BITLIFE: The Bitcoin Layer 2 AI Computing Power Network

2.1

The Bitlife research team

9 July 2025

Abstract

Bitcoin's transaction throughput and programmability limit its potential in decentralized AI computing applications. Existing Layer 2 solutions often introduce new trust assumptions and fail to directly integrate security with Bitcoin's consensus mechanism. This paper introduces BITLIFE—a decentralized Layer 2 network built on AI agents. The network employs a dual consensus mechanism combining Proof of Work (PoW) and the Pulse Protocol, with the R2 AI agent serving as its core component. Each agent functions as an AI computing node operating within the decentralized network, enabling the protocol's pioneering implementation of persistent verification for Layer 2 chains and verifiable state transitions on the Bitcoin network. This approach transcends mere data recording, providing enhanced security leveraging Bitcoin's Proof of Work mechanism.

Furthermore, we have deeply integrated the Pulse Protocol algorithm bridging technology with our aggregation protocol to enable secure transmission of Bitcoin assets. Finally, we developed a modular and Turing-complete execution engine—leveraging a rapid consensus mechanism to deliver sub-second-level soft finality. BITLIFE unlocks Bitcoin's vast, untapped capital potential, laying the foundation for next-generation decentralized AI computing applications and establishing the core infrastructure for the Bitcoin-based AI computing ecosystem. The key innovations of this paper include: the first proposed recursive Pulse Protocol framework for Rollup protocols, enabling continuous settlement of Bitcoin L2 state; the design of an RBTC token economic model that ensures daily price stabilization through the Pulse Protocol's growth algorithm: $r = 0.3\% + 2\% \times \sqrt{(\text{burned}/\text{produced})} + 0.5\% \times n$; and the creation of an R2 AI agent ecosystem where each agent simultaneously generates BTC and RBTC, creating a dual value capture mechanism.

BITLIFE's total RBTC token supply stands at 210 million, with 75% generated through computational mining to incentivize participants, 10% produced by the R2 AI agent, 5% allocated to ecosystem development, 7% retained by the founding team (locked for 13.2 years), and 3% dedicated to marketing. The R2 AI agent is priced between $10 and $1,000, with payment requiring a 50% USDT and 50% RBTC contribution—a design that tightly integrates the agent's economic value with the network token. The initial block reward for POW mining is 500 RBTC, with a halving interval of 21,000 blocks (approximately 145.83 days), a block generation time of 10 minutes, and a block storage capacity of 1 MB.

1 Introduction

Bitcoin [1] holds immense potential in the decentralized AI computing domain, yet its core architecture limits transaction throughput and programmability. The non-Turing completeness of Bitcoin scripts makes it difficult to implement complex logic directly on-chain, while the limited block size (approximately 1 MB every 10 minutes) further constrains the development of high-throughput applications. These technical constraints mean that despite Bitcoin's unparalleled security and decentralization, it struggles to directly support emerging applications such as AI computing networks. With the rapid advancement of artificial intelligence, decentralized AI computing networks impose higher demands on underlying blockchain infrastructure—not only requiring high-throughput transaction processing capabilities but also supporting complex computational task scheduling, measurement of computational contributions, and coordination of economic incentives among multiple participants.

Meanwhile, the Bitcoin network's valuation has surpassed one trillion US dollars, yet its capital utilization rate remains extremely low. A substantial portion of BTC assets remain idle, failing to participate in the emerging decentralized application ecosystem. If this enormous capital potential can be effectively unleashed, it will bring revolutionary changes to the entire cryptocurrency ecosystem. However, achieving this goal requires building a secure and scalable infrastructure layer that enables AI computing applications to obtain sufficient computational power and transaction throughput without compromising Bitcoin-level security.

However, existing Bitcoin scaling solutions all have inherent limitations. Sidechains relying on federated multi-signatures introduce centralized trust mechanisms, fundamentally undermining Bitcoin's security model. For instance, early sidechain projects experienced security incidents due to collusion risks among federated members, resulting in user asset losses. Meanwhile, while early Bitcoin aggregation designs could publish transaction data to the L1 chain, they lacked mechanisms to verify the validity of on-chain state transitions. This leaves these solutions vulnerable, as their security cannot rely solely on Bitcoin's consensus mechanism. Some solutions employ multi-signatures from security committees to resolve disputes, but this introduces additional trust assumptions that contradict Bitcoin's trustless nature.

Furthermore, existing Bitcoin Layer 2 solutions generally overlook the unique requirements of the AI computing power ecosystem. An AI computing power

network demands not only high-throughput transaction processing capabilities but also support for complex computational task scheduling, measurement of computing power contributions, and coordination of economic incentives among multiple participants. Current generic Layer 2 solutions lack dedicated optimization for these needs, making it difficult for AI computing applications to find suitable infrastructure support within the Bitcoin ecosystem. Particularly regarding the measurement and incentivization of computing power contributions, existing Layer 2 solutions lack native support, posing additional technical challenges for building decentralized AI computing power networks.

This raises a crucial question: Is it possible to build a Bitcoin L2 network that enables scalable AI computing while ensuring state validity is enforced by the Bitcoin mainnet itself—without requiring new trust assumptions?

This paper introduces BITLIFE—a Layer 2 network that provides a definitive solution through its Rollup architecture [7] and Pulse Protocol algorithm framework [2]. The network overcomes the limitations of Bitcoin and existing Layer 2 solutions, delivering scalable AI computing power while anchoring its security to the underlying Bitcoin blockchain. BITLIFE's uniqueness lies in its deep integration of AI computing networks with Bitcoin's security model, establishing a self-driven AI computing ecosystem through an innovative RBTC token economy and the R2 AI agent ecosystem. Our key contributions are as follows:

- We designed and formally proposed the first Rollup protocol employing a recursive Pulse Protocol algorithm framework, designed to settle continuous Layer 2 state transition statement chains on Bitcoin. The protocol ensures security by directly anchoring L2 validity to L1. Unlike existing solutions, it handles sequential state statement sequences rather than isolated individual statements—a pivotal technological breakthrough enabling full Bitcoin Layer 2 functionality.
- Collaborative Integration of Bridging and Aggregation: Leveraging the Pulse Protocol algorithm, we have designed and implemented a secure asset bridging solution. Its core innovation lies in deep integration with our aggregation protocol, ensuring both asset security and aggregation validity are safeguarded by a unified trust model, enabling seamless and secure asset transfers. This bridge supports cross-chain liquidity for various assets including BTC and RBTC.

- Innovative RBTC Economic Model: We have designed a token economy system based on the dual consensus mechanism of Proof-of-Work (POW) mining and the Pulse Protocol. The total RBTC supply is 210 million tokens, with 75% generated through computational power mining, 10% produced by the R2 AI agent, 5% allocated to ecosystem development, 7% retained by the founding team (locked for 13.2 years), and 3% dedicated to marketing. The smart contract algorithm ensures a daily price increase of 0.3%–3%, with a sustainable AI mining incentive mechanism calculated by the formula: $r = 0.3\% + 2\% \times \sqrt{(\text{burned/produced})} + 0.5\% \times n$. Here, 0.3% represents the base growth rate, $2\% \times \sqrt{(\text{burned/produced})}$ serves as the burn acceleration factor, and $0.5\% \times n$ constitutes the halving bonus, capped at 3% per day.
- R2 AI Agent Ecosystem: We have built a computing power network centered around AI agents, where each R2 AI agent functions as an AI computing node operating on a decentralized network, continuously generating dual value in BTC and RBTC. The agents are priced between $10 and $1,000, with payment processed in a 50% USDT + 50% RBTC ratio. This design tightly integrates the agents' economic value with the network token, creating a positive feedback loop between network effects and token value.
- Comprehensive system architecture implementation: We have designed a twin-subsystem model that clearly decouples the high-throughput AI computing execution layer (L2) from the L1 settlement security layer, while implementing a full performance optimization suite including a decentralized sorter, parallel execution engine, and native storage engine. This architecture supports peak loads of tens of thousands of transactions per second while maintaining sub-second soft finality confirmation times.

2 Network Architecture

BITLIFE employs a dual-layer architecture: on one hand, it utilizes the Proof of Work (POW) consensus mechanism to enable rapid block generation; on the other hand, it anchors security to the Bitcoin network through a aggregation framework. The POW layer allows AI computing nodes to quickly sort transactions and generate blocks, providing a high-throughput, EVM-compatible runtime environment, while the aggregation layer periodically submits and records the state of this L2 chain to the Bitcoin blockchain. This design positions Bitcoin as the core layer ensuring security and data availability, with the BITLIFE network serving as a scalable and efficient AI computing layer.

The adoption of this dual-layer architecture was not arbitrary. The Proof-of-Work (POW) consensus mechanism boasts the longest practical history in the blockchain sector, having been thoroughly validated for its decentralization, security, and resistance to attacks. Compared to emerging consensus mechanisms, POW's simplicity and determinism make it the ideal choice for high-security application scenarios like AI computing networks. Additionally, the aggregation framework ensures that all state transitions in the BITLIFE network are ultimately verified and settled on the Bitcoin blockchain, inheriting Bitcoin-level security guarantees. A key design principle of this architecture is modularity: each component can be independently upgraded and optimized without affecting the operation of other parts.

2.1 Network Participants and Roles

Figure 1: Network Architecture

The network is maintained by two key participants: AI computing nodes and full nodes. These participants play distinct yet complementary roles in ensuring network security and operational viability, collectively forming a secure, efficient, and decentralized network ecosystem.

AI Computing Nodes: Verification constitutes the core of the Proof of Work (POW) consensus mechanism. These nodes are responsible for generating and validating L2 blocks to ensure network security and vitality. To join the AI computing node network, candidates must meet the following requirements: Within the AI computing node pool, participants must stake RBTC tokens, with their influence in the consensus process proportional to their total staked amount—including tokens delegated by other RBTC holders. The RBTC staking mechanism not only provides economic security guarantees but also closely aligns participants 'interests with the network's long-term health. Additionally, AI computing nodes must meet minimum hardware requirements, including sufficient computing power, storage capacity, and network bandwidth, to efficiently process transactions and participate in the consensus process.

Within the AI computing node network, several specialized sub roles exist. The Aggregation Operator is a dedicated rotating role selected from a designated pool of validators. This operator is responsible for packaging L2 state transitions into batches, generating cryptographic proofs, and submitting them to Bitcoin L1 for settlement. To ensure accountability and prevent fraud, operators must stake substantial BTC as collateral on L1. The operator role rotates periodically to mitigate censorship risks and centralization tendencies.

The rotation mechanism, based on staking weights and a random selection algorithm, ensures fairness and unpredictability. During each Epoch, the protocol selects a group of operator candidates from the active validators pool and determines the final Aggregation Operator through a predefined election algorithm.

Full nodes: Full nodes store complete replicas of the BITLIFE Network blockchain, independently verifying all transactions and state transitions without relying on validators. They play a critical role in enforcing protocol rules and ensuring network transparency. The openness of the BITLIFE Network, which allows anyone to operate full nodes, embodies its decentralized ethos. While full nodes do not participate in consensus processes, their independent verification efforts provide enhanced security and transparency. Their existence ensures that even if all AI computing nodes collude maliciously, the network's state history can still be independently verified and audited.

2.2 Bilevel

BITLIFE offers a dual-tier integrity model, enabling users and applications to choose between speed and Bitcoin-level security. This design addresses the varying security and performance requirements across different use cases, providing tailored solutions for each scenario. By offering two distinct levels of determinism, BITLIFE allows developers and users to select the most appropriate security tier based on their specific business needs.

Soft finality: Once a block containing a transaction is confirmed through BITLIFE's POW consensus mechanism, the transaction achieves soft finality within sub-second latency. This delivers a rapid-response user experience, with its security ensured by the economic staking mechanism of the validator pool. For most daily transactions and AI computing power scheduling tasks, soft finality provides adequate security assurance. The economic security of soft finality stems from the following game theory analysis: Any attacker attempting to tamper with a confirmed transaction must control over 50% of the network's staked assets—a requirement that would drastically reduce their RBTC stake, rendering such attacks economically unfeasible.

Hard Finality: Hard finality represents the highest level of security assurance, achieved when the L2 state containing the transaction is settled and finally confirmed on the Bitcoin blockchain. Due to the challenge period of the optimistic aggregation mechanism, this process typically takes approximately seven days. The security of hard finality relies solely on a single honest

party to detect fraud, making its security comparable to that of Bitcoin itself. Hard finality is applicable to large-scale asset transfers and critical state change operations, scenarios requiring the highest level of protection. Once confirmed, the transaction's state change is permanently recorded on the Bitcoin blockchain and remains irreversible unless the Bitcoin network undergoes a reorganization.

In extremely rare cases where a successful L1 challenge results in a discrepancy between the L2 state and the confirmed L1 state, the protocol will terminate automatically. Subsequently, the network recovery process will be guided by the social consensus reached among all stakeholders to ensure the security of user assets. Although this emergency recovery mechanism is extreme, it serves as a final safety net for the entire system, guaranteeing that user assets remain protected even under the worst-case scenario.

3 Set the L2 state on Bitcoin

As a Layer 2 Rollup protocol, BITLIFE's security is derived from Bitcoin. This chapter details the core mechanism underpinning this relationship—the settlement mechanism. Settlement refers to the process whereby L2 state transitions executed in BITLIFE's high-throughput environment must be recorded and ultimately confirmed on the Bitcoin L1 network, enabling BITLIFE to inherit Bitcoin's security guarantees. The challenge lies in implementing this mechanism within Bitcoin's constrained, non-Turing-complete scripting environment. Our solution is a novel settlement protocol inspired by the Pulse Protocol algorithm paradigm.

3.1 Defining the Second-Level State Proposition

The core of blockchain is defined by the State Transition Function (STF), denoted as Υ. This deterministic function specifies how the network state (s) evolves. The state contains all account balances and contract data, represented by a 32-byte Merkle root. The STF takes the current state st and a batch of L2 transactions (T) to generate the next state st+1:

$$st+1 = \Upsilon(st, T)$$

Here, t denotes the index of the transaction batch. The blockchain's complete historical record originates from the initial genesis state (s0). In the context of BITLIFE, the state encompasses not only traditional account balances and contract data but also staking information of AI computing nodes,

operational status of R2 AI agents, and economic model parameters of RBTC tokens. This additional state information is critical for the proper functioning of the BITLIFE network.

The Status Declaration (Φ) is an official statement submitted by the aggregation operator to the Bitcoin L1 smart contract, designed to confirm the new L2 state generated after processing a specific batch of transactions. Serving as an anchor, it links L2 activities to L1, enabling the BITLIFE network to inherit Bitcoin's security features.

$$\Phi = \{st\text{-}1, st, T\}$$

3.2 Cryptographic primitives

This settlement agreement relies heavily on two advanced cryptographic primitives: Simple Non-Interactive Argumentations (SNARGs) and hash-based one-time signature schemes.

3.2.1 Groth16 SNARG

According to the Groth16 paper [4], the SNARG for relation R comprises three polynomial-time algorithms (Setup, Prove, and Vfy):

- $\delta \leftarrow$ SNARG SETUP (R): A setup algorithm that generates a universal reference string δ for a given relationship. This algorithm is typically executed through a multi-party computation protocol to ensure that no single participant possesses enough secret parameters to generate a fraudulent proof.
- $\pi \leftarrow$ SNARG.Prove (R, δ, Φ, ω): A proof algorithm that generates proof parameters π given a public reference string δ, an assertion Φ, and a witness ω. The proof generation process is computationally intensive and typically requires specialized hardware accelerators to enhance efficiency.
- 0/1 $\leftarrow$ SNARG.Vfy (R, δ, Φ, π): A verification algorithm designed to accept or reject proofs. This SNARG satisfies perfect completeness, computational reliability, and the complete conciseness defined by us.

Definition 1 (Complete Conciseness): A protocol (Setup, Prove, Vfy) is considered completely concise if the runtime of the AI computing node Vfy

exhibits a polynomial relationship with the security parameter λ, and the size of the proof document π also follows a polynomial relationship with λ.

The selection of Groth16 as the proof system was carefully considered. Groth16 proofs occupy only three group elements (approximately 192 bytes) and require merely a few constant-time pairing operations for verification. These characteristics make Groth16 particularly suitable for validation in Bitcoin script environments with computational constraints. Although Groth16 requires a trusted setup, the universal reference string generated through a multi-participant ceremony effectively mitigates this risk.

3.2.2 Hash-Based One-Time Signature (HOTS)

The Bitcoin Script Language and its OP_CHECKSIG opcode [6] were originally designed for verifying transaction signatures, not for validating arbitrary off-chain messages. Although proposals like BIP348 aim to extend this functionality, they require modifications to the network consensus mechanism. To overcome this limitation, we adopted a hash-based one-time signature scheme (HOTS) [5,8]. The advantage of this approach lies in the fact that the hash function is native to the Bitcoin Script and involves extremely low computational costs, enabling the HOTS scheme to operate efficiently within the constraints of the Bitcoin Script.

The improved HOTS algorithm comprises four algorithms:

- (sk, pk) ← HOTS.SETUP(λ): Generates a pair of keys (private and public keys) based on security parameters. The key generation process employs a random oracle model, ensuring the unpredictability and non-forgery of the key pair.
- s ← HOTS.publish pk, b: Releases a commitment into the Bitcoin script, enabling it to verify message signatures of length b. The commitment mechanism ensures the public key is immutable on the chain.
- w ← HOTS.SIGN(sk, m): Sign the message m using the key to generate the witness w. The signing process is deterministic; given the same key and message, it always produces the same signature.
- (0/1, m) ← HOTS_verify pk, w): Verify witness w. If valid, return 1 and display the original message m on the stack for further on-chain processing.

This final feature—the public disclosure of signed messages on the chain—is a crucial component of linked continuous state declarations. In this manner, the signed data in each state declaration can serve as input for the next declaration, naturally forming an unforgeable chain of declarations.

3.3 Overview of the Agreement

The entire settlement agreement is encapsulated within a smart contract modeled after the Pulse Protocol algorithm—a complex graph structure composed of pre-signed Bitcoin transactions rather than a single, integrated contract. Participants must collectively pre-sign this transaction graph and strictly adhere to its predefined interaction path. While the original Pulse Protocol primarily handles event-based and bridging Bitcoin-based stake settlements on external chains [3], BITLIFE's protocol is more sophisticated: it settles a continuous sequence of staking units, each representing an independent change in the L2 state, while ensuring the sequence's continuity and non-interruption.

This protocol can be regarded as a recursive structure. In Section 3.4, we first detail the sub-protocol designed to handle individual state declarations; subsequently, in Section 3.5, we explain how this single-declaration verification mechanism is recursively integrated into a broader protocol capable of processing consecutive declaration chains. By combining these two components, we have developed a comprehensive aggregation protocol for handling state transactions on the Bitcoin platform via the BITLIFE Network.

3.4 Resolution of a Single Claim

3.4.1 Pulse Protocol Algorithm 2 Paradigm

The on-chain verification of statements employs an optimism mechanism. In this scheme, the verification process is implemented through Bitcoin scripts. However, as demonstrated by the groundbreaking work of the Pulse Protocol Algorithm Consortium on the Groth16 validator, such a monolithic validator is too large to execute directly within a single Bitcoin transaction. Consequently, the Pulse Protocol Algorithm 2 paradigm [2] decomposes the extensive verification program into a series of smaller subprograms (i.e., "blocks"). The protocol then operates using a fraud-proof game mechanism: unless the challenger can identify computational errors between two specific blocks, the operator's statement is considered correct by default. This binary search approach is efficient; assuming the verification program is divided into k

subprograms, identifying the subprogram with potential errors requires at most $\log_2(k)$ rounds of interaction.

3.4.2 Protocol Roles

- 1. Certification Committee: No new entity needs to be established—the existing AI computing node pool of the BITLIFE Network serves as the Certification Committee. This committee jointly handles the pre-signature of the transaction graph defined by the protocol. Committee members are elected based on their RBTC staking weights, ensuring that participants with greater economic stakes in the protocol also bear greater security responsibilities.
- 2. Agreement Participants: Active participants in the settlement game include a designated operator (responsible for submitting claims) and an arbitrary number of supervisors. Supervisors may be any individual (including other validators), whose role is to oversee the operator and investigate fraudulent claims.

3.4.3 Single Claim Verification Agreement

Figure 2: Single Claim Verification Agreement

The protocol for verifying individual claims employs a timed challenge-response mechanism based on the Bitcoin timelock mechanism. The entire protocol consists of three main phases, each corresponding to a series of interconnected Bitcoin transactions pre-signed by the certification committee.

1. Configuration Phase: Upon project initiation, a trusted public configuration authority generates the required scripts. Using the SNARG_setup function via STF(Υ), a universal reference string δ is created. The verification procedure $f(\pi)$ = SNARG.Vfy(R, δ, Φ, π) is decomposed into a set of smaller subroutines {fi(Ii)}. The system produces three types of locking scripts: a declaration script for initiating the workflow, an assertion script for operators to submit all fi results, and a set of rebuttal scripts specific to each subroutine.

2. Claim Phase: The operator initiates the process by executing a claim transaction, which spends funds from the previous UTXO and locks its collateral (e.g., d BTC). To unlock this input, the operator must sign and publish the state claim Φt using its HOTS key. This operation triggers a timer Δ challenge. Observers can monitor the claim and locally execute STF; if the

claim is invalid, they may initiate a challenge transaction before the timeout expires.

3. Challenge and Resolution Phase: If an observer initiates a challenge transaction (which may require a smaller collateral amount of c BTC to prevent unreasonable challenges), the pessimistic path is executed. The operator must then execute the verification transaction, signing and publishing the results of all sub-programs {fi} with their HOTS key. If the operator correctly publishes all fi results and the observer fails to submit a valid rebuttal transaction in time, the operator can ultimately confirm the claim and recover their collateral by accepting the transaction.

3.4.4 Security Attributes

- Integrity: An honest operator who strictly adheres to the protocol and submits valid status declarations will never face unjust penalties, as all subroutine results published by an honest operator are guaranteed to be correct.
- Reliability: Un Honest operators making fraudulent claims cannot evade penalties, as honest observers can always construct valid counterarguments.
- Efficiency: The entire claim verification process is guaranteed to be completed within the time frame specified by the agreement timeline.

3.5 Resolution of a Series of Claims

The aforementioned agreement is sufficient to resolve individual, isolated claims. However, the settlement process requires continuous handling of a series of claims reflecting the ongoing evolution of the L2 status. This is achieved by extending the agreement to recursively link the claims into a chain.

3.5.1 Associate the claim with HOTS

The essence of chain-based claims lies in the structure of the transaction graph. Each claim transaction generates a special UTXO called a claim connector, in addition to other outputs. To submit the next claim (claim N+1), the operator must consume the claim connector UTXO generated by claim N. The locking script for this connector requires the operator to sign and publish the claim N+1 packet using its HOTS key. This design naturally links adjacent claims into a chronological, tamper-proof chain.

3.5.2 Main Transaction Diagram and Parallel Verification

Figure 3: Main Transaction Diagram and Claim Chain

This recursive structure generates a transaction graph featuring a main path that connects a series of claim requests. A key feature of this design is that submitting a subsequent claim does not require waiting for the final resolution of the previous claim verification sub-protocol. If claim N is successfully challenged, the protocol invalidates its state, thereby automatically nullifying the prerequisites for all subsequent claims.

3.5.3 Era and Reconfiguration

The reconstruction process is coordinated and executed by the L2 system contract. The designated operator prepares all necessary information for the next phase of the transaction graph, while each validator independently generates the transaction graph, signs it, and submits the signature to the L2 contract. The verification process is completed once a supermajority (Nf) of valid signatures is collected. The reconstructed transactions record the updated configuration parameters on Bitcoin.

Figure 4: Transaction Graph Reconstruction

3.6 Abstract

In essence, the BITLIFE settlement protocol establishes a continuously operational and easily manageable transaction graph modeled after the Pulse Protocol algorithm on the Bitcoin platform. This graph features a cyclic structure, composed of multiple subgraphs formed by reconfiguring transactions across Ethereum blocks. Each block subgraph contains a chronologically linked chain of state declarations, with each declaration supported by its own verification subgraph. This architecture enables BITLIFE to achieve high scalability and programmability while securely leveraging Bitcoin's unparalleled Proof-of-Work consensus mechanism.

4 State Transition Functions and Batch Execution

While Chapter Three establishes the protocol for handling Bitcoin state claims, this chapter defines the computational process claimed by the claimant: BITLIFE STF. The correct state transition for a batch of L2 blocks constitutes the fundamental unit driving the rollup mechanism.

4.1 BITLIFE Network STF

The STF of BITLIFE Network adheres to the battle-tested Ethereum EVM principles [9], providing developers with a familiar and robust development environment. As a Bitcoin aggregation technology, it enhances the EVM by incorporating additional system-level logic and dedicated contracts to address its unique requirements—such as handling bridged Bitcoin assets, processing messages from Layer 1, and managing the RBTC token economy model.

4.1.1 Gas Fee and Service Charge

Transaction fees on the BITLIFE network are paid exclusively in BTC. This design provides Bitcoin users with a seamless and consistent experience, as they can directly use their existing assets for network operations without acquiring new native tokens. BITLIFE implements a multi-dimensional gas model: execution fees cover EVM computation costs; storage fees cover L2 state modification costs; and AI computing fees cover AI computing service costs.

4.1.2 Agreement Contract

The protocol contract includes: a system configuration contract managing core protocol parameters; an AI computing node management contract defining the verifier set lifecycle; an RBTC economic model contract implementing the Pulse Protocol price growth algorithm; a Bitcoin lightweight client contract serving as an L1-L2 message transmission gateway; and a bridge contract handling cross-chain asset transfers.

The RBTC price growth formula is the core of this economic model:

$$r = 0.3\% + 2\% \times \sqrt{(\text{burned}/\text{produced})} + 0.5\% \times n$$

The formula comprises: 0.3% as the base growth rate (minimum guarantee), 2% × √(burned/produced) as the destruction acceleration factor, and 0.5% × n as the halving multiplier (where n represents the number of halving cycles). The maximum growth rate is capped at 3% per day.

4.2 Proof of Pipeline

The protocol employs a multi-stage, asynchronous, recursive proof system based on POW + Pulse Protocol AI computing nodes.

4.2.1 Achieving integrity and upgradability through CodeControlGroup

The integrity of the entire verification process relies on the core security mechanism of CodeControlGroup. The Code Commit Proof is a unique encrypted fingerprint generated by zkVM for each complete program suite; any code modification produces a distinct Code Commit Proof. CodeControlGroup is an authorization list enforced through cryptographic technology, with an index-based structure that maps each context to a valid Code Commit list.

4.2.2 Four-Stage Recursive Proof Process

Figure 5: Schematic diagram of the validation process architecture

- Phase 1 (Single-Block Verification): The off-chain simulation retrieves input data, then re-executes the state transition in zkVM to generate zero-knowledge proofs.
- Phase 2 (Batch Aggregation): Verify state continuity between adjacent blocks and record program commitments.
- Phase 3 (Batch recursion): The core recursive step, which verifies the authorization history of all executed programs and compresses the proof chain to a fixed size.
- Phase 4 (Proof Packaging): The recursive proof is encapsulated into a Groth16 proof, enabling efficient verification on the Bitcoin network.

4.3 On-chain verification via Bitcoin scripts

The on-chain verification script template serves as the ultimate arbitrator of trust relationships. Each epoch's script hard-codes a set of immutable trust anchors, including GenesisState, CodeControlRoot, and the code commit version of the non-upgradable wrapper.

5 Connect Bitcoin to the BITLIFE Network

A secure asset aggregation mechanism requires establishing a reliable asset transmission protocol between L1 and L2. This chapter provides a detailed introduction to the BITLIFE Asset Bridge.

5.1 Role

- User: The asset holder responsible for initiating transfers between Bitcoin and the BITLIFE network.

- Trader: Assists users with deposit and withdrawal operations, simplifying the complexity of the Pulse Protocol algorithm.
- Certification Committee: Uses the same validation set as the aggregation agreement and is responsible for the pre-signed transaction graph.
- Observer: An observer that can monitor protocols and detect malicious activities without requiring administrative privileges.

5.2 Cross-chain Asset Flow

Figure 6: Cross-chain Asset Flow

5.2.1 Asset Deposit (PEG-in)

- Initiate a request: The user submits a PeginRequest to all brokers, specifying the UTXO and target address.
- Preparation: The broker provides the complete Pegin transaction details and corresponding trade charts for user verification.
- Broadcasting and Mining: When users initiate transactions on the Bitcoin network, BITLIFE automatically detects and mines the corresponding assets.

5.2.2 Asset Peg-out

- Initiate combustion operation: The user initiates a Burn transaction at Layer L2, specifying the L1 withdrawal amount as defined by PSBT.
- Front-end funds for brokers: Brokers monitor Burn transactions, add input signatures, and execute them on the Bitcoin network.

5.3 Recovery of the Broker Fund

Brokers initiate the verification process through KickOff transactions, employing the same optimistic challenge-response mechanism as described in Chapter 3. During the challenge phase, the monitor verifies the legitimacy of claims, while the assertion and penalty stages require brokers to provide the Groth16 ZKP.

5.4 Escape hatch

The bridge features an emergency escape hatch, ensuring users can maintain control over their assets even when the L2 protocol fails. Users initiate

emergency withdrawals by directly broadcasting forced withdrawal transactions to Bitcoin L1, and the broker provides liquidity before transferring the funds to the user's address.

6 Safety Analysis

This chapter provides a comprehensive analysis of the underlying security mechanisms of the BITLIFE Rollup.

6.1 Security of Smart Contracts Based on the Pulse Protocol Algorithm

In smart contracts based on the Pulse Protocol algorithm, at least three roles must collaborate: the transaction graph proposer initiates the contract instance and stells BTC; the provers assume that m out of n provers are honest, with the remainder being semi-honest; and the monitor oversees the on-chain state and can initiate challenges upon detecting anomalies. Security objectives include validity (every transaction must be valid after pre-signature), integrity (no new transactions can be added to the transaction graph), and flexibility (security assumptions can be adjusted according to specific scenarios).

6.2 Bitcoin Settlement Guarantee Mechanism

This section focuses on demonstrating the security features of Bitcoin settlement:

Theorem 5 (Completeness): An honest operator that strictly adheres to the protocol and submits valid state declarations will never face unfair penalties.

Proof. An honest operator will publish valid statements and subroutine results within the specified time window, ensuring no inconsistencies occur. Consequently, no observer can execute a rebuttal script, and the operator will not be penalized.

Theorem 6 (Reliability): Dishonest operators submitting fraudulent claims cannot evade punishment, as honest observers can always construct valid counter-transactions.

Proof: If an dishonest operator fails to publish Φ within the Δ-declaration time frame, they will be penalized. Using the proof-of-concept

algorithm based on binary search, an observer can inevitably select a subprocedure for challenge.

Theorem 7 (Efficiency): The entire statement verification process is completed within the time limit defined by the protocol time lock.

Proof. Each phase of the protocol has a limited time constraint. The maximum confirmation time is Δstatement + max{Δchallenge, Δassertion + Δ rejection}.

Figure 7: Security of Bitcoin Settlement

6.3 Resistance to the Review System

Unlike traditional L2 architectures that rely on a single sorter, BITLIFE employs a validator polling mechanism for block generation. This decentralized sorting approach ensures that no single entity can unilaterally review transactions.

7 System Architecture

A robust system architecture is crucial for achieving BITLIFE's dual objectives—high performance and security with minimized trust. The core of our design lies in adopting a twin-system model.

7.1 Decoupling of L2 Execution from L1 Settlement

- AI Computing Node Subsystem (Performance Domain): A high-performance blockchain that processes L2 transactions and delivers sub-second soft finality.
- Rollup subsystem (Security Domain): Anchors the state of the AI computing power node subsystem within the Bitcoin network.

Figure 8: Overview of the Twin System Architecture

7.2 Validation Subsystem

The validation subsystem is specifically designed for performance optimization and consists of three core components: a decentralized sorter (operating on the POW protocol), a parallel execution engine (overcoming the sequential execution limitations of EVM), and a blockchain-native storage engine (a high-performance persistence layer). These components are integrated

through an active computation pipeline, collectively forming the core drivers of BITLIFE's transaction throughput and low latency.

7.3 Aggregation Subsystem

The cryptographic foundation of BITLIFE's security lies in its recursive proof system—the Asynchronous ZKP Generator—which generates concise and irrefutable validity proofs for all state transitions. The L1 termination protocol is implemented around three core software entities: Operators (Transaction Graph Generator, Batch Committer, Challenge responder), Monitors (Proof AI Computing Node, Challenger), and AI Computing Node (Transaction Graph Validator, MuSig signer).

7.4 Transaction Life Cycle

Figure 9: Detailed Description of the Transaction Lifecycle

The transaction lifecycle comprises: Process (user initiates L2 transaction), Block Consensus (validators agree on sequencing), Execution (parallel processing of transactions to compute new states), Persistence (state writes to storage achieving soft finality), Subsystem Handshake (asynchronous state data transfer to the aggregation subsystem), State Declaration (operators aggregate batches for Bitcoin settlement), and Challenge Response (generation of ZKP proofs when observers raise challenges).

7.5 Conclusion

Ultimately, the BITLIFE architecture provides a clear blueprint for Bitcoin's scalable development: it positions Bitcoin as the ultimate decentralized trust and settlement layer, while the BITLIFE Network serves as the high-throughput, verifiable computing layer built upon it.

8 Limitations and Future Development Directions

8.1 Limitations

- Dependence on the integrity of the AI computing node set: The current bridge and settlement protocols rely on the principle of honest majority within the active AI computing node set for their security. Although this security is ensured through cryptographic measures, it constitutes a trust assumption that extends beyond Bitcoin's own proof-of-work mechanism.

- Centralized Operator and Activity Assurance: The current model employs a single polling-based aggregation operator responsible for transaction sequencing and settlement. While highly efficient, the offline status of operators may become a potential failure point affecting system activity.
- Dependence on broker liquidity: The rapid withdrawal and emergency exit mechanisms rely on an active network of third-party brokers to provide initial liquidity.

8.2 Future Directions

- Leverage future Bitcoin protocol upgrades: The upcoming protocol enhancements (such as the introduction of contract operation codes like OP_CTV and OP(cat)) are expected to directly enable more trustless smart contract capabilities on Bitcoin.
- Advanced Proof System: Continuously evaluates and integrates the latest advancements in zero-knowledge proof systems and other cryptographic technologies.
- Enhanced RBTC economic model: Optimize price growth algorithm parameters and explore additional token application scenarios.

9 Conclusion

This paper introduces BITLIFE—a scalable and EVM-compatible AI computing layer designed for Bitcoin, whose security is based on the honest majority principle. Built on the Pulse Protocol architecture, BITLIFE supports both complex general computing tasks and AI computing scheduling while directly anchoring its security to the Bitcoin network. Our core contribution lies in proposing an innovative recursive settlement protocol—the first to enable continuous and verifiable settlement of Bitcoin Layer 2 state transitions.

This protocol, combined with a collaborative asset bridging mechanism employing the same security model and an execution layer fully compatible with EVM, collectively establishes a comprehensive and practical decentralized AI computing power application platform. Meanwhile, our designed RBTC economic model achieves sustainable token value growth through the innovative Pulse Protocol pricing algorithm, providing long-term economic incentives for network

participants. As the core product of the ecosystem, the R2 AI agent offers a standardized access mechanism for the decentralized AI computing power network.

We consider BITLIFE to be the pivotal foundation for building the top-tier infrastructure of the BTCFi and AI computing ecosystem. By establishing a clear architecture—with Bitcoin serving as the final settlement layer and BITLIFE functioning as an efficient, verifiable AI computing layer—our research provides a practical solution to unlock Bitcoin's immense potential. We hope this design philosophy, centered on reducing transaction costs and enhancing censorship resistance, will drive the development of more scalable and secure Bitcoin-based applications.

Appendix A: Source Code of the Pulse Protocol Smart Contract

This appendix provides the complete Solidity source code for the Pulse Protocol smart contract. The contract implements the RBTC price growth algorithm, with the formula: $r = 0.3\% + 2\% \times \sqrt{(burned/produced)} + 0.5\% \times n$, and a maximum growth rate cap of 3% per day. All calculations are performed with 1e18 precision using the Babylonian method for square root computation.

A.1 Contract Statement and Constant Definitions

```
// SPDX-License-Identifier: MIT

pragma solidity ^0.8.20;


/**

* @title PulseProtocol

* @author RBTC Team

* @notice Pulse Protocol - RBTC Price Growth Algorithm

* @dev Formula: r = 0.3% + 2% x sqrt(burned/produced) + 0.5% x n

*      - 0.3%: Base growth rate (minimum guaranteed)

*      - 2% x sqrt(burned/produced): Burn acceleration factor

*      - 0.5% x n: Halving bonus (n = halving count)

*      - Maximum rate: 3% per day

 */
```

```
contract PulseProtocol {

// ============ Precision Constants ============

uint256 public constant PRECISION = 1e18;

uint256 public constant BASIS_POINTS = 10000;  // 10000 = 100%


// ============ Formula Parameters (basis points) ============

uint256 public constant BASE_RATE_BP = 30;       // 0.3% = 30 bps

uint256 public constant BURN_FACTOR_BP = 200;    // 2% = 200 bps

uint256 public constant HALVING_BONUS_BP = 50;   // 0.5% = 50 bps

uint256 public constant MAX_RATE_BP = 300;       // 3% = 300 bps


// ============ Halving Parameters ============

uint256 public constant HALVING_INTERVAL = 21000;   // Halving period

uint256 public constant BLOCKS_PER_DAY = 144;       // Daily blocks

uint256 public constant INITIAL_BLOCK_REWARD = 500 * 1e18;
```

A.2 State Variables and Events

```
// ============ State Variables ============

address public owner;

uint256 public currentPrice;        // Current price (USDT, 1e18 precision)        uint256 public currentPrice;        // Current price (USDT, 1e18 precision)

uint256 public totalBurned;         // Total burned (RBTC)         uint256 public totalBurned;         // Total burned (RBTC)

uint256 public totalProduced;       // Total produced (RBTC)

uint256 public currentBlock;        // Current block height        uint256 public currentBlock;        // Current block height

uint256 public lastUpdateDay;       // Last update day

uint256 public genesisTimestamp;    // Genesis timestamp

uint256 public lastRewardBlock;     // Last reward block

uint256 public totalMined;          // Total POW mined          uint256 public totalMined;          // Total POW mined


// Price history
```

```
struct PriceRecord {
        uint256 day;
        uint256 price;
        uint256 rate;
        uint256 burned;
        uint256 produced;
        uint256 halvingCount;
    }
mapping(uint256 => PriceRecord) public priceHistory;
uint256 public totalDays;

// ============ Events ============
event OwnershipTransferred(address indexed previous, address indexed newOwner);
event PriceUpdated(uint256 indexed day, uint256 oldPrice, uint256 newPrice, uint256 rate);
event Burned(address indexed from, uint256 amount, uint256 totalBurned);
event Produced(uint256 amount, uint256 totalProduced);
event HalvingOccurred(uint256 halvingCount, uint256 blockNumber);
```

A.3 Core Computing Functions

```
// ============ Core Calculation Functions ============

    /**
* @notice Calculate current halving count
* @dev n = floor(currentBlock / HALVING_INTERVAL)
     */
function getHalvingCount() public view returns (uint256) {
        return currentBlock / HALVING_INTERVAL;
    }

    /**
```

```
* @notice Calculate current block reward
* @dev Halving every 21,000 blocks, starting from 500 RBTC
    */
function getCurrentBlockReward() public view returns (uint256) {
        uint256 halvings = getHalvingCount();
        if (halvings >= 26) return 0;  // Near zero after 26 halvings
        return INITIAL_BLOCK_REWARD >> halvings;
    }

    /**
* @notice Calculate price growth rate r
* @dev Formula: r = 0.3% + 2% x sqrt(burned/produced) + 0.5% x n
* @return rateBps Growth rate (basis points, 100 = 1%)
    */
function calculateGrowthRate() public view returns (uint256 rateBps) {
        // Base rate 0.3%
        uint256 baseRate = BASE_RATE_BP;

        // Burn acceleration factor
        uint256 burnFactor = 0;
        if (totalProduced > 0) {
            uint256 burnRatio = (totalBurned * PRECISION) / totalProduced;
            uint256 sqrtRatio = sqrt(burnRatio);
            burnFactor = (BURN_FACTOR_BP * sqrtRatio) / PRECISION;
        }

        // Halving bonus: 0.5% x n
        uint256 halvingCount = getHalvingCount();
        uint256 halvingBonus = HALVING_BONUS_BP * halvingCount;
```

```
        // Total rate = base + burn factor + halving bonus
        uint256 totalRate = baseRate + burnFactor + halvingBonus;

        // Cap at maximum rate 3%
        if (totalRate > MAX_RATE_BP) {
            totalRate = MAX_RATE_BP;
        }

        return totalRate;
    }
```

A.4 Daily price updates

```
    /**
* @notice Calculate new price
* @return newPrice New price (1e18 precision)
     */
function calculateNewPrice() public view returns (uint256) {
        uint256 rateBps = calculateGrowthRate();
        return (currentPrice * (BASIS_POINTS + rateBps)) / BASIS_POINTS;
    }

    /**
* @notice Daily price update
* @dev Can only be called on a new trading day
     */
function updateDailyPrice() external returns (uint256) {
        uint256 dayElapsed = (block.timestamp - genesisTimestamp) / 1 days;

        if (dayElapsed <= lastUpdateDay) {
            revert NotTimeToUpdate();
```

```
        }

        uint256 oldPrice = currentPrice;
        uint256 rateBps = calculateGrowthRate();

        // Update price
        currentPrice = (currentPrice * (BASIS_POINTS + rateBps)) / BASIS_POINTS;

        // Record history
        totalDays++;
        PriceRecord storage record = priceHistory[totalDays];
        record.day = totalDays;
        record.price = currentPrice;
        record.rate = rateBps;
        record.burned = totalBurned;
        record.produced = totalProduced;
        record.halvingCount = getHalvingCount();

        lastUpdateDay = dayElapsed;

        emit PriceUpdated(totalDays, oldPrice, currentPrice, rateBps);

        return currentPrice;
    }
```

A.5 Square root calculation (Babylonian method)

```
    /**
* @notice Square root calculation (Babylonian method)
* @param x Input value
* @return y Square root
```

```
    */
function sqrt(uint256 x) public pure returns (uint256) {
        if (x == 0) return 0;

        uint256 xx = x;
        uint256 r = 1;

        if (xx >= 0x100000000000000000000000000000000) { xx >>= 128; r <<= 64; }
        if (xx >= 0x10000000000000000) { xx >>= 64; r <<= 32; }
        if (xx >= 0x100000000) { xx >>= 32; r <<= 16; }
        if (xx >= 0x10000) { xx >>= 16; r <<= 8; }
        if (xx >= 0x100) { xx >>= 8; r <<= 4; }
        if (xx >= 0x10) { xx >>= 4; r <<= 2; }
        if (xx >= 0x8) { r <<= 1; }

        r = (r + x / r) >> 1;
        r = (r + x / r) >> 1;
        r = (r + x / r) >> 1;
        r = (r + x / r) >> 1;
        r = (r + x / r) >> 1;
        r = (r + x / r) >> 1;
        r = (r + x / r) >> 1;

        uint256 r1 = x / r;
        return (r < r1 ? r : r1);
    }
}
```

The aforementioned smart contract fully implements the Pulse Protocol price growth algorithm. Upon deployment, it requires an initial price input (0.01 USDT, equivalent to 1e16), with daily price updates achieved through the updateDailyPrice() function. volume and output volume are recorded using the

recordBurn() and recordProduction() functions, respectively. The contract also offers features such as price forecasting and historical record queries; detailed implementation is available in the complete source code.

References

[1] S. Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System, 2009. http://bitcoin.org(bitcoin.pdf

[2] Robin Linus. BitVM: Performing any computation on Bitcoin, December 2023. https://bitvm.org(bitvm.pdf

[3] Robin Linus, Lukas Aumayr, Alexei Zamyatin, Andrea Pelosi, Zeta Avarikioti, Matteo Maffei. BitVM2: Connecting Bitcoin with a Layer 2 Network. https://bitvm.org(bitvm_bridge.pdf

[4] Bitcoin Wiki. Script, 2025。https://en.bitcoin.it/wiki/Script

[5] Kalodner, H., Goldfeder, S., Chen, X., Weinberg, S. M., & Felten, E. W. (2018). Arbitrum: A Scalable Private Smart Contract. At the 27th USENIX Security Symposium.

[6] Buchmann, J., Dahmen, E., Ereth, S., Hulsing, A., & Ruckert, M. On the Security of the Winternitz One-Time Signature Scheme, 2011.

[7] Wood, G. (2014). Ethereum: A secure, decentralized, universal transaction ledger.

[8] Rubin, J. (2020). BIP-0119：checktemplateverify。