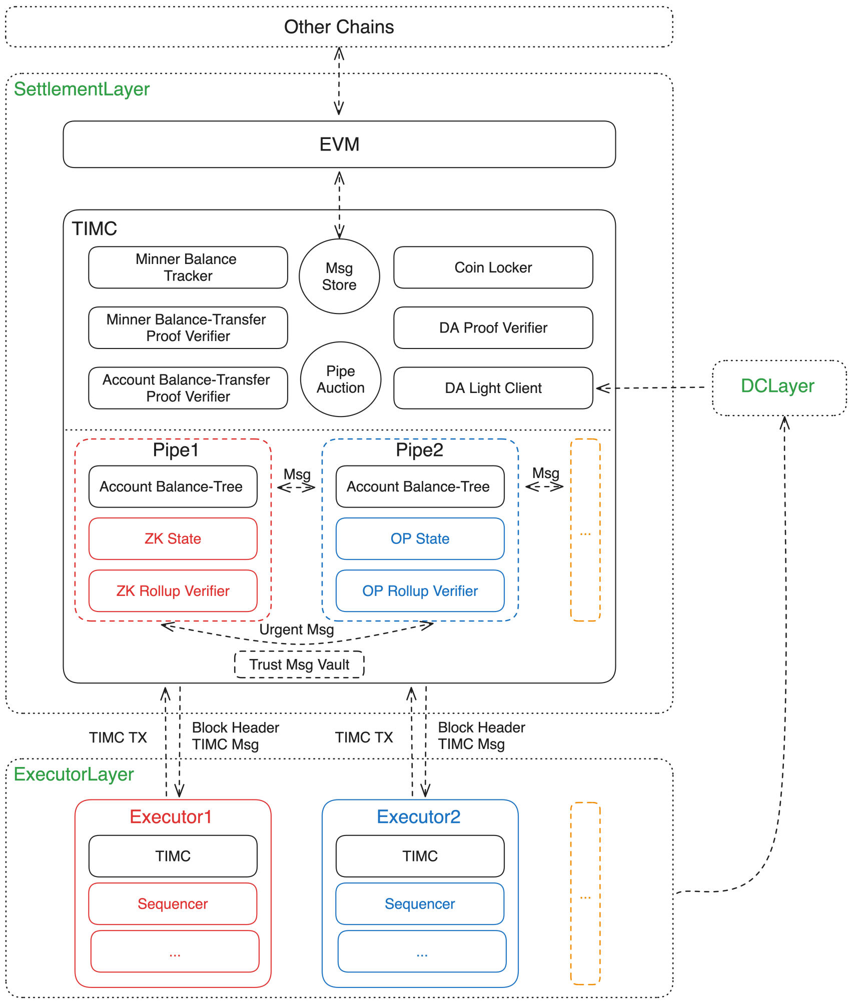

Figure 10.1. *TIMC architecture*

10.1. Pipe Registration

For entities wishing to become Axiomesh executors, participation in the auction for the usage rights of execution layer Pipes is mandatory. This involves pledging Axiomesh tokens. The duration of usage rights for Pipes auctioned off is fixed. The auction is held periodically at the settlement layer. After the auction period ends, the bidder with the highest token pledge will acquire the leasing rights for the Pipe being auctioned. Subsequently, they can develop their executor integrated with Axiomesh's TIMC module and register executor information with Axiomesh's TIMC to officially operate the executor.

After the auction ends, the tokens pledged by bidders who did not win the lease rights for the Pipe will be automatically refunded. The tokens pledged by the winning bidder will be locked. Once the executor information is registered with TIMC, TIMC will generate an address for the executor and transfer the tokens locked by the winning bidder during the auction to the executor's address. This fund consists of two parts: 1. Used for gas consumption during subsequent Rollups (gas price is dynamically calculated and maintained by TIMC module of the settlement layer, independent of the gas price calculation method for regular transactions on the settlement layer); 2. Used for security guarantees of the executor. Since the verification mechanism of the executor using Rollup is independently maintained and registered in Pipe by the executor through contracts, Axiomesh cannot guarantee the security of user assets in case of issues with the executor. Therefore, this fund also serves as compensation in case the executor is attacked. The starting collateral is adjusted to ensure that the fund can adequately cover the above two purposes.

Upon expiry of the Pipe lease, the original lessee of the Pipe has the priority to renew the lease. The funds required for renewal are dynamically adjusted by the TIMC module of the settlement layer based on the network situation and the load of the executor on the settlement layer. When the original lessee of the Pipe opts not to renew, the executor's status will be frozen, and the corresponding Pipe will be released and participate in the auction. Users' funds can be safely withdrawn via an escape mechanism. Note that the status of this executor can still be unfrozen and continued after obtaining the lease rights for the Pipe in the future.

To lower the requirements for dApp operators to obtain the usage rights of Pipes, Axiomesh will reserve a certain number of Pipes to be operated by the public executor network. Executor nodes only need to pledge a small amount of funds to join the public executor network. They will share the gas fee revenue of transactions in the network with other executor nodes. These nodes will submit Batches to the settlement layer to earn gas revenue in turns according to a certain algorithm. When some nodes misbehave, they will be penalized and expelled from the executor network. dApp operators with no special requirements can deploy their dApps in the public executor network.

10.2. Status Confirmation

Rollup from the executor to the settlement layer refers to the process where transactions from users within the executor are confirmed by the executor before being finally confirmed by the settlement layer. Transactions that are only confirmed by the executor may be invalidated due to malicious behavior or attacks on the executor. The rollup method of the executor is customizable, and the TIMC protocol of Axiomesh supports executors to register custom Rollup verification contracts, and supports subsequent dynamic updates and upgrades. Of course, Axiomesh also provides pre-implemented verification contracts for OPRollup and ZKRollup as well as default executor implementations. For specific performance or functional requirements, it is also possible to fork changes based on standard implementations, as long as they comply with the TIMC protocol standards.

The entire Rollup process is implemented through communication between the settlement layer's TIMC module and the executor's TIMC module. The overall process is as follows:

1. The executor typically batches these blocks periodically into a Batch, generates execution proofs for the batch based on the Rollup algorithm, and submits them to the executor's TIMC module along with the Batch.

2. The executor's TIMC module uploads the complete Batch data to the DC layer based on Batch information, waits for DC layer confirmation, and obtains storage proofs.

3. The executor's TIMC module generates a proof of transferring gas fees from transactions in this Batch to the executor's Minner address.

4. The executor's TIMC module maintains the state of the account balance tree. After executing the transactions in this batch, a proof of updating the account balance tree will be generated accordingly.

5. The executor's TIMC module packages Batch information, storage proof, execution proof, Minner balance update proof, and account balance tree update proof into a settlement layer TIMC transaction and sends it to the settlement layer.

6. The settlement layer's TIMC module, upon receiving the TIMC transaction, executes corresponding verification logic:

   - Verifies the validity of the storage proof through the DC layer light client in the TIMC module.
   - Validates the execution proof through the Rollup verification contract registered in the Pipe.
   - Ensures the validity of gas fee collection via native token through the Minner balance update verification contract in the TIMC module (to ensure that the executor collects transaction gas fees using Axiomesh's native tokens).
   - Validates the changes in executor account balances through the account balance tree update verification contract in the TIMC module (to ensure the safety of user assets when extracting and transferring funds in the executor, also providing users with a minimum fund escape route through this verification contract).

7. After the settlement layer's TIMC module verifies the TIMC transaction, it executes the processing logic of the TIMC transaction and packages it into a block in the settlement layer.

8. When the TIMC transaction sent by the executor is confirmed by the settlement layer, the settlement layer's TIMC's light client monitors this information and marks the previous batch awaiting confirmation as finally confirmed.

The above describes how the executor leverages the TIMC protocol to have a batch of blocks confirmed by the settlement layer.

10.3. Capital Flow

10.3.1. *Deposit*

The TIMC protocol ensures liquidity of settlement layer tokens in the executor layer, meaning both the settlement layer and all executors will use a common token for settling gas fees. When users want to interact with a dApp located on a particular executor, they first need to transfer some AXC from the settlement layer to the executor (or directly purchase tokens on the executor), a process also known as depositing.

User deposits are implemented by calling the deposit method of the settlement layer's TIMC module external contract. When this transaction is executed, the user's funds are transferred and locked into the corresponding executor's Pipe, and an entry deposit record is attached to the block header, awaiting processing by the corresponding executor.

When the executor's TIMC module detects this record, it generates a system transaction for the deposit, which is later included in a Batch and rolled up to the settlement layer. The deposit is marked as processed with the help of an account balance tree update proof.

10.3.2. *Gas payment*

When users send transactions on the executor, gas fees are deducted using the native token on the executor. Gas prices are determined by the executor itself, and all gas fees from transactions are transferred to a pre-defined Minner account on the executor (the Minner account is a special address on the executor that cannot send or receive transfer transactions; on the settlement layer, it is controlled by the executor's Pipe contract). The settlement layer maintains the latest balance of the Minner account, updating it during Rollup with balance update proofs. Executors can then extract collected gas fees on the settlement layer to ensure liquidity of settlement layer tokens within the executor.

10.3.3. *Withdrawal*

When users wish to withdraw funds from the executor to the settlement layer or transfer them to another executor, they need to call the withdrawal method of the executor's TIMC module. The executor's TIMC module processes the transaction, updates the user's balance, and includes the withdrawal operation in subsequent TIMC transactions sent to the settlement layer during Rollup. The settlement layer verifies the validity of the account constraint update proof. If withdrawing to the settlement layer, the funds are directly transferred to the receiving address. If transferring funds to another executor, a withdrawal record is generated and attached to the block header, initiating the withdrawal process.

Note that the speed of fund withdrawal depends on the type of executor; ZKRollup offers faster state confirmation, hence faster withdrawals, while OPRollup's state confirmation requires a longer wait.

10.4. Convey Witness

The previous sections described the flow of native token funds between the settlement layer and the executor layer. However, for the transmission of ordinary messages (i.e., facts), the TIMC module also provides a similar mechanism to ensure reliable message transmission. Since the security of the executor layer in Axiomesh is guaranteed by the settlement layer, message transmission in Axiomesh does not require a protocol like Cosmos' IBC, which requires both chains involved in the transmission to run each other's light clients. The TIMC module only needs to hold a light client of the settlement layer, eliminating the need to run multiple light clients, significantly reducing the complexity of maintaining components during network expansion and lowering the development and adaptation costs for message transmission between any two executors.

When users want to send a message to a dApp on the settlement layer or another executor, they need to call the sendMsg method of the TIMC module contract on the executor. Similar to the fund withdrawal process, once the transaction is executed, the TIMC module attaches the message to subsequent TIMC transactions sent to the settlement layer during Rollup. Upon verification of validity by the settlement layer, if the message recipient is a dApp on the settlement layer, the message is stored in the Msg Store for dApp consumption. If the message recipient is a dApp on an executor, the message is attached to the block header's msg list, awaiting consumption and processing by the receiving executor.

Note that for executors of the ZKRollup type, message transmission is relatively fast due to quick state confirmation. However, for OPRollup, state confirmation requires a longer waiting period, which is unacceptable for messages requiring high timeliness. To address this issue, the TIMC protocol supports the sender executor staking a certain amount of tokens in the Trust Msg Vault to achieve the ability for fast confirmation.

Chapter 11

TOKENOMICS

Axiomesh aims to establish a compatible incentive-based economic model, providing a comprehensive and rich economic framework for Axiomesh builders, AXC holders, as well as a wide range of potential users and applications. This chapter will provide a detailed explanation of the economic model.

11.1. AXC

AXC (Axiomesh Credit, hereinafter referred to as AXC) is the basic unit of circulation in the Axiomesh network. Similar to traditional currencies where the smallest unit of the dollar is cents, AXC also has a minimum counting unit, which we refer to as "Mol". The specific conversion relationship is:

$$1\ \texttt{AXC} = 10^9\ \texttt{GMol} = 10^{18}\ \texttt{Mol}$$

As the fundamental unit of circulation in Axiomesh, AXC will play a role in various key aspects such as incentives, usage, governance, and staking. Additionally, AXC plays a central role in the governance of Axiomesh.

11.2. Use Cases of AXC

The main applications of AXC include:

- **Service Fees**: This comprises three main components
  - Gas Fees: AXC serves as the native token of Axiomesh and is used as the sole means of payment for transactions on both the settlement and execution layers of Axiomesh. It is also the only native asset of the Axiomesh ecosystem.
  - Storage Fees: Payment for plaintext and privacy storage services provided by Data Guardians.
  - Privacy Fees: Payment for privacy asset services provided for the settlement and execution layers.
- **Proof of Stake (PoS)**: Axiomesh is a permissionless blockchain network where node operators can stake a sufficient amount of AXC to gain the right to validate transactions on the network and earn rewards from validation. Axiomesh also supports Delegated PoS, allowing users to stake their AXC and delegate it to specified node operators to earn PoS rewards.
- **Governance**: AXC holders will have the opportunity to participate in the governance of Axiomesh in the future.

11.3. Gas Fees

Gas Fee is a common mechanism in blockchain design to prevent DDoS attacks. To prevent an excessive number of requests from overloading the blockchain, all users of the blockchain system are required to pay a fee associated with the complexity of their transactions. We also refer to this fee as the Gas Fee. The specific calculation logic is as follows:

$$\texttt{Gas Fee} = \texttt{Gas Price} \times \texttt{Gas Used}$$

Gas Price refers to the unit price of each Gas, and Gas Used represents the amount of Gas consumed by the current transaction. To draw an analogy with automotive fuel, Gas Price is akin to the price of fuel, while Gas Used indicates the quantity of fuel consumed during the journey. Therefore, the total cost of a trip is the product of the fuel price and the fuel consumption.

Axiomesh's Gas Fee involves three aspects:

1. Validator: In this aspect, Gas is similar to that of mainstream blockchains like Ethereum. Users pay AXC to validators for the computational resources consumed.
2. Data Custodian: To save storage space on-chain, a large amount of data is stored on Data Custodian nodes. In this case, users need to pay AXC for the storage resources provided by Data Custodians.
3. Executor: In this aspect, Gas is similar to using Layer 2 projects like Optimism. Users pay AXC for the computational resources consumed by the Executor when submitting transactions to the Validator.

Users do not need to pay for all three parts simultaneously. Compared to traditional blockchain projects, Axiomesh's Gas Fee is cheaper and more stable.

To ensure dynamic adjustment of network conditions and stable node services, Axiomesh will adopt a dynamic adjustment mechanism for Gas Price. Gas Price will be dynamically adjusted according to the network's congestion levels.

11.4. Mining

Axiomesh incentivizes node operators to provide high-quality services by rewarding the nodes that produce blocks. This reward is referred to as the Mining reward.

11.4.1. *Lock*

To ensure that nodes provide a more stable service quality and to prevent malicious behavior, Axiomesh will adopt a lock model. Node operators need to lock a certain amount of AXC as collateral. This collateral will be refunded the remaining amount when the node exits. If there are no slash losses incurred during the node's operation period, the collateral will be fully refunded.

11.4.2. *Penalty*

When a node encounters issues during operation, such as network fluctuations leading to unstable service, the node will be penalized. If the penalized node is a block-producing node, all rewards for that block will be forfeited, including Gas rewards and block rewards. Additionally, the node's probability of producing blocks will be reduced.

11.4.3. *Slash*

When a node is repeatedly penalized during operation, the node will be subject to slash of a portion of its collateral. The slash process will last for a maximum of one month. Specifically, upon slash, 20% of the node's collateral will be immediately deducted. Over the following four weeks, 20% of the total collateral will be deducted weekly until complete slash is achieved. Node operators can take action at any time after slash occurs. Specifically, they can:

1. Top up the collateral to the required amount. In this case, the node will be removed from the slash list and placed in the observation list. During this one-month observation period, the node cannot be elected as a Validator. After the observation period expires, the node will be reconnected to the network.
2. Apply for exit via governance. Upon approval of the exit application, the system will return the remaining collateral to the original account.

11.5. Referral Incentive

Through referral incentives, the aim is to incentivize Axiomesh contributors to enhance ecosystem influence. The referral reward specifically refers to the ability for transactions to carry a "referral address" field. When a block is eventually produced, the Axiomesh network will verify and issue a referral reward to the "referral address".

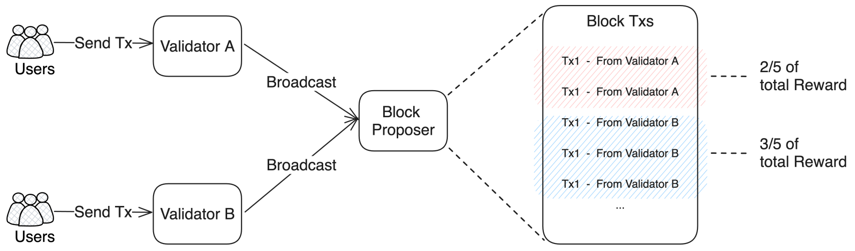

Figure 11.1. *Referral incentive flow*

We will present the complete token economics model of AXC in a dedicated whitepaper at a later time.

CONCLUSION

The core vision of Axiomesh is to build a blockchain infrastructure capable of supporting mass commercial adoption, powering thousands of Web3 commercial applications and driving the development of the digital economy. To achieve this ambition, we have proposed a modular blockchain architecture with unified liquidity and innovatively introduced protocols such as Native Smart Accounts system, Decentralized Confidential Protocol with Auditability, and Trusted Inter Module Communication protocol.

Axiomesh supports native smart accounts, ensuring efficient execution and flexible identity authentication. It also provides practical features such as gas sponsorship, seamless login with Web2 accounts, and account recovery. These significantly lower the barriers for traditional developers and end-users to enter the Web3 ecosystem, thereby can effectively facilitating the migration of large-scale Web2 users to Web3.

The DCPA protocol of Axiomesh utilizes advanced cryptographic technologies to ensure high security and privacy of transaction data and user identities while complying with regulatory requirements. This solves the challenges of maintaining sensitive information for individual users and business secrets for enterprise users in the public chain commercial environment, thereby eliminating users' privacy concerns in the mass commercial adoption of blockchain.

Additionally, Axiomesh adopts a TIMC protocol to securely and reliably transmit data and value between its internal components. Through this protocol, Axiomesh can scale execution and storage to meet the high-performance processing requirements and massive data storage needs of mass commercial applications.

Furthermore, a sustainable economic model provides robust guarantees for the sustainable development of the Axiomesh ecosystem.

In summary, Axiomesh will empower the real economy with Web3 technology, drive the widespread adoption of Web3 applications, and realize mass commercial adoption of blockchain. As blockchain diversifies, it will provide a more equal and trustworthy cooperation environment for enterprises and users to participate in production activities driven by interests, thereby building a novel production relationship centered on trust and fostering new business models. Ultimately, the mass commercial adoption of blockchain is expected to provide powerful innovation momentum for economic growth on a global scale.

REFERENCE

[1] B. H. Digital, "The relentless rise of stablecoins." https://mirror.xyz/0x63F8A82711ECA999D5Cf588d5b7afE4DB673C7ED/ekt7LwxhuqZ4t1q5CN1oOHR3nlQGglMpaCB3ZBu2B3Y. Accessed:2024-02-20.

[2] R. A. Emre Tiftik, Khadija Mahmood, "Global debt monitor politics and climate finance in a high-debt world." https://www.iif.com/portals/0/Files/content/Global%20Debt%20Monitor_Nov2023_vf.pdf. Accessed: 2024-03-13, Published:2023-11-16.

[3] "Top assets by market cap." https://companiesmarketcap.com/assets-by-market-cap/. Accessed: 2024-03-13.

[4] "Cryptocurrency prices by market cap." https://www.coingecko.com/. Accessed: 2024-03-13.

[5] S. Kumar, R. Suresh, D. Liu, B. Kronfellner, and A. Kaul, "Relevance of on-chain asset tokenization in 'crypto winter'." https://web-assets.bcg.com/1e/a2/5b5f2b7e42dfad2cb3113a291222/on-chain-asset-tokenization.pdf. Accessed: 2024-2-15.

[6] J. Thorpe, Y. Qiao, J. Eyolfson, S. Teng, G. Hu, Z. Jia, J. Wei, K. Vora, R. Netravali, M. Kim, and G. H. Xu, "Dorylus: Affordable, scalable, and accurate {gnn} training with distributed {cpu} servers and serverless threads," in *15th USENIX Symposium on Operating Systems Design and Implementation (OSDI 21)*, pp. 495–514, 2021.

[7] S. G. Sami Kassab, "State of DePIN 2023." https://messari.io/report/state-of-depin-2023, Jan. 2024. Accessed: 2024-2-15.

[8] D. for Digital, "Paperless trade for uk businesses to boost growth." https://www.gov.uk/government/news/paperless-trade-for-uk-businesses-to-boost-growth. Accessed: 2024-2-20, Published:2022-10-12.

[9] Commonwealth, "Quantitative analysis of the move to paperless trade." https://messari.io/report/state-of-depin-2023. Accessed: 2024-2-20.

[10] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[11] "Ethereum whitepaper." https://ethereum.org/whitepaper. Accessed: 2024-2-4.

[12] O. Avellaneda, A. Bachmann, A. Barbir, J. Brenan, P. Dingle, K. H. Duffy, E. Maler, D. Reed, and M. Sporny, "Decentralized identity: Where did it come from and where is it going?," *IEEE Communications Standards Magazine*, vol. 3, no. 4, pp. 10–13, 2019.

[13] "What is a crypto hardware wallet and how does it work?." https://crypto.com/university/what-is-a-hardware-wallet. Accessed: 2024-03-12, Published:2022-06-17.

[14] S. Jain, "What is a multi-party computation (mpc) wallet?." https://www.alchemy.com/overviews/mpc-wallet. Accessed: 2024-03-12, Published:2022-09-02.

[15] "Metamask." https://metamask.io/. Accessed:2024-03-13.

[16] "Ethereum name service." https://ens.domains/. Accessed:2024-03-13.

[17] V. Buterin, “Erc 4337: account abstraction without ethereum protocol changes.” `https://medium.com/infinitism/erc-4337-account-abstraction-without-ethereum-protocol-changes-d75c9d94dc4a`. Accessed: 2024-03-13, Published:2021-09-29.

[18] D. D. E Duffield, “Dash: A payments-focused cryptocurrency.” https://github.com/dashpay/dash/wiki/Whitepaper, 2018.

[19] E. Ben Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, “Zerocash: Decentralized anonymous payments from bitcoin,” in *2014 IEEE Symposium on Security and Privacy*, pp. 459–474, IEEE, May 2014.

[20] S. Noether, “Ring signature confidential transactions for monero.” Cryptology ePrint Archive, Paper 2015/1098, 2015. `https://eprint.iacr.org/2015/1098`.

[21] V. Buterin, J. Illum, M. Nadler, F. Schär, and A. Soleimani, “U.s. treasury sanctions notorious virtual currency mixer tornado cash.” `https://home.treasury.gov/news/press-releases/jy0916`. Accessed: 2024-03-12, Published:2022-08-08.

[22] M. Yin, D. Malkhi, M. K. Reiter, G. G. Gueta, and I. Abraham, “Blockchain privacy and regulatory compliance: Towards a practical equilibrium.” `AvailableatSSRN:https://ssrn.com/abstract=4563364orhttp://dx.doi.org/10.2139/ssrn.4563364`. Published: 2023-9-6.

[23] “aptos-whitepaper_en.pdf.” `https://aptosfoundation.org/whitepaper/aptos-whitepaper_en.pdf`.

[24] “sui.pdf.” `https://docs.sui.io/paper/sui.pdf`.

[25] A. Yakovenko, “Solana: A new architecture for a high performance blockchain v0.8.13.” `https://solana.com/solana-whitepaper.pdf`. Accessed: 2024-2-4.

[26] “Polkadot-whitepaper.pdf,”

[27] “Cosmos network - internet of blockchains.” `https://v1.cosmos.network/resources/whitepaper`. Accessed: 2024-2-4.

[28] Ethereum, “What is layer 2?.” `https://ethereum.org/en/layer-2/`. Accessed: 2024-03-13.

[29] “Eip-1159.” `https://github.com/ethereum/EIPs/blob/master/EIPS/eip-1559.md`.

[30] S. King and S. Nadal, “Ppcoin: Peer-to-peer crypto-currency with proof-of-stake,” *self-published paper, August*, vol. 19, no. 1, 2012.

[31] M. Castro, B. Liskov, *et al.*, “Practical byzantine fault tolerance,” in *OsDI*, vol. 99, pp. 173–186, 1999.

[32] E. Buchman, J. Kwon, and Z. Milosevic, “The latest gossip on bft consensus,” *arXiv preprint arXiv:1807.04938*, 2018.

[33] M. Yin, D. Malkhi, M. K. Reiter, G. G. Gueta, and I. Abraham, “Hotstuff: Bft consensus in the lens of blockchain,” *arXiv preprint arXiv:1803.05069*, 2018.

[34] Y. Chen, X. Ma, C. Tang, and M. H. Au, “Pgc: Decentralized confidential payment system with auditability,” in *Computer Security – ESORICS 2020* (L. Chen, N. Li, K. Liang, and S. Schneider, eds.), (Cham), pp. 591–610, Springer International Publishing, 2020.

[35] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza, “Succinct non-interactive zero knowledge for a von neumann architecture,” in *Proceedings of the 23rd USENIX Conference on Security Symposium*, SEC’14, (USA), p. 781–796, USENIX Association, 2014.

[36] J. Groth, “On the size of pairing-based non-interactive arguments,” in *Advances in Cryptology – EUROCRYPT 2016* (M. Fischlin and J.-S. Coron, eds.), (Berlin, Heidelberg), pp. 305–326, Springer Berlin Heidelberg, 2016.

[37] S. D. Galbraith, ,Mathematics Department, University of Auckland, New Zealand, P. Wang, F. Zhang, ,College of Information Engineering, Shenzhen University, Shenzhen 518060, China, and ,School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510006, China, “Computing elliptic curve discrete logarithms with improved baby-step giant-step algorithm,” *Adv. Math. Commun.*, vol. 11, no. 3, pp. 453–469, 2017.

[38] A. Shamir, “How to share a secret,” *Commun. ACM*, vol. 22, p. 612–613, nov 1979.

[39] "Cosmos-ibc." `https://cosmos.network/ibc/`.

[40] "Polkadot-xcmp." `https://wiki.polkadot.network/docs/learn-xcm-index`.