Yala: A Bitcoin-Based Asset Protocol

Yala Labs

November 26, 2024

ABSTRACT

The Yala Protocol introduces a new method to unlock the programmability of BTC assets, focusing on developing the DeFi layer within the Bitcoin ecosystem. It addresses challenges like Bitcoin's limited scripting language and scalability issues. Yala's architecture includes layers for application, consensus, data availability, execution, and settlement, enabling DeFi transactions while preserving Bitcoin's security. The protocol integrates over-collateralized stablecoin mechanisms, insurance derivatives, and atomic swaps for seamless interaction across blockchain systems, expanding Bitcoin's potential beyond digital currency use to DeFi and complex applications.

Key words. BTC, DeFi, Stablecoin, Atomicswap, Insurance, Data Availability, Custodial mapping, Custody

1. Introduction

Bitcoin has emerged as the eighth-largest asset globally, with its valuation and security mechanisms universally recognized. Despite its acclaim, Bitcoin's utilization has predominantly been as a primary digital currency. Before Ethereum's advent, numerous endeavors aimed to extend Bitcoin's utility beyond mere cryptocurrency applications, albeit these were largely experimental due to Bitcoin's intrinsic limitations.

The advent of Ethereum, with its Turing-complete smart contracts, marked a significant shift in the blockchain ecosystem. Ethereum's smart contract functionality enabled the development of decentralized applications (dApps) that go beyond simple value transfer, encompassing a wide range of use cases such as decentralized finance (DeFi), non-fungible tokens (NFTs), and decentralized autonomous organizations (DAOs). This expanded programmability facilitated the implementation of complex logic and state transitions on the blockchain, in contrast to Bitcoin's more limited scripting capabilities.

As a result, most of the subsequent complex logic implementations after Ethereum's launch took place on the Ethereum network and other newer blockchains. The approval of Bitcoin exchange-traded funds (ETFs) underscored a pivotal moment, heralding traditional finance's acknowledgment of cryptocurrency value. This necessitated Bitcoin to introduce compatible financial products to enhance its appeal and broaden its user base.

As of now (from DeFiLlama[1]), Bitcoin's Total Locked Value (TVL) is a mere $2.8 billion, in the contrast to its market capitalization of $1.8 trillion, and significantly dwarfed by the $97 billion TVL across various blockchain networks. Despite holding a 51.9% market share, Bitcoin's representation in the DeFi TVL is relatively small. This scenario underscores the vast potential for DeFi within the Bitcoin ecosystem, signifying a crucial trajectory for Bitcoin's evolution.

DeFi functionalities on Bitcoin primarily utilize the Lightning Network, with the RSMC (Revocable Sequence Maturity Contract) and HTLC (Hashed Timelock Contract) protocols facilitating peer-to-peer transactions and off-chain transaction executions on Bitcoin. The challenges in deploying complex logic applications on Bitcoin are primarily attributed to its scripting language, characterized by a lack of Turing completeness and scalability issues.

Ethereum's whitepaper delineated several Bitcoin scripting deficiencies, including Turing incompleteness, value-blindness, statelessness, and blockchain unawareness. These limitations are anticipated to be mitigated through various Bitcoin protocol enhancements, such as introducing the `OP_RETURN` opcode in 2014, enabling the incorporation of off-chain states. Subsequent upgrades like SegWit and Taproot have imparted a degree of Turing completeness to Bitcoin. These enhancements, coupled with an off-chain execution layer interacting with scripted contracts in the Witness component, can refine Bitcoin's computational capabilities, encompassing most of Bitcoin's Layer 2 solutions.

This article introduces the solutions proposed by Yala Finance for implementing complex transactions based on BTC assets. We analyze existing BTC Layer 2 solutions and the issues they face. The Yala Protocol presents a novel approach to unlock the programmability of BTC assets, focusing on developing a DeFi layer within the Bitcoin ecosystem. The protocol aims to overcome the challenges of conducting complex transactions using Bitcoin's native assets, such as its limited scripting language and scalability issues. Yala's architecture includes an Application Layer, Consensus & Data Availability Layer, Execution Layer, and Settlement Layer, enabling native DeFi transactions for BTC assets while maintaining the security and consensus of the Bitcoin network. The Yala Protocol considers existing feasible solutions for complex transactions involving BTC assets, mitigating risks in cross-chain asset mapping through key management, decentralized asset custody, and one-time signatures, making the cost of potential attacks outweigh the benefits. The Yala Finance system features essential components such as Vaults, an automatic stabilizer, and an insurance module, providing a comprehensive DeFi ecosystem for BTC assets.

This article also designs a solution for completing the entire DeFi process on the BTC main chain, implementing a simple state machine on BTC that requires other blockchains to assist in the transaction state transition without affecting consensus on the BTC main chain. The innovative approach of the Yala Protocol aims to unlock the full potential of the Bitcoin blockchain,

[1] `https://defillama.com/chain/Bitcoin`

extending its utility beyond its primary use as a digital currency and enabling the integration of DeFi and other complex applications within the BTC ecosystem.

2. Background

2.1. Challenges for running DeFi on Bitcoin

BTC requires a scalable solution to serve as the infrastructure for DeFi and even more ecosystem applications. We first consider what issues a blockchain must solve to run DeFi applications:

Security: The blockchain must ensure robust security to prevent issues such as double-spending, unauthorized modifications to transaction records, and other malicious activities. High TVL also reinforces trust in DeFi platforms.

Programmability: Critical for DeFi, the blockchain should support complex financial transactions and protocols, ensuring reliable and accessible outcomes from contract executions [6].

High throughput and low transaction costs: DeFi applications demand high transaction throughput and low costs, necessitating a scalable and efficient infrastructure.

Governance mechanism: A robust governance framework is crucial for encouraging community involvement, consensus, and managing updates to the network.

Risk control: The financial risks in DeFi exceed those in traditional trading, requiring dependable price data sources and risk mitigation strategies.

Given these prerequisites, Bitcoin's original design seems less suitable for the complexities of DeFi. However, potential solutions include:

1. Bitcoin's capability to support advanced smart contract logic has improved with recent upgrades. Using Merkleized Abstract Syntax Trees (MAST), DeFi logic can be encapsulated into scripts within the Witness component, intended for off-chain node execution rather than on the main chain.
2. Capitalizing on Bitcoin's unmatched security, DeFi transactions could be streamlined to Unspent Transaction Output (UTXO) transactions against Pay to Taproot Hash (P2SH) addresses, ensuring secure settlements for DeFi contracts. Incorporating yield farming concepts could encourage Bitcoin users to contribute their assets, enhancing TVL.
3. Considering the high transaction fees and prolonged block times on the Bitcoin network, limiting throughput, a significant portion of DeFi transactions should be processed off-chain. Bitcoin would then finalize the settlements, ensuring consensus and security.
4. Implementing governance mechanisms for DeFi community members is crucial for protocol governance, network upgrades, and parameter settings within the Bitcoin ecosystem.
5. Bitcoin's volatility, accentuated by its 10-minute block intervals, increases risk in its DeFi ecosystem compared to other chains, necessitating reliable oracles for price feeds and effective risk management measures.
6. Introducing a Bitcoin-native stablecoin could significantly stabilize liquidity in DeFi applications, serving various roles, including lending, hedging, and settlements.

2.2. Existing solutions

There are various proposed solutions for expanding Bitcoin to support complex applications such as DeFi, primarily Bitcoin Layer 2 solutions taking the form of rollups or EVM-compatible sidechains with similar underlying approaches. Rollup solutions generate fraud or validity proofs for off-chain transaction execution, solutions that are then validated by publishing the proofs to scripts within Bitcoin's Taproot witness data. At the same time, the EVM sidechain approach involves launching a new EVM-compatible blockchain network that interacts with the Bitcoin mainchain cross-chain. However, practical roll-up implementations face challenges, including constraints on proof complexity due to limited witness data size. Additionally, posting data on Bitcoin's blockchain does not automatically ensure the authenticity of off-chain roll-up transactions. As a result, most current Bitcoin rollup projects opt for a "sovereign rollup" or "client validation model" where validators synchronize and validate the entire rollup state data off-chain, failing to leverage Bitcoin's consensus and security model, exemplified by the client-validated RGB protocol.

A practical rollup strategy involves leveraging the BTC network to validate transaction commitments. It's crucial also to have mechanisms that ensure assets within the rollup can be safely accessed or recovered during unexpected incidents or system breakdowns, even when rollup nodes/sorters are unavailable or decline transactions. A secure emergency channel allows for the retrieval of assets. BitVM, integrating fraud proofs with Taproot's MAST scripts, is a notable solution many rollup projects adopt. However, BitVM's major drawback is its absence of an emergency asset release feature, posing a risk of asset loss in the event of attacks or failures.

Besides the direct Bitcoin expansion BTC L2 solutions, DLC.link proposed a different solution, mapping BTC to Ethereum L2 based on the DLC protocol, to participate in the ETH DeFi ecosystem as dlcBTC. This only provides cross-chain interoperability for Bitcoin, relying on the security and consensus of other chains, not exploiting the characteristics of the BTC ecosystem. However, the use of Discreet Log Contracts (DLCs) is a very valuable BTC contract primitive, essentially a condition judgment between transaction parties, determining the final direction of funds based on an agreement by two or more parties on the outcome of a specific event chosen by one or multiple oracles.

We believe native DeFi applications and other ecosystem applications for Bitcoin should be built directly utilizing Bitcoin's native asset types, including BTC itself, Atomical tokens, Taproot Assets, Omni Layer tokens, and more, with data structures encoded either in `OP_RETURN` outputs or through `OP_FALSE OP_IF...OP_ENDIF` script constructs within the witness data exhibiting similar formats. These diverse native asset encodings can be unified under the "inscriptions" model enabled by the Ordinals protocol, providing a standardized format for associating arbitrary data with satoshis and storing this content off-chain in Taproot script-path spend scripts. Building upon inscriptions allows layering more complex business logic onto off-chain Prover servers, such as introducing new operational primitives like "list", "collateralize", "destroy", "authorize" alongside "mint", "deploy", and "transfer" under the inscription JSON "op" field, with combinations evolving into higher-level functionalities like swaps, lending, inscribed financial instruments ("InscriptionFi"), and even complex decentralized applications spanning social networks and gaming ("SocialFi" and "GameFi") - thereby standardizing Bitcoin's native assets into a richly extensible framework for native DeFi and broader applications deeply integrated with Bitcoin itself.

The Yala Protocol is a proposed solution that addresses the challenges of implementing complex transactions using Bitcoin's native assets. Yala draws on the concept of decoupling to apply it to the Bitcoin ecosystem, leveraging the security of the

BTC network and the Turing-completeness of other blockchains. Yala's native operability aims to realize the programmability of BTC assets, and the protocol plans to first implement a DeFi layer on Bitcoin based on this theoretical architecture.

2.3. Native operability and interoperability on Bitcoin

We evaluated the limitations and reference technologies of current solutions. Taproot's MAST enables complex transaction logic on Bitcoin, while DLCs streamline transaction processes by moving most operations off-chain, ensuring both security and privacy. Leveraging these technologies, transaction contracts are stored similarly to Ordinals, facilitating direct on-chain storage and off-chain execution by nodes. This approach allows contracts to be executed like Ordinals, blending efficiency with blockchain integrity.

This approach maintains the intrinsic qualities of Bitcoin assets in DeFi, addressing the need for high throughput and cost efficiency. In DeFi's multifaceted transaction processes, BTC doesn't have to be involved in every step. The primary value of the BTC main chain in executing contracts lies in its security and consensus capabilities. Therefore, we should preserve these aspects while separating all other system functions from BTC. A modular approach allows us to extend Layer1 functionalities, enhancing main chain scalability. By decoupling system functions and strategically distributing tasks across modules, we significantly boost system throughput and concurrently reduce transaction costs.

Blockchain interoperability allows for the sharing of information and value across different blockchain networks, facilitating direct interaction between users and developers with multiple platforms. The aim of achieving Bitcoin interoperability is to enable those holding BTC to leverage their assets in transactions on other blockchains without selling. These methods include cross-chain bridges, atomic swaps, and relay chains. A key challenge is conducting transactions with BTC assets on other chains without significantly burdening BTC's resources. Thus, mapping BTC assets for use on trusted alternative blockchains presents a viable strategy for providing BTC interoperability, though it's not without its challenges.

Most Bitcoin Layer 2 solutions utilize cross-chain bridges, where security hinges on creating cryptographic proofs off-chain that are later verified on the Bitcoin blockchain. This approach, however, runs into considerable difficulties, especially because the Bitcoin script language, designed for simplicity and security, isn't well-suited for complex verifications. This limitation raises concerns about the practicality of efficiently executing these advanced verification processes within Bitcoin's existing framework. Furthermore, coordinating emergency risk management across different blockchains adds layers of complexity to these systems. L2 projects are exploring the reactivation of the `OP_CAT` operator to enhance data interaction capabilities, but this move could inadvertently increase the risk of Distributed Denial of Service (DDoS) attacks, adding another dimension of concern.

Another approach is Drivechain, whose idea was detailed in two proposals: BIP 300, involving a system called Hashrate Escrows, and BIP 301, related to Blind Merged Mining. The Drivechain proposal aimed to let developers customize sidechains and use Bitcoin locked on the main blockchain to manage assets on these sidechains. However, neither BIP 300 nor BIP 301 received the necessary implementation approval. As a result, the Drivechain approach is generally regarded as unreliable.

Atomic swaps facilitate direct cryptocurrency exchanges between two parties on the Bitcoin network, leveraging its script language without needing a central authority or intermediary. This process uses special types of scripts, such as those found in Hashed Timelock Contracts (HTLCs) within the Lightning Network. These scripts ensure that transactions can only be completed if certain conditions are met, creating a secure and trustless exchange mechanism. While atomic swaps offer a decentralized way to trade cryptocurrencies, they face challenges like the "replacement cycle attack," which questions the security of the Lightning Network. Despite these concerns, atomic swaps are a significant advancement for peer-to-peer transactions.

2.4. Bitcoin's State Transfer

In December 2023, Robin Linus, the head of the ZeroSync project, published a whitepaper titled "BitVM: Compute Anything On Bitcoin", sparking thoughts on enhancing Bitcoin's programmability. The paper proposes a solution to achieve Turing-complete Bitcoin contracts without altering the Bitcoin network's consensus, allowing any complex computation to be verified on Bitcoin. This solution uses Lamport signatures (i.e., bit commitment) to transfer the state between UTXOs. Lamport signatures can validate complex transaction processes and guarantee that the actual signed content cannot be tampered with under any attack. However, in the BitVM solution, the standard Lamport signature size is 8192 bytes, and with additional required metadata, the signature size exceeds 8860 bytes, while the witness stack size is only 2121 bytes. Therefore, BitVM heavily relies on OP_CAT to link Lamport signature data, but the use of OP_CAT makes BTC vulnerable to DDoS attacks, and OP_CAT has not been considered for reactivation.

Lamport signatures are a type of one-time signature, where generating a signature for each message requires generating a new pair of private and public keys. The private key consists of 256x2 random u256 values arranged in a two-dimensional matrix. The public key is the hash of each u256 in the private key, forming a two-dimensional matrix. The hash function can be any cryptographically secure hash function. Lamport signatures indeed provide the effect of mapping a public key to a specific, tamper-proof value, making it resistant to ambiguity. Based on this characteristic, Lamport signatures can enable global "key-value" pair storage on Bitcoin. This key-value pair storage can transfer data between script programs, splitting large computations into many independent parts or decoupling complex transaction processes into multiple independent steps. Ethan Heilman discussed a scheme to enhance the security of ECDSA signatures by using signature length in the Bitcoin-Dev mailing list. Robin Linus also discussed the rationale for using the signature length as proof of work in Bitcoin Script in "Proof of Work in Bitcoin Script". This scheme combines ECDSA with Lamport signatures to commit to the signature size, and the signature length effectively serves as a proxy for the transaction hash value of the spending transaction. Repeating this process multiple times can provide a certain level of cryptographic security. The flaw in this scheme is that it requires a very high number of signatures for a single transaction to ensure high security. We can enhance security by using a small number of repeated signatures and unrelated random contents in the signature.

3. Yala architecture

The architecture of Yala Finance is composed of UTXO operations on the BTC main chain, asset management on the target

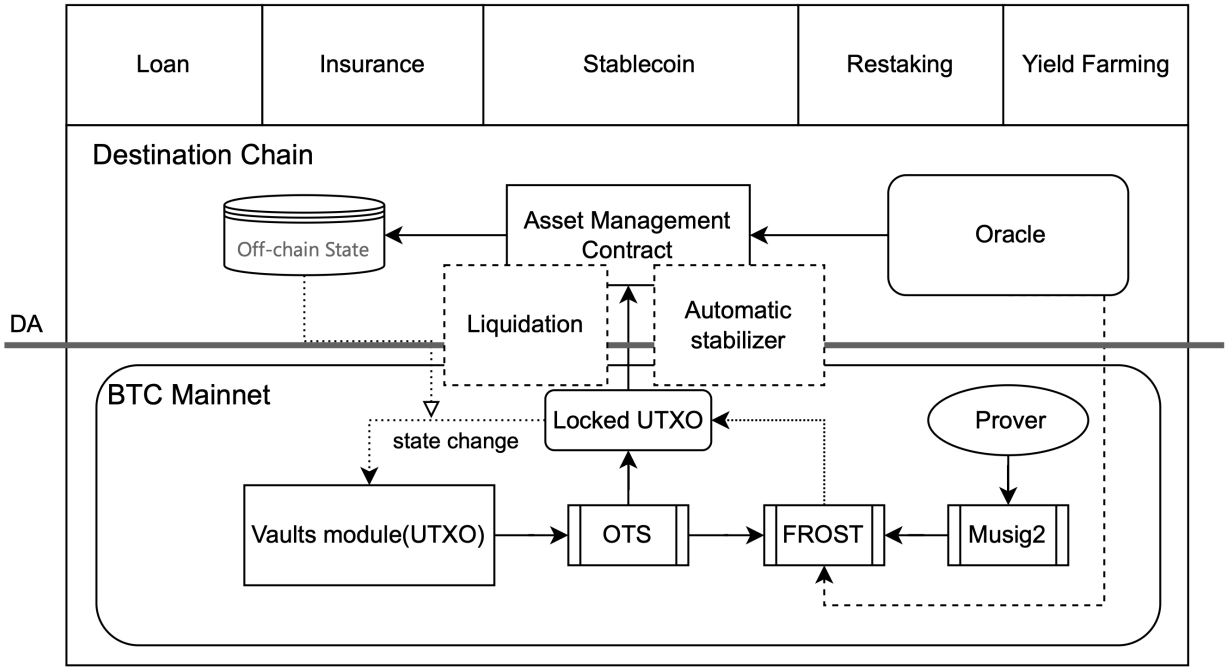

Fig. 1. Yala architecture

chain, and a data availability layer. Yala's entire architecture design revolves around the UTXO state machine model. We considered two methods for handling script states on the BTC chain: 1) generating states directly on the BTC main chain using one-time signatures and 2) mapping BTC assets onto other chains to manage assets, with final settlement occurring on the main chain. Yala Finance previously employed a foundational data structure based on Ordinals, where transaction execution scripts require transaction requests initiated by participants at the Prover application layer. The off-chain state changes occurring during transaction execution are then managed and maintained in a global state by the Prover. This structure stores UTXO states in SegWit, and state changes are executed in a relatively centralized manner. We have abandoned this state management method but will still support SegWit-based BTC assets in the new architecture.

For the operation of UTXO states on the BTC main chain, we discussed Andrew Poelstra's proposal–"Script State From Lamport Signatures[9]". We incorporated Ethan Heilman's approach[4] to eliminate dependency on OP_CAT in the signature verification process. This approach uses one-time signatures to facilitate state transitions within BTC scripts, while multi-signature arrangements manage interactions between lenders, pools, Provers, and Oracle networks. The liquidation process still requires coordination with other chains to adjust the transfer addresses of funds and update UTXO states. With this setup, we have designed vaults directly on the BTC main chain; however, this method remains experimental. Since BTC scripts lack introspection capabilities for transaction data, complex state changes within BTC transactions rely on smart contracts from other chains. In the future, storing encoded data in the Witness may be possible to allow BTC scripts to track transaction progress independently.

The second method leverages asset-mapping. Using conditional scripts, BTC assets are locked in UTXOs on the main chain, while equivalent assets are mapped onto contracts on the target chain. This allows the receiving contract on the target chain to perform complex financial operations directly, enabling users to trade with the mapped assets on that chain. The mapping process can utilize multisig cross-chain bridges, atomic swaps, and Discreet Log Contracts (DLCs). We have designed a practical atomic swap mechanism, which has already been deployed. However, security for multi-sig cross-chain bridges requires special attention.

3.1. Execution processing

Yala Finance divides the Execution Layer into two parts: the BTC mainnet and the target chain. The UTXO is the core state storage and the final state transition environment. Multisignature locking scripts determine the unlocking conditions for UTXOs. Participants engage in the Yala Finance transaction process through the Vaults Module. State generation and changes are managed through one-time signatures during users' participation. The unlocking process requires both a liquidation mechanism and the involvement of an automated stabilizer. The target chain's Asset Management Contract (AMC) is the state storage and relay environment for complex transactions. Nearly all state changes in mapped transactions occur within the AMC, which also assists in state changes for OTS-based UTXO scripts.

Oracle module: Vaults require real-time market price acquisition of collateral assets to determine when conditional executions are triggered. The Yala Foundation is responsible for maintaining the Oracle Module and Oracle Security Module (OSM). The Oracle Module obtains price inputs from off-chain sources. Off-chain Oracle nodes retrieve necessary data through off-chain data APIs, format it, and return it to the Oracle Module. To protect the system from attackers attempting to control the majority of oracles, Yala Protocol uses the OSM to receive price inputs rather than directly from oracles. OSM acts as a defense layer between oracles and the protocol, publishing prices with a 30-minute delay to allow for emergency defenses in case of ora-

cle compromise. The Yala Foundation makes decisions on emergency oracles and the duration of price delays.

YU Vaults Module: $YU is a stablecoin asset initially issued by the Yala Foundation. All issued $YU is immediately deposited into the $YU Vaults Module as the Stablecoin Reserve Pool.

3.2. Application layer

The Application Layer of Yala includes modules for lending, stablecoins, insurance, restaking, and yield farming, with the stablecoin module serving as the foundation of the Yala Finance system.

Yala's stablecoin, $YU, uses an over-collateralization mechanism. Borrowers can initiate a Loan Module, transferring excess BTC assets into a UTXO. Automated nodes monitor when the Loan Module conditions are met. At this point, the corresponding amount of $YU is transferred to the borrower's address. Loan UTXOs can only be initiated by overcollateralized participants who enter a staking contract with the Yala Module. The Oracle network determines pricing for the UTXO. This same process is applied in the asset-mapping model.

The insurance module is uniquely designed to address the challenges of long BTC block intervals and high transaction risk. It implements a decentralized insurance mechanism using Takaful.

In the restaking module, BTC assets are mapped to other chains via atomic swaps. By expanding the multi-signature channels in atomic swaps from a 2-of-3 multi-sig to an (n-1)-of-n multi-sig channel, this method aims to enable a BTC-based restaking solution.[11].

The yield farming module involves a more complex logical design and is currently only considered for implementation within the asset-mapping model.

3.3. Data availability layer

DeFi transactions require high throughput and frequently generate large amounts of transaction data. Publishing DeFi-generated data to BTC would incur significant costs; hence, DeFi transaction data cannot be directly stored on BTC. In the Yala system, off-chain state updates and consensus are achieved by Prover nodes, which also maintain the security and data availability of the off-chain state. This approach saves the cost of uploading data to BTC while providing real-time updates for DeFi states.

Data availability refers to the capability of data stored in the blockchain network to be effectively accessed and used by all participating nodes. Unlike data availability in other blockchains, BTC assets are in UTXO format, comprising two state changes, and the BTC block interval is long, about 10 minutes. When transacting BTC assets, we do not need to deal with any data related to the said assets. Provers capture the global state and balance in the form of witness scripts. Thus, the implementation of data availability in Yala only needs to target off-chain state changes of assets in witness script format, which Provers maintains. The on-chain state change of UTXO is reflected in transfer transactions, which can provide security for off-chain state changes. The change in the off-chain state will eventually be reflected in on-chain UTXO transfer transactions, with BTC's consensus and security verifying the final state of asset transactions on Yala. Assuming that Provers are trustworthy and that the final changes in the off-chain state synchronize with UTXO transactions, the off-chain state's data availability can be independent of the BTC main chain. Our challenge is implementing Provers' trustworthiness and the secure real-time update of off-chain states.

Solutions for data availability include Data Availability Sampling (DAS), which has been validated in the EVM ecosystem. In Yala's DA layer, we can directly use DAS to reduce verification costs; validators only need to randomly download part of the data blocks to verify that all data is available. Erasure coding and KZG Polynomial Commitments will also implement DAS in Yala's DA layer. The content to be verified by Yala's DA layer includes transaction data, the Merkle tree of transaction data, and Commitments. Transaction data will be directly verified by Provers, who will also generate the Merkle tree. Provers can also produce commitments, and we may consider adding dedicated validators in the future. The proof content will be distributed and made public for a period set to one month (referencing the storage time of blobs on ETH), during which anyone can verify the proof content. A specialized project is already in development for the BTC DA layer solution; Nubit[2] is a Bitcoin-Native data availability layer with instant finality. We are considering using Nubit's DA layer solution directly to realize Yala's decentralized Prover network and DA layer to ensure the trusted premise assumption of Yala's DA layer Provers.

In addition, considering another performance requirement of DeFi based on BTC — the secure real-time update of off-chain states, we have incorporated the Mempool. Mempool is a dynamic staging area for unconfirmed transactions, storing information related to unconfirmed transactions. BTC's long block interval could limit DeFi's TPS (Transactions Per Second). By using Mempool, Yala can complete interim state changes of DeFi transactions after Prover consensus, pack multiple DeFi transactions into a batch, compute all state changes within the batch to aggregate into the final UTXO state change transfer transaction, and submit it to the BTC main chain for settlement. This process is essentially an off-chain state change Rollup, except that its verification process is executed off the BTC chain, ultimately reflected as a UTXO transaction on the BTC main chain.

3.4. Settlement layer

The BTC main chain finalizes transactions. We ultimately attribute the security of complex DeFi logic to the security of UTXO transactions, successfully relying on BTC's consensus and security and utilizing BTC's maintenance of the UTXO state ledger as the system's transaction state data availability.

4. BTC Assets State Machine Model

The asset state definition and transition are essential to implement complex DeFi transaction logic on BTC. We discussed state management models suited to different DeFi protocols, noting that both models can complement each other without conflict. Since BTC's mainnet operates on the UTXO model, we define simple UTXO script states using a Lock-Unlock mechanism. Within various state types, Yala will employ one-time signatures with multisig. The BTC Asset State Machine Model outlines the lifecycle of BTC assets through deposit, transfer, lock, liquidation, and redemption stages, along with their state transitions.

[2] Nubit: Bitcoin-Native Data Availability Layer with Instant Finality

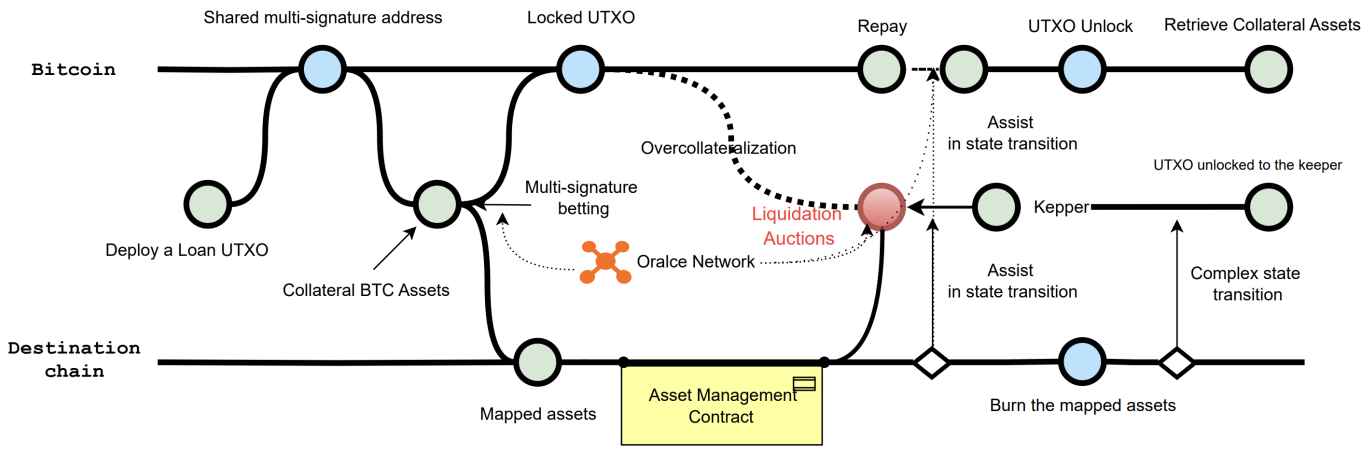

Fig. 2. Yala Loan protocol state machine

4.1. State Types

The main states of UTXOs (Unspent Transaction Outputs) currently include Unspent, Spent, Locked, and Pending. The Unspent and Spent states are the basic transaction statuses. Unspent is the initial state of a UTXO, meaning that the output has not yet been used and can be used as an input to construct a new transaction. Unspent TXOs are stored as key-value pairs in the UTXO pool. When a UTXO is used as an input for a new transaction, it is marked as "Spent." Once spent, the UTXO becomes unavailable and cannot be used for future transactions. When a UTXO is spent, the code removes the UTXO from the UTXO set, meaning that the UTXO is no longer recorded in the UTXO set, indicating that it has been spent. Pending is when a new transaction is created that includes a UTXO as input, putting the UTXO into a "Pending" state until the transaction is confirmed by miners and recorded on the blockchain. Once the transaction is confirmed, the UTXO state is updated to "Spent."

The Locked state is not directly reflected by the standard UTXO definition but rather by additional conditions (such as time locks, multi-signature, etc.). Although these UTXOs appear unspent on the blockchain, they are restricted by specific situations and cannot be spent until those conditions are met. The output in a user-initiated transaction includes a locking script (i.e., ‘scriptPubKey‘), which defines the conditions required to spend that UTXO. This could be a simple address (like P2PKH) or more complex conditions (such as P2SH, P2WPKH, Tapscript, multi-signature, etc.).

The Locked state is controlled by scripts in UTXO using the Script language. The Script language is a stack-based language without loops and complex flow control, ensuring it is not Turing complete. This reduces complexity and makes execution time predictable, avoiding denial-of-service attacks (as all full nodes verify each transaction) due to infinite loops or other logic flaws. Consequently, UTXO scripts have no variables, memory, or persistent state information. Each Script execution process is independent and cannot rely on previous execution results.

4.2. Implementation of Complex States

Yala Finance implements a complex BTC asset state machine model based on UTXO Script. The main implementation idea of Yala Finance is to use a Locked UTXO to map BTC assets onto other stateful blockchains to achieve complex transaction logic. The main methods of asset mapping include multi-signature, lightweight proofs, witnesses, sidechains, relays, distributed keys, atomic swaps, etc. Among these, multi-signature and atomic swaps are relatively suitable for implementation within the UTXO model. The asset mapping process effectively locks BTC assets in a UTXO, allowing users to obtain corresponding assets on the target chain. The unlocking condition of this UTXO is the completion of the relative settlement of the locked assets on the target chain. The assets on the target chain can undergo complex state transformations, with their value level and final settlement tied to the UTXO on the BTC main chain.

Yala Finance also explores the implementation of a complex state machine on the BTC main chain. One-time signatures have an anti-ambiguity property, allowing the public key to be mapped into a concrete, immutable value. Users can use public keys to sign UTXOs with different preimages. During script execution, a state value is passed on the stack. For example, in the following Script code, if a signature corresponding to the preimage of ‘hash1‘ is provided, the signature will be verified, leaving the value ‘1‘ on the stack. If a signature corresponding to the preimage of ‘hash0‘ is provided, it will be verified, leaving the value ‘0‘ on the stack. Alternatively, the preimage can be revealed by verifying the bit value, executing the corresponding unlocking script.

```
OP HASH160
OP DUP
<0xf592e757267b7....>// This is hash1
OP EQUAL
OP DUP
OP ROT
<0x100b9f19ebd53....>// This is hash0
OP EQUAL
OP BOR
OP VERIFY
```

Passing a 1-bit state value can already enable complex transaction logic. Lamport signatures are used in BitVM as a bit commitment, establishing a link between two Bitcoin UTXOs to enable stateful Bitcoin scripts. In Taproot addresses, a large program is committed, with operators and validators engaging in extensive off-chain interactions, leaving minimal on-chain footprints. If both parties cooperate, complex off-chain computation can be executed without leaving on-chain traces. However, if a dispute arises, on-chain execution is required. Yala Finance will

implement simple conditional scripts on the BTC main chain and combine multi-signature for basic DeFi applications with other chains necessary to update the state in the UTXO.

4.3. State Machine on Bitcoin Mainnet

We initiate the collateral lending process on the Bitcoin main chain using multisig bets. Users first create a Loan UTXO, which deploys the Yala-provided fund-locking protocol script, with parameters adjusted based on Oracle and automatic stabilizer feedback. The betting party can be a Yala fund management contract on another chain. Once the Loan UTXO is created, the collateral is locked. After successful locking, Yala contracts on other chains issue $YU stablecoins to the borrower.

If the user repays the loan, they can unlock the UTXO and reclaim their collateral. The liquidation process is triggered if the user fails to repay on time or the collateral value falls below the liquidation threshold. Keepers participate in liquidation by transferring funds to the Yala contract to purchase the bad debt. After triggering liquidation, the UTXO unlocks, and funds are sent to the Yala contract address. A new multi-sig UTXO is created, and Yala signs the transaction upon receiving the liquidation funds, unlocking the UTXO and transferring it to the Keepers.

This state transition process on Bitcoin involves cross-chain contract cooperation, oracle network supervision, and off-chain state transfer assistance. The betting process uses one-time signatures to determine the trade's state, depending on which party fails the bet. Off-chain state data is transmitted during the liquidation process, requiring further security design and analysis, with the Nubit Prover network used for transaction verification.

4.4. State Machine on the Destination Chain

Yala Finance's solution relies on contracts on the destination chain for state transfer and asset mapping. In asset mapping, BTC assets are locked in UTXOs on the main chain and cross-chain mapped through multisignatures. Users receive mapped assets on the destination chain. Yala Finance implements asset management contracts on the destination chain for asset issuance, collateral lending, and other complex DeFi logic. Liquidation requires Keepers' participation, but the state transition occurs directly on the destination chain, where Keepers can choose to acquire the collateral UTXO or mapped assets. The mapped asset is burned after the locked UTXO on the main chain is unlocked.

5. Automatic stabilizer and the Yala Foundation

In its initial phase, the Yala Foundation operates as a centralized entity under community oversight, ensuring $YU's stability through BTC-backed lending protocols. The Foundation manages lending parameters, interest rates, and emergency responses. The protocol's BTC assets are secured through Bitcoin vaults using threshold signatures, with the BTC Stability Reserve Pool managed by Yala token holders who vote on system parameters and upgrades. The Foundation issues initial $YALA and $YU tokens through verified BTC-collateralized positions. While the Foundation cannot alter the underlying BTC collateral positions, it implements an Automatic Stabilizer system to manage the $YALA and $YU supply dynamics, automating the minting and burning process based on market conditions.

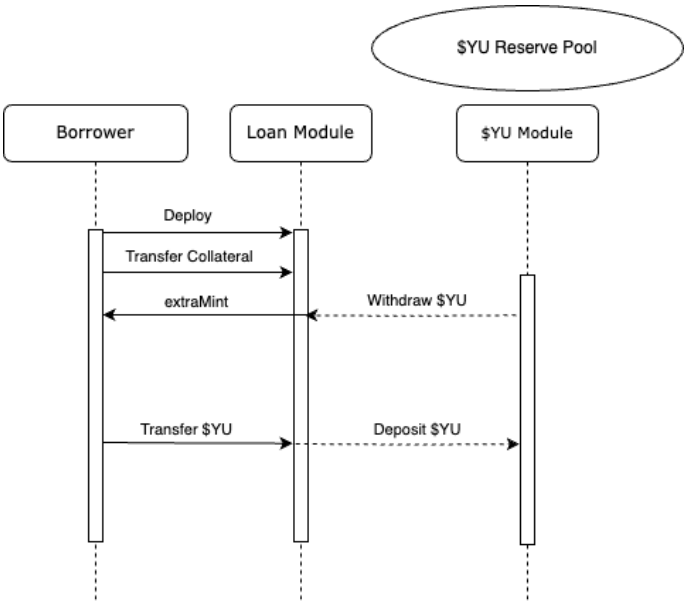

Fig. 3. Automatic stabilizer on credit process

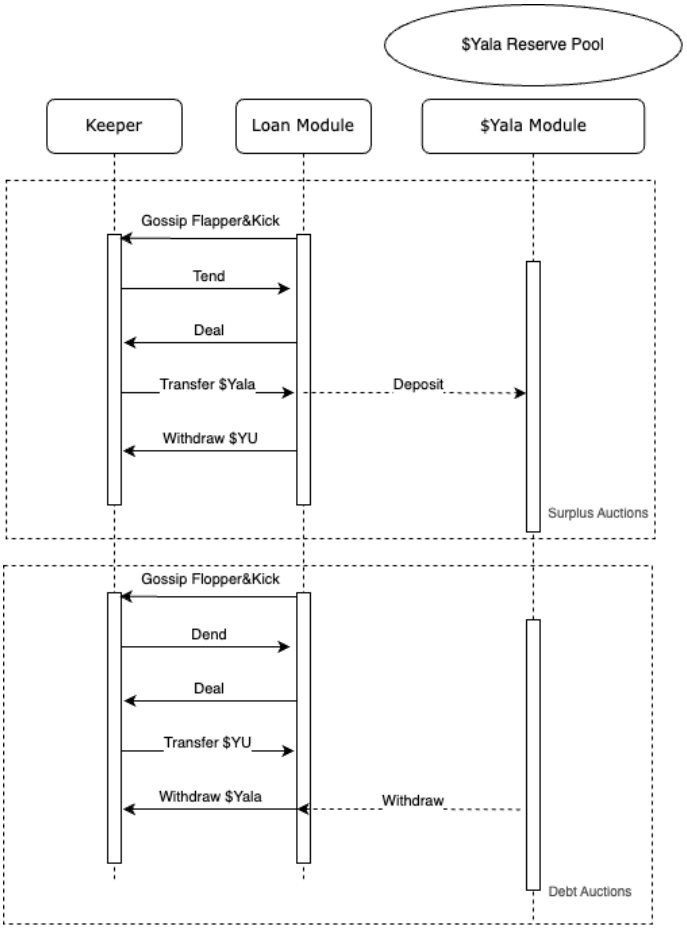

Fig. 4. Automatic stabilizer on system stabilization process

5.1. Credit process

The Yala Foundation will mint all $YU to the $YU Reserve Pool until reaching its max supply, then transition to the Stabilized Lending phase. In this phase, new $YU isn't directly minted; instead, an "extraMint" method simulates minting by transferring

$YU from the Reserve Pool to borrowers. Repayments are redeposited into the Reserve Pool, mimicking the original minting phase. The Automatic Stabilizer automates withdrawals and deposits related to the Reserve Pool, managing the $YU Module throughout.

5.2. System stabilization process

The Automatic Stabilizer manages $YU's price stability through a dual-auction system during the BTC lending cycle. The process handles minting and burning operations, focusing on maintaining stability during liquidation events. Surplus value is generated from the Loan Module's management fees (interest rates). When loans are repaid, excess interest becomes surplus tied to specific module addresses. Keepers participate in auctions using $Yala tokens, with winning bids securing the surplus. The $Yala tokens received through these auctions are added to the $Yala Reserve Pool, reducing the governance token supply to maintain economic equilibrium.

During rapid collateral price declines, bidders may pay significantly less than their maximum willingness, resulting in the liquidation process generating less $YU than the loan's value. This triggers the Debt Auction phase of the Loan Module, which announces the total debt and its address for bids. Bidders use $Yala for their offers, with the auction favoring the lowest bidder. The winning bidder transfers $YU to the Loan Module, secured by bidding with $Yala. The Automatic Stabilizer then compensates the winner by drawing $Yala from the Reserve Pool, effectively simulating the minting of governance tokens.

6. Liquidation

If a user's collateral value in the Loan Module drops sharply and fails to repay interest or add more collateral before reaching the minimum collateralization ratio, Keepers can initiate liquidation[5, 8]. The process begins when Keepers respond to liquidation signals from the Insurance Module, specifying the Module address and liquidation amount. The Module becomes unlocked after sending BTC to the specified address and receiving verification from the Prover. The Keeper can then withdraw the collateral. The liquidation fee is predetermined by the Yala Foundation in the Insurance contract, allowing Keepers to purchase collateral at a discount for arbitrage opportunities.

Suppose the collateral value associated with a user's Loan Module experiences a significant decline, and the user fails to settle interest payments or bolster collateral funds promptly, triggering a drop below the stipulated minimum pledge rate. In that case, Keepers can submit a liquidation request to the Loan Module [5, 8]. Keepers must adhere to the liquidation amount broadcasted by the Insurance Module and the corresponding Module's address. They can then execute a direct transfer to this address. Upon consensus agreement from the Prover, the Module undergoes unlocking, transforming into a White Module. Subsequently, the Keeper initiates a withdrawal request, securing the escalated collateral. The clearing fee, configured within the Insurance contract script by the Yala Foundation, presents an opportunity for Keepers to acquire collateral at a discounted rate, capitalizing on arbitrage gains.

We consider multiple liquidation execution methods, some of which have similar implementation methods. One is to complete it directly on the BTC main chain, where the collateral is locked in a UTXO. This is a multi-signature management module with predefined unlocking conditions from the start. The second approach uses atomic swaps, where the counterparty for the collateral user is set as an open-source contract. The counterparty contract can be on the BTC main chain, another chain, or Layer 2. The third approach achieves BTC asset cross-chain interoperability through decentralized custody; we must complete decentralized multi-signature managed asset custody on the BTC main chain and interact with financial operations with the target chain.

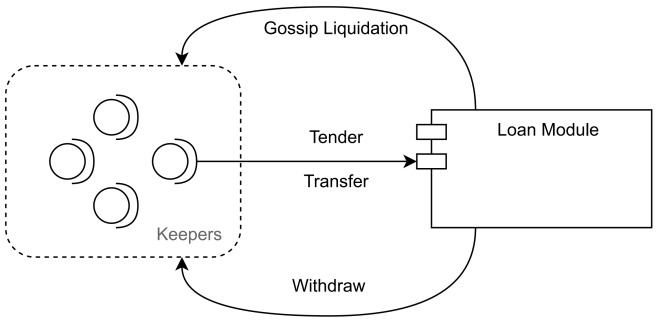

Fig. 5. Liquidation

In the first approach, the UTXO unlocking condition can be signed by an oracle, making the execution process of the module similar to Discreet Log Contracts (DLCs). The second approach relies on atomic swaps and uses Hash Time-Locked Contracts (HTLC) to facilitate trustless digital asset exchanges. Utilizing HTLC to control the assets locked by both parties, we manage the complex financial transaction process, which may also depend on oracles. Atomic swaps support two types of transactions: cross-chain exchanges between two different cryptocurrencies on two separate blockchains and on-chain (including Layer 2) exchanges of other digital assets on the main chain. The third approach involves mapping BTC assets onto another blockchain to complete operations, ensuring that custodians do not act maliciously and that user assets remain secure.

7. Custodial mapping of BTC assets

The custodial mapping of BTC assets is a cross-chain mechanism that typically involves locking BTC in UTXOs on the Bitcoin main chain and mapping those assets to other chains through technologies such as multi-signature. When users meet specific conditions (such as loan repayment), the UTXO is unlocked, and the corresponding mapped assets are released or destroyed. This process often requires the collaboration of oracles and cross-chain contracts to ensure the secure and accurate transfer of assets. This allows BTC assets to be used on other chains while maintaining their security on the main chain.

The cross-chain process can be divided into the issuance and destruction of mapped assets. Assets to be moved cross-chain are securely locked on the source chain, and equivalent mapped assets are issued on the target chain—this represents the cross-chain movement and issuance of mapped assets. Destroying the mapped assets on the target chain and unlocking the assets on the source chain is bringing the assets back cross-chain. In these processes, the technical goal is to ensure the secure atomicity of the lock-issue and destroy-unlock actions. Almost all BTC cross-chain solutions use MPC (multi-party computation), involving multiple institutions and community participants as trusted parties for multi-signature authentication. However, cross-chain bridge signatories are few, leading to issues of excessive centralization and vulnerability to attacks.

In Yala Finance's implementation of the BTC asset cross-chain solution, there is a greater focus on the security of user assets and the decentralization of asset custody. We will also implement more complex mapping operations, directly integrating collateralized lending and stablecoin systems within the acceptance contracts on the target chain.

7.1. Key Management

Key management refers to managing and protecting the entire process of generating, storing, using, and destroying cryptographic keys. It ensures the security of encrypted assets and prevents unauthorized access or tampering. By employing methods such as distributed management, multi-signatures, and Hardware Security Modules (HSMs), key management can reduce the risks of asset leakage and loss, especially during cross-chain operations and asset custody. It also ensures the secure transfer and verification of assets across different platforms and chains.

Key management for BTC asset custody and cross-chain operations is crucial for ensuring asset custody security and cross-chain mapping safety. Key management ensures the security of assets and prevents unauthorized access. Through distributed management and multi-signature mechanisms, key management effectively mitigates risks, such as single points of failure or malicious attacks. Additionally, its role in cross-chain asset mapping and verification cannot be overlooked. By using external verification mechanisms such as oracles, key management ensures the transparency and security of cross-chain operations, safeguarding the secure transfer and management of user assets across different chains.

The security strategies for key management include the following aspects:

1. **Multi-signature:** Verifying transactions through a combination of multiple keys to prevent the risk of a single key being compromised or altered.
2. **Distributed Storage:** Avoiding the storage of all keys in a single location, thereby reducing the risk of attacks.
3. **Hardware Security Module (HSM):** Using dedicated hardware devices to generate and store keys, enhancing their protection.
4. **Key Rotation and Destruction:** Regularly updating keys and ensuring that expired keys are destroyed to prevent over-reliance on a single key.

When used for key management, the MPC (Multi-Party Computation) protocol enhances key security by splitting the key into multiple shares and distributing them among different parties. This ensures that no single participant can obtain the complete information of the key, thus mitigating the risks of single points of failure and singular key leakage and ensuring the security of the key during use.

Additionally, the MPC protocol supports dynamic key updates and recovery, providing flexible security guarantees when collaborating with multiple parties. However, implementing MPC protocols typically requires higher computational costs and network bandwidth.

A reasonable key management approach is necessary to establish a layered defense for solutions such as locked custody and automated asset management tools. Cubist[3] has identified the flaws in existing key management solutions and designed a more effective implementation.

[3] https://cubist.dev/

In an MPC (Multi-Party Computation) protocol, multiple parties connected over a network perform joint computations without disclosing their private inputs, facilitated by complex cryptographic techniques. However, the network infrastructure and the associated cryptography can be costly, which is essential for proper protocol implementation—particularly for those prioritizing security over shortcuts. Consequently, MPC wallets require substantial cryptographic resources and generate significant network traffic, leading many legitimate implementations to take approximately ten seconds to produce a signature.

7.2. Existing Solution Reference

From a systematic perspective, asset custody services provided by alliances consisting of multiple independent custodians exhibit better robustness and resilience against single points of failure, thereby achieving higher security. Asset custody solutions frequently encounter incidents of funds being stolen or attacked, making it difficult for clients to distinguish whether the claimed losses are due to hacking or internal fraud and misuse.

There are various asset custody mapping solutions on Bitcoin, primarily including:

- **Centralized:** Custody is managed by a trusted central institution, fully endorsed by that institution, such as H-Tokens, BTCx, and imBTC.
- **Consortium:** Multi-signature accounts controlled by a consortium of members, backed by the reputation of consortium members, such as WBTC, cBTC, and XBTC.
- **Decentralized (with deposits/collateral):** Custody provided by permissionless custodians secured by over-collateralized cryptographic assets (e.g., tBTC and renBTC).

Yala's custodial mapping solution draws inspiration from the MakerDAO[7] and WBTC models, similar to the approach of tBTC[10]. tBTC is a multi-signature address, over-collateralized, and randomly combined automated protocol. The signer group consists of multiple signers, making them important participants in the protocol. The signer group provides a BTC custody address for users to lock BTC assets. The selection of the signer group is facilitated by a random seed supplied by a random network, and signers need to provide over-collateralized assets to participate in the asset mapping process.

The overall process requires multiple rounds of participation from users and signers, accompanied by numerous restrictions, such as minimum amounts and waiting periods. Additionally, issues related to asset price oracles, asset utilization, and various frictions are involved due to over-collateralization.

TBTC and renBTC offer the most reliable security, as adversaries are unlikely to initiate attacks that would not be profitable. However, these solutions also have significant drawbacks.

The first drawback is that over-collateralization leads to inefficiency: tBTC requires custodians to provide collateral worth 150% of the client's asset value, while renBTC requires 300% collateral.

The second drawback is that these solutions do not support homogeneous collateral as custodial assets.

7.3. Decentralized Asset Custody

Asset custody requires consideration of decentralization, security, and efficiency. Each custodian participating in asset custody must provide a fund as a margin, which is kept with the assets under custody and will be used to reimburse any losses incurred by

a misbehaving custodian. This allows for sharing risk and goes beyond having a single point of control over the management of funds. In terms of security, the system remains secure as long as the adversary does not steal more assets than the margin paid, i.e., it is preferable that the adversary only withdraws the margin of the custodian under his control instead of launching an attack. We refer to the scheme of Zhaohua Chen and Guang Yang[3]. This scheme uses a portfolio design to decentralize custodians and assets into many small custodian groups. Each group has full control over its assets, which reduces operational costs and increases activity (a single surviving group can process transactions). The core processing structure is:

Definition 1 (Decentralized Asset Custody scheme). A custody scheme (S, A, μ) consists of the following three parts:

- $S = \{1, 2, \cdots, n\}$ denotes the set of all custodians (or simply nodes);
- A denotes a family of m k-subsets of S, such that each element in A (i.e. a k-subset of S) represents a custodian group under the given custody scheme;
- $\mu \in [1/2, 1)$ denotes a universal authentication threshold for all custodian groups, i.e., the asset controlled by that group can be settled arbitrarily with approval of strictly above μk group members.

In decentralized asset custody schemes, random sampling can effectively reduce the number of custodian groups, thereby improving system manageability. In the original design, depending on whether a symmetric or polynomial approach is used, the number of custodian groups may become excessively large, making the system costly and difficult to operate. Random sampling selects only a subset of custodian groups to act as representatives, significantly simplifying the structure.

Importantly, this approach does not significantly compromise the system's security while reducing the number of custodian groups. Theoretical analyses demonstrate that the optimized scheme retains a high probability of security and establishes a lower bound on its efficiency factor. This indicates that random sampling achieves a practical balance between feasibility and security, providing a more efficient implementation path for decentralized custody mechanisms.

8. Takaful

Takaful operates by pooling contributions from members into a fund, with premiums financing mutual assistance for losses. This fund, along with all related activities such as investments and profit distribution, is managed by an operator. In cases where the Takaful Insurance Fund falls short of covering claims, operators use qard hasan (a benevolent loan) from company shareholders to fill the gap. Though not required, future surpluses are first allocated to repay these loans, benefiting the shareholders.

Surpluses remain after claims and expenses are reimbursed to participants, with the operational cycle and calculations outlined by the operator in a scripted contract. The fund employs a detailed rebate mechanism, with thresholds designed to regulate fund withdrawals and maintain stability. Commonly, five thresholds are established: three for the Takaful fund to manage its levels and two for Qard Hasan to guide loans and repayments. Loans are made to elevate the fund to a safe level during deficits, and as the fund grows, reaching certain triggers allows for investments, distribution of excess to participants and the operator, and reserve setting for the future. Surplus in the qard hasan beyond a defined point triggers dividends to participants, ensuring equitable benefit distribution.

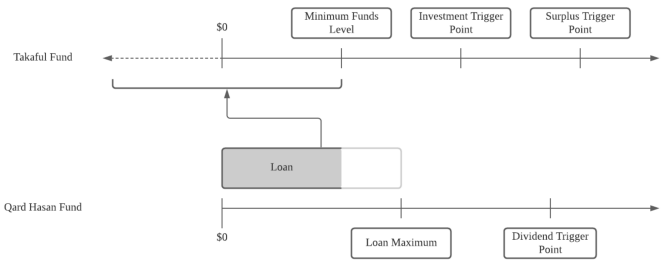

Fig. 6. KD modified risk model

Assume that all n participants pay a premium of d to the Takaful insurance fund. The R_t denotes the reserve after all payments have been made in the period starting at time t. The net return is the total income minus the total expenses for a given period. The total amount of benefits withdrawn from the Takaful Insurance Fund is S. The net benefit is the total revenue minus the total expenses for a given period, i.e., $nd - S$. h is the surplus trigger point, and any amount exceeding the surplus trigger point at the point in the calculation cycle is the fund's surplus, which should be distributed to the participants, the Takaful operator, and the shareholders. Takaful fund reserves satisfy the recurrence relation

$$R_t = h \wedge (R_{t-1} + nd - S)_+ \tag{1}$$

8.1. Insurance model in Yala

Yala's insurance model merges the profit-sharing and agent models, drawing inspiration from Takaful. In the profit-sharing approach, the operator allocates the Takaful fund's surplus proceeds once they surpass a predetermined surplus trigger. Conversely, in the agent model, the operator earns proceeds for fund management, creating a balanced structure that rewards both fund stewardship and communal surplus.

In the Yala insurance model, insurers oversee the Takaful Module and Qard Hasan Module. The Takaful Module, central to the system, manages the insurance pool and the lending script contract. Here, policyholders, termed Participants, choose their insurance module and initiate a script contract linked to the Takaful Module. This setup, pre-approved by Insurers, automatically entitles Participants to payouts, triggering a process where, upon permission, the Prover consensus transitions the module from Black to White, enabling Insurers to access the funds for claims.

The Insurance Pool's resources are allocated for claims and Insurers' fees, embodying shared risk among Participants. Insurers can invest to generate additional revenue once the pool's funds surpass the investment threshold. Beyond a set surplus trigger, this excess income is shared between Participants and Insurers, with any leftover reserves earmarked for future insurance periods.

The Qard Hasan Module is triggered when a deficiency occurs, and the Insurance Pool reaches the Minimum funding level. The funds in the Qard Hasan Pool are used to cover the Insurance Pool's losses until its funding is restored to the Loan Maximum Point. That is, when $(R_{t-1} + nd)(1 - \rho^w) < S$, the Qard Hasan loan $Q = S - (R_{t-1} + nd)(1 - \rho^w)$ will be offered.where ρ^w is the agency fee rate and ρ^m is the profit sharing rate. If the body of Insurance Pool funds exceeds the surplus trigger at stage t stage, the reserve satisfies

$$R_t = h \wedge \left[((R_{t-1} + nd)(1 - \rho^w) - S)(1 - \rho^m)\right]_+ \tag{2}$$

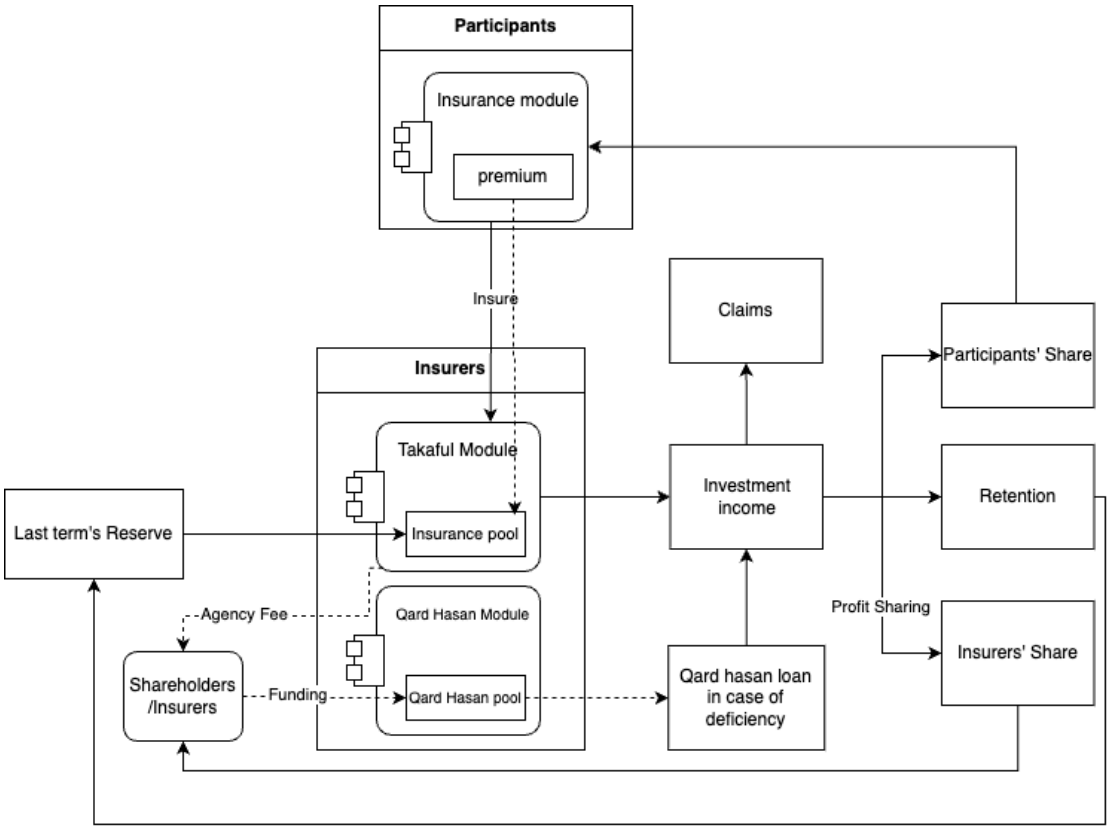

Fig. 7. Insurance model architecture

Each participant receives benefits

$$G_t = \left[\left(\left(\frac{R_{t-1}}{n} + d\right)(1-\rho^w) - \frac{S}{n}\right)(1-\rho^m) - \frac{h}{n}\right]_+ \tag{3}$$

If we set the profit threshold point $h = 0$, the earnings for each participant in the model simplify to:

$$G_t = \left[\left(\frac{R_{t-1}}{n} + d\right)(1-\rho^w) - \frac{S}{n}\right](1-\rho^m) \tag{4}$$

If the body of insurance pool funds does not reach the surplus trigger at stage t, no profit sharing will be calculated for the insurers, which will satisfy the reserve.

$$R_t = \left[\left(\frac{R_{t-1} + nd}{1-\rho^w} - S\right)\right]_+ \tag{5}$$

Since Yala Finance is a high-risk trading scenario, there could be a large demand for short-term insurance. Insurers could reasonably increase insurance rates for short-term traders. Contract parameters would need to be changed at any time in conjunction with the market price provided by the prognosticator, and Insurers would have the right to reject high-risk trades. The Yala insurance model could also become a Defi application. The Yala insurance model could also be a standalone derivative in the BTC ecosystem, where policyholders and shareholders could provide capital for long-term gains.

8.2. Key external actors

8.2.1. Keepers

Keepers, often automated and independent entities, are motivated by arbitrage to enhance liquidity in decentralized systems. In Yala Finance, they help stabilize $YU's price by selling it when above the target and buying when below. During liquidations in Yala's Vault module, Keepers engage in auctions for surplus, debt, and collateral, contributing to the protocol's market efficiency.

8.2.2. Oracles

Yala Finance requires real-time information about the market prices of collateral assets in the Yala Finance Vault to determine when to trigger liquidation. Oracles provides data services for USD prices, collateral prices, and $YU prices to both the Yala Foundation, insurers, and the liquidation module [1, 2]. The Yala Foundation oversees the Oracle and OSM Modules, which source price inputs from the BTC Layer 2 Oracle node. This node fetches off-chain data, reformats it, and supplies it to the Oracle Module. To safeguard against attackers controlling prophecy machines, the OSM (Office of the Secretary) intermediates, receiving prices indirectly to enhance security. It introduces a 30-minute delay before releasing prices to the protocol, allowing time for emergency actions if needed. The Yala Foundation decides on this delay and emergency protocols.

Given the limited collateral types in early Yala Finance, the Oracle system doesn't require extensive price data. It operates on a publish-subscribe model, with the Oracle Module collecting and broadcasting price data at set intervals after OSM processing. Each node in the system runs a daemon to keep its price data current.

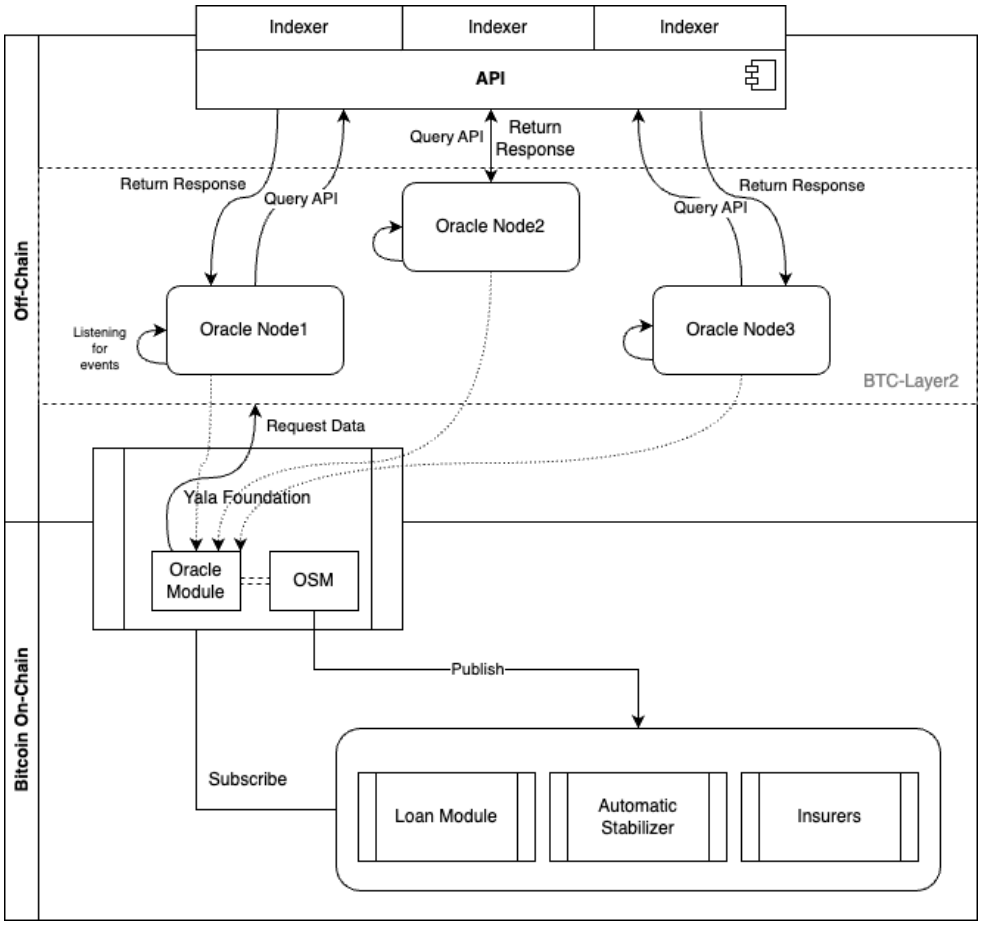

Fig. 8. Oracle service

8.2.3. Insurers

Given the high volatility unique to Yala Finance, collateralized borrowers require insurance services. However, to preserve $YU stability, the Yala Foundation doesn't directly provide these services or assume borrower risks. Instead, when starting a Loan Module, borrowers can activate an Insure Module tailored to that specific module.

Insurers are insurance operators responsible for maintaining the Qard Hasan fund pool, setting various parameters for insurance contracts, and publishing them on the insurance platform. Before initiating the Insure Module, borrowers need to select an insurance contract on the insurance platform and then direct the source to the script code segment to complete the publication of the Insure Module.

9. Interoperability

9.1. Secure state channels between modules

In Yala's architectural design, state changes operate independently, creating state channels running within UTXOs on BTC. These state channels provide security, with the occurring changes being off-chain, thus not burdening the BTC main chain's load. The execution function for state changes is stored in MAST conditional circuits scripts in the Witness on BTC. It simply evaluates if the conditions for state change are met without conducting substantial complex calculations, ensuring the contract's security, simplicity, and transparency.

Analyzing the size and content of BTC blocks, according to information provided by bitaps, although the block weight of BTC blocks is limited to 4MB, the average size of the block weight is only 1724885 bytes (Data on March 10, 2024) when the Base size is close to 1MB. Therefore, at least 2MB of space can be fully provided to the witness, with an average of approximately 3500 transactions per block. We can estimate that if each transaction uses taproot script-path spend scripts in the witness to place contracts, each contract averages 600 bytes.

BRC-20 Modules' conditional circuits prescribe the rules for state changes, necessitating MultiSig Vault. The interaction process broadly aligns with the MultiSig Vault setup, involving DeFi participants (excessive collateral borrowers or atomic swap participants), oracles, and Token Modules. The oracles are decentralized and considered trustworthy after processing by the Oracle Module's OSM. Token Modules are divided into Stablecoin Modules implemented by MAST and token contract modules on other chains. The Stablecoin Module determines the permission conditions allowing users to withdraw $YU, and participants also initiate transfers to it during settlement, with the transfer operation being freely executed without permission. In atomic swap scenarios, token contract modules on other chains have Turing completeness and the contract security of different chains. Token Modules are automatically executed under the consensus of Provers and can be considered trustworthy. We believe such state channels are trustworthy under conditions where only the single transaction participant is untrustworthy.

When DeFi participants initiate the transaction module, its source will be the DeFi transaction script published by the Yala community. The transaction process forms a 2-of-3 multisig state channel between the participant, Oracle Module, and Token Module. The unlocking conditions are two: one is controlled by a time lock, and if participants do not return the borrowed assets within the specified time, the assets will be directly paid to the Token Module. MAST conditions control the other, setting the wager on whether participants can repay the loan. In the borrowing scenario, currently, only over-collateralized loans with pre-set interest values can be implemented, meaning the interest and loan duration are predefined from the start, not variable over time, due to the script size limitations currently supported by the Witness.

9.2. Atomic swap

We can readily observe the limitations of operating assets on the BTC main chain. Therefore, we can expand BTC by enhancing its interoperability to facilitate more complex DeFi transactions with BTC assets. Participants need to initiate an atomic swap module on the BTC main chain, pledge a certain amount of BTC or other assets, and then obtain an equivalent amount of wrapped tokens (yBTC) on another blockchain to engage in the DeFi ecosystem of different chains.

We refer to the atomic swap protocol proposed by TierNolan on Bitcointalk[4]:

1. Party 'A' generates some random data, x (the secret).
2. Party 'A' generates Tx1 (the payment) containing an output with the chain-trade script in it. See below for this script and a discussion of it. It allows coin release either by signing with the two keys (key 'A' and key 'B') or with (secret 'x', key 'B'). This transaction is not broadcast. The chain release script contains hashes, not the actual secrets themselves.
3. Party 'A' generates Tx2 (the contract), which spends Tx1 and has an output going back to key 'A'. It has a lock time in the future and the input has a sequence number of zero, so it can be replaced. 'A' signs Tx2 and sends it to 'B', who also signs it and sends it back.
4. 'A' broadcasts Tx1 and Tx2. Party 'B' can now see the coins but cannot spend them because it does not have an output going to him, and the transaction is not finalized anyway.
5. 'B' performs the same scheme in reverse on the alternative chain. The lock time for 'B' should be much smaller than for 'A'. Both sides of the trade are now pending but incomplete.
6. Since 'A' knows the secret, 'A' can claim his coins immediately. However, 'A', in the process of claiming his coin, reveals the secret 'x' to 'B', who then uses it to finish the other side of the trade with ('x', key 'B').

The protocol process begins with user A initiating an atomic swap contract white module targeted at a specific chain. The randomly generated number x and the lock time t_1 are written into the unlock script of the white module. There is a contract on the target chain that interacts with the user, defined as role B, which executes automatically. The unlock condition for the white module is [if (x for H(x) known and signed by B) or (signed by A & B),] if not successfully signed and unlocked, a refund transaction to the user will be initiated after time t_1. Then, user A initiates a transfer transaction to the white module, transferring a certain amount of BTC. The Prover captures the white module, triggering the execution of the atomic swap contract on the target chain and initiating an equivalent yBTC transfer transaction to the user's account on that chain with the unlock condition. [if (x for H(x) known and signed by A) or (signed by A & B),] If not successfully unlocked within time t_2, the transaction will be canceled, where $t_2 < t_1$. User A signs the transaction on the target chain and uses and reveals the secret x, acquiring yBTC on the target chain. Contract B captures the secret x and simultaneously unlocks the white module. The final unlock of the white module requires consensus through the Prover; at this time, the ownership of the white module belongs to the Prover network. The BTC assets pledged within the white module will only be used if a user from another chain exchanges yBTC back to BTC. The Oracle monitors the entire atomic swap process, and the contract on the other chain will decide whether to allow atomic swaps based on whether the number of pledged BTC counted by the Oracle is equal to the number of issued yBTC.

10. Future

While Yala has considered implementing native DeFi solutions for BTC assets, several challenges remain in ensuring interoperability between BTC and other blockchain networks. The initial approach of directly implementing atomic swaps and locks to facilitate two-way communication between BTC and other assets faces issues such as the lack of guaranteed counterparties, long cross-chain latency, value-blindness, and the risk of fraud.

The common approach adopted by most cross-chain bridges, involving multiple signatures from trusted organizations, could be considered as a potential solution. However, this raises significant security concerns, as it is well-known that many DeFi and cross-chain bridge projects have been subject to insider theft, where the custodians of the managed assets were responsible for the theft of funds. Decentralized asset management requires higher security in cross-chain asset mapping from BTC to other blockchains.

Yala needs to address the problem of how to tolerate untrustworthy multi-signatory parties in decentralized asset management, as well as how to increase the speed of cross-chain transactions. One potential solution could be to decouple transaction confirmation from transaction execution and improve the efficiency of multisignatures, which could significantly increase the speed of cross-chain transactions. Additionally, it might be possible to prevent malicious behavior by ensuring that the costs

[4] Bitcointalk.org - Atomic Swap Protocol by TierNolan

outweigh the benefits of an untrustworthy multi-signatory attacking the managed assets.

Beyond these technical challenges, Yala envisions becoming a cornerstone for Bitcoin's evolution into a programmable asset platform. This transformation will enable several key developments:

1. **Institutional Adoption:** Development of standardized Bitcoin DeFi infrastructure aligned with traditional finance requirements, professional risk management frameworks, and enterprise-grade security solutions with modular key management.
2. **Cross-chain Innovation:** Implement next-generation atomic swap mechanisms with enhanced efficiency, novel state verification methods, and improved cross-chain composability while maintaining Bitcoin's security model.
3. **Bitcoin DeFi Standards:** Establishment of industry reference implementations for Bitcoin-native DeFi protocols, open specifications for secure Bitcoin state transitions, and community-driven protocol improvements fostering ecosystem growth.

These developments align with Yala's core mission of unlocking Bitcoin's full potential while maintaining its fundamental security and decentralization principles. Overall, Yala recognizes the need to develop robust and secure cross-chain interoperability solutions to enable the seamless integration of BTC assets with other blockchain ecosystems while ensuring a high level of decentralization and trust in the asset management process.

11. Conclusions

Yala Protocol is dedicated to building a comprehensive DeFi and programmability infrastructure based on BTC assets. We have designed a native asset interaction protocol based on the BTC mainchain and established robust interoperability mechanisms from BTC to other blockchains. The protocol's architecture consists of several innovative components:

- The native operability of BTC assets is implemented based on Modular UTXO architecture, leveraging overcollateralized stabilization BTC assets to provide a full suite of services, including the Vaults module, liquidation algorithm, and automatic stabilizer.
- To address the unique challenges of BTC's block time and asset volatility, we introduced an insurance derivatives module that serves dual purposes: reducing risks for collateralized lending users while enabling profitable interest-bearing opportunities through the insurance mechanism.
- At the interoperability level, we implemented atomic swaps to achieve secure 2-way pegging between BTC and other blockchains while leveraging Turing-complete smart contracts on external chains for complex operational logic.

These innovations position Yala as a pioneer in Bitcoin's DeFi ecosystem, offering a secure, efficient, and scalable infrastructure that maintains Bitcoin's fundamental principles while expanding its utility. By combining Bitcoin's unmatched security with advanced DeFi capabilities, Yala creates a foundation for the next generation of Bitcoin-based financial applications.

References

[1] Al-Breiki, H., Rehman, M. H. U., Salah, K., and Svetinovic, D. Trustworthy blockchain oracles: review, comparison, and open research challenges. *IEEE access 8* (2020), 85675–85685.

[2] Caldarelli, G. Overview of blockchain oracle research. *Future Internet 14*, 6 (2022), 175.

[3] Chen, Z., and Yang, G. Decentralized asset custody scheme with security against rational adversary, 2021.

[4] Heilman, E. Signing a bitcoin transaction with lamport signatures. Bitcoin Development Mailing List, April 2024. Accessed from Bitcoin Development Mailing List.

[5] Kokorin, I., De Graaf, T., and Haentjens, M. The failed hopes of disintermediation: Crypto-custodian insolvency, legal risks and howto avoid them. *Singapore Journal of Legal Studies* (2020), 526–563.

[6] Liu, X., Ji, S., Wang, X., Liu, L., and Ren, Y. Blockchain data availability scheme with strong data privacy protection. *Information 14*, 2 (2023), 88.

[7] MakerDAO. The maker protocol: Makerdao's multi-collateral dai (mcd) system, 2024. Accessed: 2024-11-19.

[8] Perez, D., Werner, S. M., Xu, J., and Livshits, B. Liquidations: Defi on a knife-edge. In *Financial Cryptography and Data Security: 25th International Conference, FC 2021, Virtual Event, March 1–5, 2021, Revised Selected Papers, Part II 25* (2021), Springer, pp. 457–476.

[9] Poelstra, A. Script state from lamport signatures. *Blockstream Research* (July 2024).

[10] tBTC. A decentralized redeemable btc-backed erc-20 token, 2024. Accessed: 2024-11-19.

[11] Team, E. Eigenlayer: The restaking collective. *URL: https://docs. eigenlayer. xyz/overview/whitepaper* (2023).