Oraclizer: The Oracle State Machine Ensuring Regulatory Compliance

Facilitating Capital Efficiency between Traditional and Decentralized Finance in Tokenized Capital Markets

Horizen Korea, Oraclizer Core
Jinwook.Kim (Jay@oraclizer.io)
v.0.1.2 – November 14, 2024

Abstract

Oraclizer is a novel thesis introducing the first regulatory-compliant oracle state machine that **achieves state synchronization between on-chain and off-chain (even non-blockchain) oracle segments.** Through **zk proof-based oracle state verification**, it ensures the validity of bidirectional state synchronization, and by implementing a Layer 3 architecture utilizing the Validium approach and offloading zk verification computations, **it achieves cost reductions of over 93% compared to conventional oracle solutions**. Furthermore, unlike traditional oracles that charge per individual call, **Oraclizer realizes state synchronization oracle without user`s additional costs** by automatically managing continuous state updates on-chain after the initial state synchronization through a single oracle call.

Oraclizer adopts as its system design principle the Regulatory Compliance Protocol (RCP) paper, published in March 2024 by the Oraclizer Core team, which standardizes 31 key regulations from 15 global financial regulatory authorities that may impact tokenized capital markets. This integration inherently embeds and ensures financial regulatory compliance throughout the system. Notably, through integration with DAML(CANTON)-based financial RWA tokenization solutions, which guarantee contract completeness through privacy and robust authority models, it implements **complete state synchronization contracts** that differentiate it from existing oracle technologies. This facilitates **the maximization of capital efficiency** between TradFi and DeFi, while promoting tokenization and liquidity innovation for various real-world assets (RWA), including gaming assets and real estate.

Content

4. **Core Components**
   - Oracle Caster: RWA Tokenization
   - OIP: Oracle Interoperability Protocol
   - OSS: Oracle State Synchronizer
   - Drivers: CANTON Integration

5. **Implementation & Integration**
   - DeFi Protocol Integration
   - Traditional Finance Connection
   - Gaming RWA Integration

6. **Performance Analysis**
   - Scalability Metrics
   - Cost Analysis
   - Security Evaluation
   - Compliance Verification

7. **Future Development**
   - Research Directions
   - Roadmap
   - Ecosystem Expansion

8. **Conclusion**

1. Introduction

Background

As digital transformation accelerates for financial inclusion and capital efficiency innovation in capital markets, DLT has opened new frontiers in financial innovation. However, despite offering similar token dynamics, the 'silo issues' between TradFi tokenization solutions, as well as between TradFi and DeFi, remain unresolved, resulting in fragmented ecosystems. While TradFi manages massive capital based on strict regulatory compliance and stability, its closed structure limits innovation. Conversely, although DeFi demonstrates innovative financial products and high capital efficiency, **institutional investor participation remains restricted due to the absence of regulatory compliance mechanisms**.[1]

Problem Statement

The interoperability of tokenized assets in capital markets entails extensive technical and regulatory challenges. Financial regulatory authorities require strict regulatory compliance, such as AML and KYC, which cannot be exempted in on-chain environments.[2] To date, no solution has been presented that achieves efficient state synchronization while meeting these regulatory requirements. Current oracle systems demonstrate the following fundamental limitations:

1. **Limited State Synchronization**
   - Functionality confined to unidirectional data transfer or minimal bidirectional messaging
   - Inability to fully represent and control the state of complex financial products
   - Failure to ensure state consistency in cross-chain environments

2. **Absence of Regulatory Framework**
   - Lack of systematic requirements integration due to lack of regulatory research
   - Absence of mechanisms to verify regulatory compliance
   - Failure to meet core regulatory requirements including financial privacy and AML

3. **Inefficient Mechanisms with Limited Scalability**
   - Inefficient and high transaction costs due to monolithic architecture and per-call pricing policies
   - Limited data sources primarily focused on public data such as price and weather
   - Poor scaling strategy with limited expandability beyond set maximum capacity

Key Innovations

Our oracle system aims for groundbreaking innovations in terms of liquidity and capital efficiency, interoperability, security and decentralization, product diversity, cost improvement, and regulatory compliance. In particular, through the Oracle Interoperability Protocol (OIP) and Oraclizer, we can achieve liquidity innovation by providing a highly simplified deployment that enables DApps on all EVM chains to interoperate with TradFi's financial RWAs on-chain.

1. **Full Spectrum Interoperability**
   - Offchain to Offchain (even non-EVM): Inter-Domain interoperability through CANTON[1]
   - EVM to EVM: Inter-Domain interoperability between EVM chains
   - Offchain to EVM: Cross-Domain interoperability (oracle) through OIP and Oraclizer

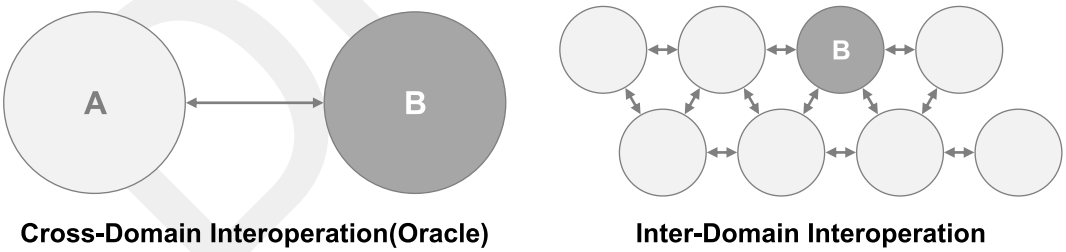

Figure 1. Interoperability models

2. **Realizing the Oracle Mauture-stage**
   - Initial Stage: Public data source oracle
   - Advanced Stage: Diverse data sources and programmable oracle
   - Mature Stage: Complete state synchronization contracts ensuring diverse data sources and regulatory compliance

1. DLT infrastructure for financial institutions, offering enhanced privacy and controls

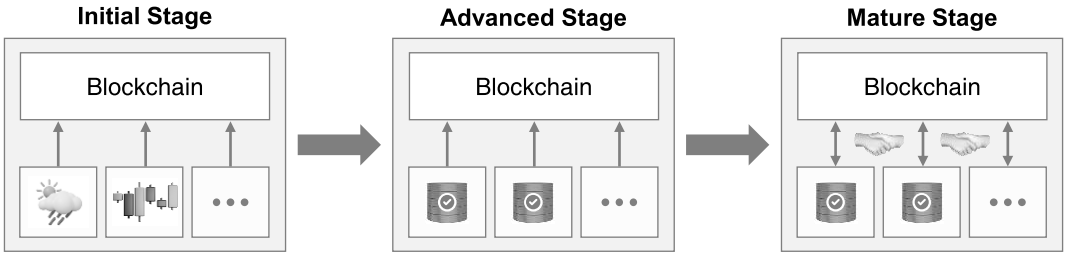

Figure 2. The Evolutionary Stage of Oracle

3. **Pursuit of Institutional-Grade System Reliability through Complete Regulatory Compliance**
   • **Finality**: Ensures financial contract completeness in Cross-Domain interoperability (enabling simultaneous settlement and clearing models like DvP[2])
   • **Offchain Privacy**: Integration of DLT solutions on a 'Need to know'[3] basis
   • **Onchain Privacy**: ZK proof method that does not partially expose transaction records, Abstraction (mixing) of KYC-verified IDs into pseudonymous Oracle Contract IDs (OCIDs)
   • **Enforceability**: Mitigation of financial risks through support of financial regulatory sanctions via contract languages with robust authority models (e.g., asset freezing, forced liquidation)
   • **Traceability**: Enabling 'System-wide AML' with OIP specification, DAML Party ID[4] and zk-basis-ID to enforce asset and (pseudo)identity across protocols

4. **Security and Decentralization**
   • Ensuring liveness through final state updates on L1
   • Stage1 rollup model through decentralized sequencer
   • High security margin via zk proofs

5. **Intuitive System Deployment: High Integration with Legacy DeFi**
   • Simple integration process with CANTON.network[5] and other RWA CANTON domains through contract invitation and minimal code modifications[3]
   • Legacy DeFi protocols on EVM chains can intuitively query and trade oracle-processed RWA states from external systems by simply referencing bridge contract addresses and calling standardized interfaces defined by OIP specifications

6. **Pioneering New RWA Markets**
   • Development of new financial products for various RWAs through bidirectional state contracts
   • Support for 'Tokenless Web3 Gaming' infrastructure for game assets where token issuance is constrained by application platform limitations
   • Risk diversification and new portfolio development through low-risk DeFi services

7. **Cost-Efficient Synchronization Method and Scaling Strategy**
   • Ensuring oracle data freshness through automated state synchronization with single-call cost
   • Over 93% reduction in gas fees per transaction through double compression of transaction batches via L3 and modular zk verification/DA layer

2. How to minimize settlement risk by taking delivery of securities at the same time as payment is made
3. How to increase privacy by only giving information to those who need it to do their job
4. An identifier that uniquely identifies a participant in a DAML smart contract. Indicates a party to the contract.
5. CANTON-based financial infrastructure designed for regulation and interoperability for financial firms; Goldman Sachs, Microsoft, and others are participating

System Overview

Oraclizer consists of a three-layer architecture comprising applications, protocols, and infrastructure.
Each layer ensures a scalable operational strategy through clear separation of responsibilities, providing dense vertical integration from in-house RWA tokenization solutions to cross-chain state synchronization.

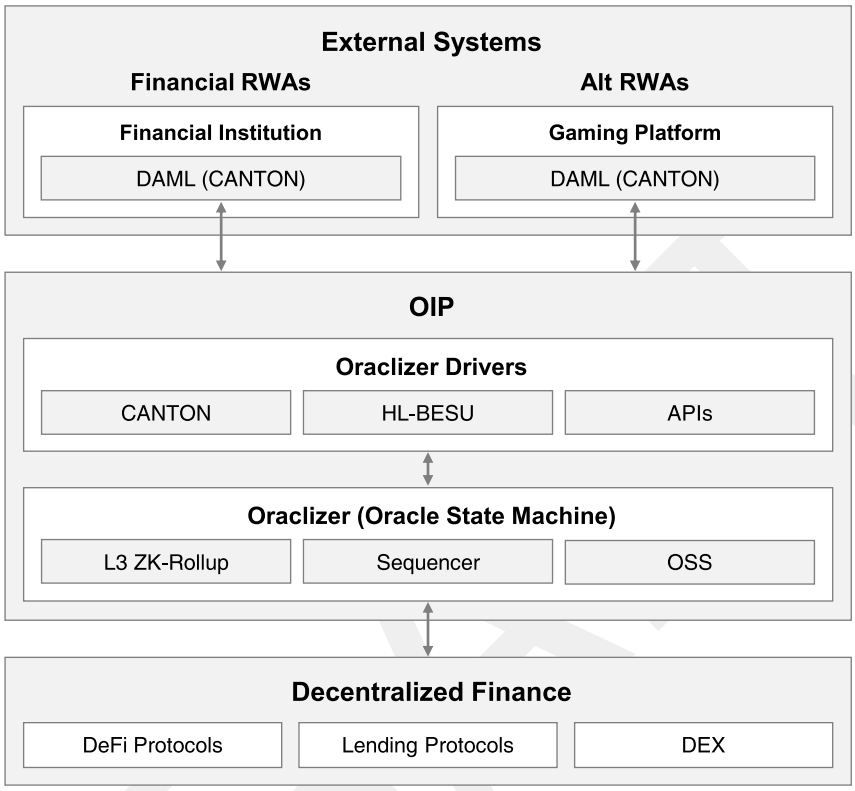

Figure 3. System Overview

1. **Application Layer:** The Application Layer processes tokenization of financial products and game assets through Oracle Caster, and provides KYC functionality with privacy protection through zk-basis ID. This layer, which directly interacts with end users, delivers user-friendly interfaces while ensuring regulatory compliance.

   - RWA tokenization (Oracle Caster)
   - zk-basis ID (KYC/Privado.ID)

2. **Protocol Layer:** OIP is both a data modeling framework and specification that inherits RCP to ensure regulatory compliance, enabling Cross-Domain interoperability between EVM and non-EVM systems as well as Inter-Domain interoperability between EVM chains. The Oracle State Synchronizer (OSS), operating on sequencer nodes elected by consensus of the decentralized sequencer, manages cross-chain state synchronization according to OIP specifications. This layer, implementing the system's core logic, ensures state consistency and synchronization efficiency.

   - Oracle Interoperability Protocol(OIP)
   - Oracle State Synchronizer(OSS)

3. **Infrastructure Layer:** The modular design is an advantageous strategy that enables sustainable integration of new blockchains and protocols while securing scaling strategies. The infrastructure providing the system's technical foundation consists of the following key components:

   - L3 zk-Rollup: Offsetting overheads from oracle state recording, zk proof generation, and decentralized sequencer
   - Validium (Aail): Data availability compression
   - Decentralized Sequencer: Stage 1 Rollup[6] for security
   - zk Verification Modular (zkVerify): Scalability and cost reduction through zk proof compression
   - RWA Registry Contract: RWA Registry and synchronized states
   - Cross-chain Bridge Interface: Inter-chain assets, states, and message interface
   - External System Integration Drivers: CANTON, Hyperledger Besu, APIs

2. Regulatory Compliance Protocol (RCP)

At this point where tokenization of capital markets is accelerating, regulatory compliance has become a necessity rather than an option.[4] RCP is the blockchain industry's first regulatory research paper published by the Oraclizer Core team. Through in-depth analysis of recommendations and financial product guidelines from 15 global financial regulatory authorities, it standardizes 31 key regulations that directly impact tokenized capital markets into five common properties.[5]

RCP	WB	ISDA	IOSCO	IMF	FSB	FATF	BIS	SFC	HKMA	EU	ESMA	FCA	MAS	FINMA	FINRA
(1)	✓		✓	✓	✓	✓	✓	✓	✓	✓				✓	
(2)		✓	✓	✓	✓	✓	✓	✓	✓	✓			✓	✓	✓
(3)						✓			✓	✓					✓
(4)									✓		✓				
(5)		✓	✓					✓				✓			
(6)			✓	✓	✓			✓		✓	✓	✓		✓	✓
(7)						✓			✓	✓			✓	✓	
(8)						✓									
(9)						✓									
(10)			✓	✓	✓	✓		✓		✓					✓
(11)		✓				✓		✓						✓	
(12)		✓	✓				✓								✓
(13)		✓	✓			✓		✓		✓					✓
(14)										✓					
(15)						✓		✓							✓
(16)			✓				✓								✓
(17)	✓	✓	✓			✓		✓	✓	✓			✓		✓
(18)	✓		✓			✓			✓		✓				
(19)										✓					
(20)			✓				✓		✓		✓				
(21)		✓	✓			✓	✓	✓	✓	✓		✓			✓
(22)									✓	✓					✓
(23)															✓
(24)											✓				✓
(25)			✓												
(26)						✓					✓				
(27)															
(28)			✓				✓								✓
(29)															
(30)			✓												
(31)															

Table 1: Recommendation and Guidance of Regulatory Authorities

(1) Customer Identity Verification (2) High-Risk/Suspicious Transaction Monitoring (3) Detection of Changes to Customer Identity Information (4) Contract Version Tracking (5) Exploration of Transaction History by Asset Type (6) External Audit (7) Setting Role-Based Permissions (8) Asset Freeze (9) Asset Recovery (10) Trading Restrictions (11) Transaction Limit (12) Cancellation or Modification of Transactions (13) Pausing of Trading (14) Suspension or Disposal of Smart Contract (kill switch) (15) Blacklist Management (16) Forced Liquidation (17) Privacy of Personal Information (18) Privacy of Financial Transactions (19) Code Security (20) Immutability of the Ledger (21) Finality of Transactions and Payments (22) Attaching Legal Documents (23) Token Expired Time (24) Token Transfer Restrictions (25) Issuance of Tokenized Cash (26) Issuance of Tokenized Securities (27) Controlling Transactions Involving Splitting Below Decimal Units (28) Token Burning (29) Gasless Support (30) Asset Class Management (31) Token Supply Control

6. Vitalik Buterin defines Stage 1 as the "Limited Training Wheels" phase, indicating that rollups are at an intermediate stage in their progression toward complete decentralization

financial market. This scenario demonstrates how RCP, unlike ERC-1400 and ERC-3643, effectively adheres to the recommendations and guidelines of regulatory authorities in the asset tokenization process.

In the financial technology domain, the RCP is pivotal for bond tokenization and lifecycle management. Utilizing Distributed Ledger Technology (DLT) and smart contracts, the RCP enforces compliance, ensures transparency, and secures operations. The following sections detail this process, including the critical role of legal counsel in notarization, which is integral to the legal and regulatory compliance checks.

The initial stage involves legal and regulatory compliance checks by issuers and their legal counsel, including notarization to ensure the authenticity and enforceability of the documents. This is represented as:

$$\Gamma_{\text{prep}} = \sum_{\omega \in \Omega} \rho(\omega, \lambda_{\text{legal}}, \sigma_{\text{RCP}}, \nu_{\text{notarization}})$$

where Γ_{prep} indicates preparatory operations, Ω the set of requirements, ρ the compliance function, λ_{legal} legal advisories, σ_{RCP} the RCP's compliance mechanisms, and $\nu_{\text{notarization}}$ the notarization process by legal counsel.

The next step, tokenization and issuance, involves using smart contracts to either create Tokenized Cash (FT) or Securities Ω_{trade} represents trading and compliance operations, η_{RCP} the RCP's function, μ_{trade} trade requests, $\nu_{\text{compliance}}$ compliance checks, and $\nu_{\text{notarization}}$ the notarization of compliance documents.

The process concludes with maturity and settlement, where assets are transferred following gasless settlements, and notarization ensures the legal validity of the settlement documents and agreements:

$$\Xi_{\text{settle}} = \zeta_{\text{RCP}}(\alpha_{\text{maturity}}, \beta_{\text{settlement}}, \nu_{\text{notarization}})$$

Ξ_{settle} indicates maturity and settlement operations, ζ_{RCP} the settlement function, α_{maturity} maturity checks, $\beta_{\text{settlement}}$ settlement executions, and $\nu_{\text{notarization}}$ the notarization process ensuring the legal validity of settlement documents and agreements.

In the **Preparation Phase**, establishing a robust framework of legal and regulatory compliance is paramount. The formulation is:

$$\Upsilon_{\text{prep}} = \bigcup_{\lambda \in \Lambda} \sigma(\lambda) \times \bigcap_{\delta \in \Delta} \varphi(\delta)$$

Υ_{prep} symbolizes preparatory operations, Λ represents legal advisories, σ maps legal advisories to their compliance metrics, Δ is the set of regulatory requirements, and φ verifies compliance for each requirement. This captures the alignment of legal advisories with regulatory requirements.

Figure 4. RCP Paper written by Oraclizer Core Team

Importance of RCP

Asset interoperability in capital markets extends beyond mere technical challenges. Regulatory compliance becomes the core standard of value in this process, and RCP provides clear coordinates for such value judgments.
Any interoperability technology that does not comply with RCP, namely the regulatory requirements of global financial regulatory authorities, is like navigation without a compass. Particularly in situations requiring financial regulatory oversight and intervention, **if there is no regulatory framework**, any interoperability and oracle technology is essentially nonexistent. To that extent, it can **nullify the entire system's existential value.**

OIP succeeds RCP

RCP, established through thorough regulatory research, is not merely a set of regulations but represents the essence of service that ensures the complete value of capital liquidity technology. OIP, which inherits the non-functional requirements derived from the five regulatory properties standardized in RCP, guarantees the following core values:

1. **Finality (Completeness)**
   - zk proof-based state synchronization: Ensuring transaction atomicity
   - OSS: Maintaining state consistency across oracle and cross-chain operations
   - DAML[7] contract: Guaranteeing legal enforceability through abstraction of rights and obligations

7. Digital Asset Modeling Language, Smart contract programming language specialized for modeling financial workflows

2. **Traceability**
   - System-wide AML: Identity integration and management at protocol level
   - Cross-chain Registry: Tracking asset states and transfer paths
   - OSS Event Monitoring: Event sourcing and management of complete state change history
   - Audit Trail System: Audit tracking functionality for regulatory authorities

3. **Confidentiality**
   - 'Need to know' basis ledger model
   - zk proof-based privacy compliance
   - Selective information disclosure mechanism
   - Encrypted state synchronization

4. **Enforceability**
   - DAML's robust authority model
   - Asset freezing capability for regulatory authorities
   - Forced liquidation mechanism
   - Automatic sanction execution upon regulatory violations

5. **Tokenizability**
   - Oracle Caster: RWA tokenization solution
   - ERC-3770: Cross-chain addressing scheme
   - Cross-chain bridge interface: Standardization of inter-chain asset transfers
   - RWA Registry: Standardized management of tokenized assets

Regulatory Enforcement Mechanisms

The regulatory enforcement mechanism is the ultimate purpose and core of regulatory compliance. While completeness, traceability, and confidentiality provide the foundation for regulatory compliance at the infrastructure level, enforceability is the key element that actualizes regulatory effectiveness on this foundation, and must be implemented across both infrastructure and application layers. If effective supervision and intervention by regulatory authorities cannot be guaranteed, all other regulatory compliance efforts inevitably lose half their significance.

Through RCP research, the Oraclizer Core team discovered that ERC-1400[6] and ERC-3643[7], the widely used security token standards, do not fully meet the requirements of modern financial regulations. They show particularly significant limitations in implementing system-wide sanction mechanisms. Overcoming these limitations requires a new Ethereum Improvement Proposal (EIP), and the Oraclizer team is currently preparing a new security token standard that enables more comprehensive regulatory compliance.

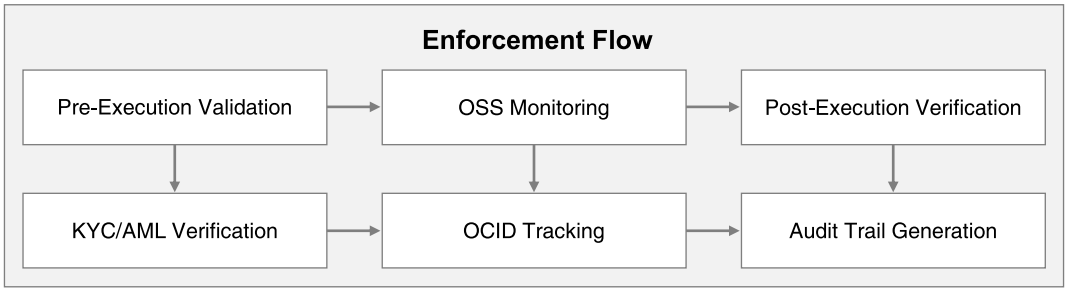

Figure 5. OIP Enforcement Mechanism

3. System Architecture

One of blockchain's greatest challenges is the reliable integration of external data.[8] Blockchain inherently guarantees strong integrity of internal states through consensus among network participants. However, the 'oracle problem' - safely reflecting external world data onto the blockchain - still lacks a complete solution.

Oracles are not mere data transmitters but core infrastructure that extends blockchain's trust boundary to the external world. This entails two fundamental challenges. First, the authenticity of oracle data must be guaranteed[9], and second, oracle data must maintain synchronization and consistency with the blockchain state.

Existing oracle solutions have only partially addressed these challenges. Most approaches focused on unidirectional data transmission, resulting in high costs and inefficiencies due to structures requiring continuous data calls and verification. Moreover, **despite existing oracles' exposure to off-chain regulatory environments, higher-order requirements such as regulatory compliance and complete state synchronization were rarely considered.**

For blockchain technology to achieve truly meaningful financial innovation, oracles must evolve beyond simple data bridges. The requirements of modern financial systems, including TradFi and DeFi integration, financial RWA tokenization, and regulatory compliance, demand a new dimension of oracle architecture.
This necessitates the following key characteristics:

1. **State Consistency:** Bidirectional state synchronization between blockchain and external systems to ensure legal completeness of contracts
2. **Regulatory Compliance:** Systematic assurance of financial regulatory requirements
3. **Scalability:** Performance capacity to process large-scale financial transactions
4. **Economic Efficiency:** Sustainable Gas fee structure

Oraclizer is a new oracle paradigm designed to realize this vision. Its oracle state machine architecture, based on zk proofs, fundamentally resolves the limitations of existing oracles while comprehensively delivering the characteristics demanded by modern financial systems. In particular, OIP, which inherits RCP's regulatory compliance framework, provides the essential foundation for blockchain technology's entry into institutional frameworks.

Oracle State Machine

The oracle state machine is a system that implements state synchronization between off-chain and on-chain systems. Unlike existing oracles that simply transmit data, Oraclizer guarantees bidirectional state synchronization and liveness by tracking all state change histories off-chain, mathematically proving the validity of these transitions, and finalizing updates to L1.

1. **State Capture**
   - Integration of DAML Party ID, Contract ID, and ERC-3770[10] addr as leaf node states in Sparse Merkle Tree (SMT)[8]
   - Detection of state changes by decentralized sequencer through DAML event sourcing or external DLT contracts
   - Collection of metadata for state changes
   - Identification of regulatory compliance requirements such as KYC

2. **State Transition Verification**
   - Verification of transition rule compliance through OIP
   - Validation of regulatory restrictions
   - Business logic validation

3. **State Proof Generation**
   - Generation of zk proofs for state verification after state changes
   - Ensuring integrity of state changes

4. **State Synchronization**
   - Submission of changed state proofs and summary information as L1 to achieve finality
   - Update of cross-chain state consistency after OSS verification of L1 finality
   - Rollback mechanism for regulatory enforcement actions such as cancellation and liquidation

L3 zk-Rollup Infrastructure

The L3 structure, serving as the core infrastructure of the oracle state machine, compensates for the scalability degradation inherent in the paradoxical adoption of consensus algorithms in rollup structures for maintaining decentralization, while securing additional scaling strategies for oracle state synchronization processing through modular architecture.

1. **Scalability**
   - Compensating for performance loss from decentralized sequencer implementation through additional zk proof compression
   - Improving throughput via off-chain data availability layer and computational offloading of zk verification through zkVerify[9]

8. A cryptographic data structure used to efficiently store and prove large key-value mappings.
9. Horizen Labs' zk Proof Verification Network (zkVerify.io). Reduce zk verification costs and simplify complexity

2. **Security**
   - Ensuring reliability of state transitions through multi-layered zk proof structure
   - Guaranteeing data availability through Avail's KZG polynomial commitment
   - Enhancing security through independent verification layer of zkVerify

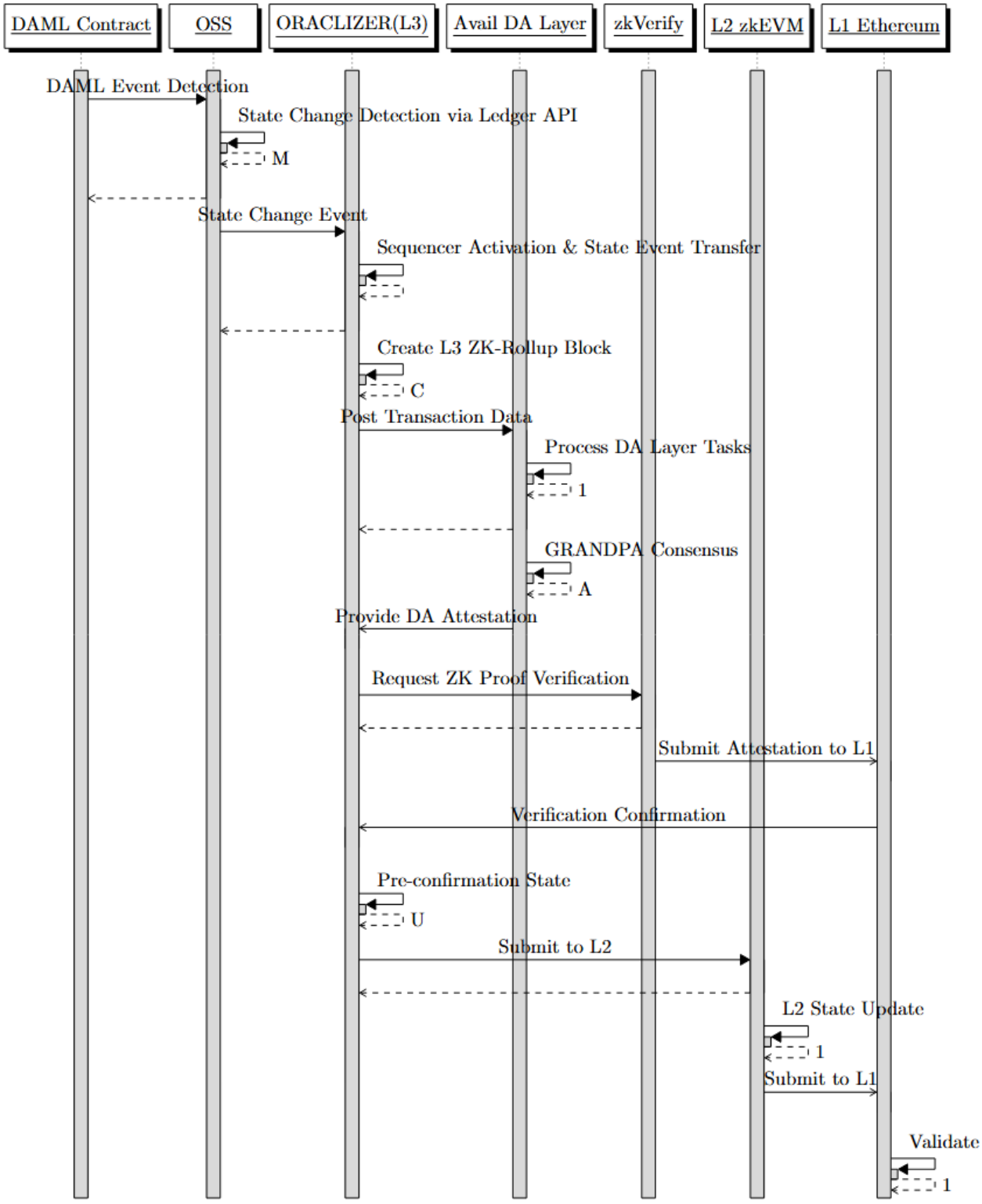

Figure 6. Oraclizer(L3) Core Flow

3. **Cost Efficiency**
   - Dramatic reduction in zk proof verification costs (zkVerify)
   - Optimization of data availability costs (Validium/Avail)
   - Up to 93% reduction in Gas fees

Decentralized Sequencer Design

Ensuring immutability in inter-layer state propagation is a critical design principle directly linked to contract finality. Particularly in higher-order layers like L3, a centralized sequencer would compromise system reliability. Since liveness can be guaranteed through L1, adopting high security margin consensus algorithms in decentralized sequencer mechanisms creates additional overhead issues. To achieve Stage 1 rollup, Oraclizer implements the 'D-quencer' consensus algorithm with BFT properties, utilizing equal-stake POS to reduce weight calculation overhead and selecting Active Asserter from the Standby Asserter group through VRF based on Boneh-Lynn-Shacham (BLS) signatures, followed by consensus through multisig of identical signatures.

$$\varphi_{32}: StateL3 \rightarrow StateL2\ \varphi_{32}(s) = (h(s), \pi_{32}, metadata)$$

$$\forall s \in StateL3: Verify(\varphi_{32}(s)) \rightarrow true$$

$$\varphi_{21}: StateL2 \rightarrow StateL1\ \varphi_{21}(s) = (h(s), \pi_{21}, metadata)$$

$$\Phi = \varphi_{21} \circ \varphi_{32}$$

** h= state hash*

** π_{32}=zk-Proof(state transitions)*

Figure 7. Cross-Layer Oracle State Propagation

1. **Decentralized Sequencer Configuration and Staking**
   - Comprising 10-100 nodes in D-quencer consensus, considering maximum fault tolerance and marginal utility
   - Eliminating weight calculations by staking equal assets (reducing overhead)
   - Ensuring fair election opportunities for Standby Asserter nodes
   - Preventing malicious actions through multisig-based mechanisms (with slashing implementation)

2. **Sequencer Election Mechanism**
   - VRF output values from each node → candidate pool election → election of node with minimum VRF output
   - Elected sequencer operates for a predetermined epoch

3. **Core Functions of Active Asserter**
   - Transaction ordering based on nonce
   - Requesting zk proof generation
   - Communication and verification with each modular (validating state transition validity)
   - State synchronization through OSS and prevention of oracle double-spending
   - Cross-chain messaging

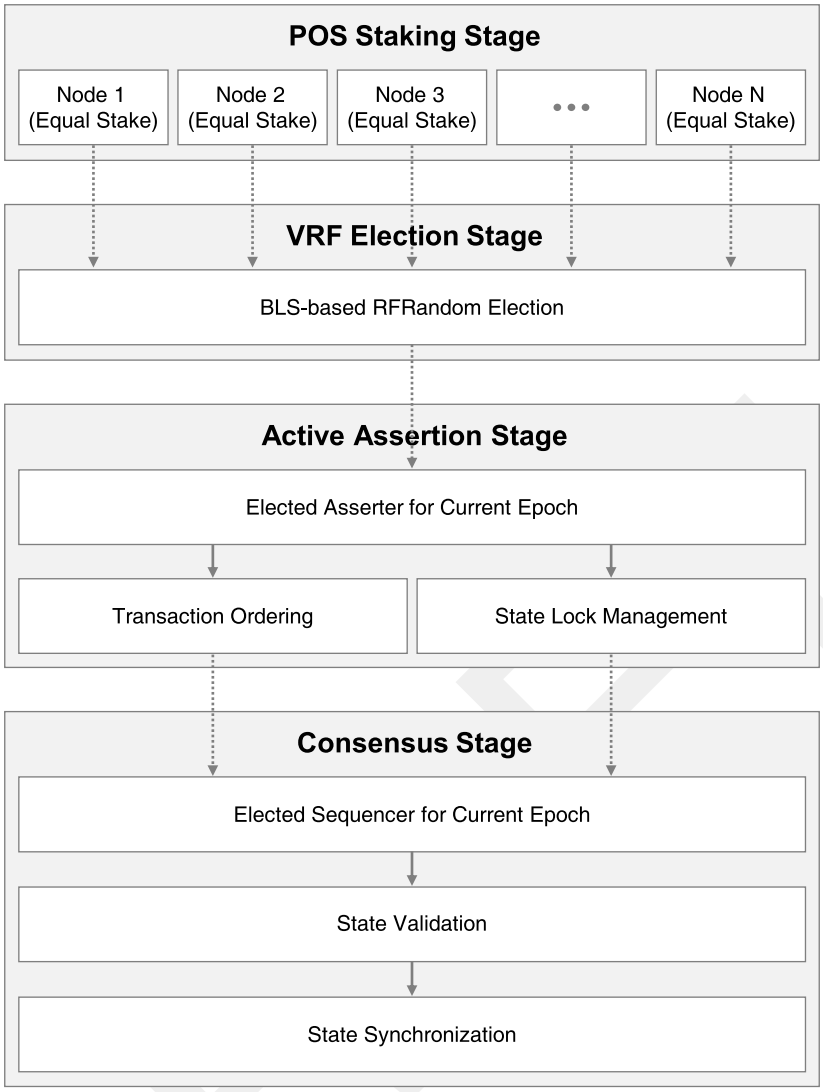

Figure 8. D-quencer Consensus

Cross-Chain Communication

Cross-chain communication managed by OSS enables secure and efficient message delivery and state synchronization between various blockchain networks and DAML-based external systems. Furthermore, it manages consistent accounts (users and contracts) across chains by supporting integrated address scheme (ERC-3770) conversion in cross-chain environments.

1. **ERC-3770 Integration**
   Oraclizer adopts the ERC-3770 address scheme to simplify fundamental address management issues in cross-chain communication by resolving cross-chain complexities in RWA Registry contracts where EVM chain DApps can intuitively contract in on-chain states, and by minimizing potential errors in state management. It standardizes and manages inter-chain address schemes within OSS, while maintaining the OIP specification unchanged to ensure compatibility with existing address schemes.

2. **Bridge Transitions**
   - Ensuring atomicity and rollback of asset transfers: Lock→Verify→Execute
   - Cross-chain state synchronization
   - State transition occurs only when state changes are detected after state hash calculation of chain-specific contracts (categorized into temporary/final phases)
   - Recovery procedures

3. **Bridge Verification**
   - Verification of bridge contracts for each chain
   - Validation of regulatory compliance requirements
   - Double-spending prevention verification

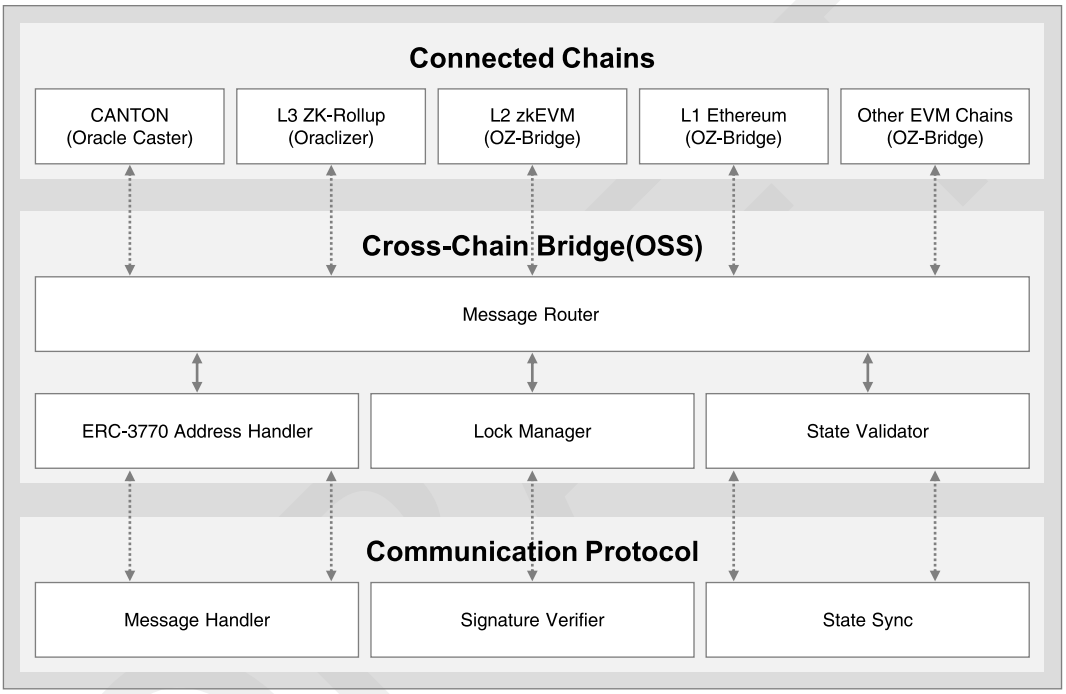

Figure 9. OSS-driven Cross-Chain Communication

4. System Architecture

Oracle Caster: RWA Tokenization

The ability to **fully comply with financial regulations**, which have the strictest regulatory environment, **implies the capability to meet regulatory requirements of any other industry** such as gaming, real estate, and logistics.[11] DAML, a Haskell-based high-order programming financial contract language used in the "U.S. Digital Dollar Project," supports lightweight formal verification for mathematical validation and provides complete privacy up to GDPR's 'right to be forgotten' through its strict ledger model, need-to-know basis 2-Phase Commit sub-transactions, and ledger projection mechanisms. Additionally, despite being inter-party (signatories) contracts, it can provide regulatory authorities with audit frameworks and controllable robust authority models, enabling secure implementation of most financial regulatory compliance properties proposed in our RCP paper from the system core.
Oracle Caster, this DAML-based RWA tokenization solution, thus establishes a regulatory-compliant foundation capable of tokenizing any form of RWA, from financial assets to real estate and gaming assets.

Even while operating in a closed on-premise environment, Oracle Caster enables transactions with on-chain DApps through integration with Oraclizer CANTON Driver, allowing tokenized assets to interface with Oraclizer's state synchronization mechanism. **The capital efficiency of RWA traded on Oracle Caster can be maximized by combining it with the highly liquid DeFi market through just a few lines of code modification in CANTON.**

OIP (Oracle Interoperability Protocol)

OIP, which simultaneously achieves complete asset mobility, traceability, and transaction party privacy, serves as Oraclizer's central nervous system, governing oracle contract atomic state synchronization, cross-chain bridges, and RWA Registry synchronization. As a protocol managing interoperability between EVM, non-EVM, and cross-chain systems, it has the following structure.
The OCID in OIP specifications is a unique identifier combining DAML party ID and on-chain KYC ID (e.g., Privado.ID), generated for each oracle contract. This represents a pseudonymous ID that ensures traceability while maintaining privacy of contract parties' identities.

OIP SPEC. v0.1.0

```
{
    "version": "0.1.0",
    "spec": {
        "ocid": {
            "damlPartyId": "string",     // Unique identifier for DAML contract participant
            "zkId": "string"        // Privacy-preserving ZK identity identifier
        },

        "address": {
            "chainId": "string",         // Target chain network identifier
            "addr": "string"             // Chain-specific address in ERC-3770 format
        },

        "asset": {
            "assetType": "enum",         // Asset classification (ERC-20, BOND, STOCK, etc.)
            "balance": "number",         // Current asset balance
            "metadata": "object",        // Additional asset-specific information
            "lockStatus": "enum",        // Asset state (UNLOCKED, TEMP_LOCKED, PERM_LOCKED,
                                            REGULATORY_LOCKED)
            "lockExpiration": "timestamp",// Expiration time for temporary locks
            "regulatoryAuthority": {     // Regulatory action details
                "id": "string",          // Authority identifier
                "orderDetails": "object" // Detailed regulatory order information
            }
        },

        "contract": {
            "contractId": "string",      // Unique contract identifier
            "contractStatus": "enum",    // Contract state (PENDING, ACTIVE, COMPLETED, etc.)
            "contractLockStatus": "enum", // Lock state of contract
            "participants": ["ocid"],    // Array of participating OCIDs
            "regulatoryAction": "object" // Associated regulatory actions
        },
```

```
"kycStatus": {
            "isVerified": "boolean",     // Verification status
            "lastVerificationTimestamp": "timestamp", // Last verification time
            "kycLevel": "enum"           // Verification level (BASIC, ADVANCED, PREMIUM)
        },

        "message": {
            "ocid": "object",            // Oracle Contract ID reference
            "asset": "object",           // Asset information
            "contract": "object",        // Contract details
            "kycStatus": "object",       // KYC verification status
            "timestamp": "timestamp",    // Message creation time
            "signature": "string",       // Cryptographic signature
            "sourceChainId": "string",   // Origin chain identifier
            "oracleNodeId": "string",    // Processing oracle node identifier
            "messageType": "enum",       // Type of message (ASSET_UPDATE, CONTRACT_UPDATE,
etc.)
            "regulatoryInfo": "object",  // Additional regulatory information
            "extraData": "object"        // Extension fields
        },

        "regulatoryAction": {
            "actionType": "enum",        // Action type (FREEZE, SEIZE, CONFISCATE)
            "authority": "object",       // Acting regulatory authority
            "orderDetails": "object",    // Detailed order information
            "actionTimestamp": "timestamp", // Action execution time
            "affectedAssets": ["assetId"], // List of affected asset identifiers
            "affectedContracts": ["contractId"], // List of affected contract identifiers
            "complianceStatus": "enum"   // Current compliance status
        }
    }
}
```

OIP normalizes cross-chain address management using the ERC-3770 scheme while maintaining compatibility with existing address systems. External interfaces use chain-specific base address formats, while internally OIP converts them to ERC-3770 format (chainId:address) through its Address Handler. The Oraclizer team is preparing a new EIP proposal to implement regulatory enforcement mechanisms (FREEZE, SEIZE, CONFISCATE, among others) in EVM.

NEW-EIP (draft)

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.19;

/**
* @title IRWARegistry
* @notice Central registry interface for managing regulatory compliance and enforcement of
Real World Assets (RWA)
* @dev This registry acts as the single source of truth for asset compliance status
*/
interface IRWARegistry {
   // ... [previous enum, event definitions]
```

```
/**
* @title IRWARegistry
* @notice Central registry interface for managing regulatory compliance and enforcement of
Real World Assets (RWA)
* @dev This registry acts as the single source of truth for asset compliance status
*/
interface IRWARegistry {
   // ... [previous enum, event definitions]

   /**
    * @notice Verifies if an asset is compliant and eligible for transaction
    * @dev This function must be called before any asset transaction
    * @param assetId Unique identifier of the asset to check
    * @return status Boolean indicating if the asset is compliant
    * @return restrictions Encoded restrictions if any are active
    */
   function checkAssetCompliance(bytes32 assetId)
       external
       view
       returns (bool status, bytes32 restrictions);
}

/**
* @title IRWAEnforceable
* @notice Interface that must be implemented by any contract handling RWA transactions
* @dev Provides hooks for regulatory compliance and emergency actions
*/
interface IRWAEnforceable {
   /**
    * @notice Pre-transaction hook for regulatory compliance
    * @dev Called before any asset transaction to ensure compliance
    * @param assetId Identifier of the asset being transacted
    * @param action Function selector of the action being performed
    * @return success Boolean indicating if the transaction can proceed
    */
   function beforeAssetTransaction(bytes32 assetId, bytes4 action)
       external
       returns (bool success);
}

/**
* @title RWARegistry
* @notice Implementation of the central RWA registry with regulatory controls
* @dev Manages asset compliance status and enforces regulatory actions across the system
*/
contract RWARegistry is IRWARegistry {
   // Mappings for system state management
   mapping(address => bool) private compliantContracts;    // Tracks verified compliant
contracts
   mapping(bytes32 => address) private assetControllers;   // Maps assets to their
controller contracts
```

```
    /**
     * @notice Ensures asset is not under regulatory restriction
     * @dev Reverts if asset is frozen, seized, or confiscated
     * @param assetId Asset to check
     */
     modifier checkRegulatory(bytes32 assetId) {
        require(
            assets[assetId].status != AssetStatus.FROZEN &&
            assets[assetId].status != AssetStatus.SEIZED &&
            assets[assetId].status != AssetStatus.CONFISCATED,
            "Asset restricted by regulatory action"
        );
        _;
    }

    /**
     * @notice Registers a contract as RWA compliant
     * @dev Only callable by authorized regulators
     * @param contractAddress Address of the contract to register
     */
    function registerCompliantContract(address contractAddress)
        external
        onlyRegulator
    {
        compliantContracts[contractAddress] = true;
        emit ContractRegistered(contractAddress, msg.sender);
    }

    /**
     * @notice Immediately freezes an asset and stops ongoing transactions
     * @dev Triggers emergency hooks in all associated contracts
     * @param assetId Identifier of the asset to freeze
     */
    function emergencyFreeze(bytes32 assetId)
        external
        onlyRegulator
    {
        RWAsset storage asset = assets[assetId];
        require(asset.status == AssetStatus.ACTIVE, "Asset not in active state");

        // Update asset status
        asset.status = AssetStatus.FROZEN;

        // Notify asset controller contract
        if (assetControllers[assetId] != address(0)) {
            bool success = IRWAEnforceable(assetControllers[assetId])
                .beforeAssetTransaction(assetId, this.emergencyFreeze.selector);
            require(success, "Emergency freeze action failed");
        }

        emit EmergencyFreeze(assetId, msg.sender, block.timestamp);
    }
}
```

```
/**
* @title RWACompliantToken
* @notice Example implementation of a regulatory compliant token contract
* @dev Demonstrates integration with RWA Registry for regulatory compliance
*/
contract RWACompliantToken is IRWAEnforceable {
   IRWARegistry public immutable registry;

   // Tracks pending transactions for each asset
   mapping(bytes32 => mapping(uint256 => Transaction)) private pendingTransactions;
   uint256 private transactionCounter;

   struct Transaction {
       address from;
       address to;
       uint256 amount;
       uint256 timestamp;
       bool active;
   }

   /**
    * @notice Ensures asset compliance before transaction
    * @dev Queries registry for current compliance status
    * @param assetId Asset to check
    */
   modifier checkRegistry(bytes32 assetId) {
       (bool compliant, bytes32 restrictions) = registry.checkAssetCompliance(assetId);
       require(compliant, "Asset not compliant");
       if (restrictions != bytes32(0)) {
           _handleRestrictions(assetId, restrictions);
       }
   }

   /**
    * @notice Initiates a compliant token transfer
    * @dev Checks compliance and handles regulatory restrictions
    * @param to Recipient address
    * @param amount Amount to transfer
    * @param assetId Identifier of the asset being transferred
    */
   function transfer(address to, uint256 amount, bytes32 assetId)
       external
       checkRegistry(assetId)
   {
       // Create pending transaction
       uint256 txId = _createPendingTransaction(msg.sender, to, amount, assetId);

       // Perform transfer if no restrictions
       _executeTransfer(txId, assetId);
   }
```

```
    /**
     * @notice Handles regulatory actions and emergency stops
     * @dev Implements IRWAEnforceable interface
     * @param assetId Asset being affected
     * @param action Regulatory action being performed
     * @return success Whether the action was handled successfully
     */
    function beforeAssetTransaction(bytes32 assetId, bytes4 action)
        external
        override
        returns (bool success)
    {
        require(msg.sender == address(registry), "Unauthorized regulatory action");

        if (action == RWARegistry.emergencyFreeze.selector) {
            _pauseAssetTransactions(assetId);
            emit AssetFrozen(assetId, block.timestamp);
            return true;
        }

        return false;
    }

    // Internal implementation functions...
}
```

OSS (Oracle State Synchronizer)

While traditional oracles merely transmitted external public data to blockchain, OSS achieves state consistency by managing OCID-based bidirectional state change tracking and updates within Oraclizer, operated by D-quencer consensus. The state roots updated from both directions are mathematically guaranteed for state and transition validity through zk proofs.[12]

Notably, OSS implements a Preemptive Lock mechanism to prevent duplicate oracle contracts (double-spending) that could occur due to network latency in cross-chain environments. For instance, when multiple DeFi protocols across EVM chains simultaneously request oracle contracts for treasury bonds identified by the same asset ID in Oraclizer's RWA Registry contract, OSS immediately locks that contract. This lock state persists until the ongoing oracle contract is either completed or cancelled, fundamentally preventing duplicate oracle contracts for the same asset in cross-chain environments.

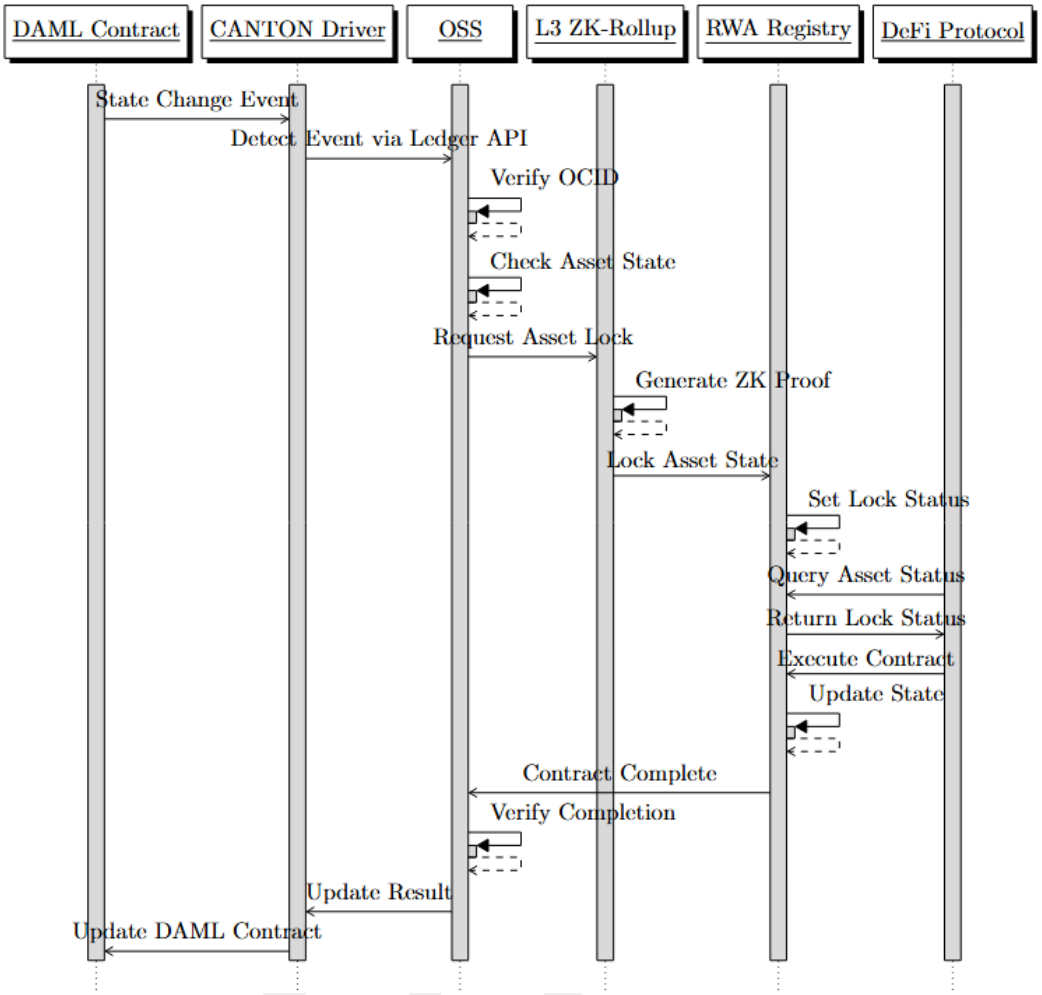

Figure 10. OSS Sequence Flow and Contract Lock(abstract)

Drivers: CANTON Integration

Through DAML's *Observer* and *Signatory* model, the Oraclizer CANTON Driver selectively observes and synchronizes states only for contracts granted oracle execution authority by oracle requests. For example, in the process of expanding liquidity for tokenized U.S. Treasury bonds, financial institutions can invite the Oraclizer party of Oraclizer-Finance (an Oraclizer CANTON domain handling financial RWA) as a *Signatory* to this contract. Only authorized assets and their state changes are propagated to the Oraclizer network through OSS and OIP.
Treasury bond tokens in ongoing DAML contracts remain locked until the DeFi contract on the EVM chain is completed. The Oraclizer Drivers plan to extend this architecture to other enterprise DLTs, including Hyperledger Besu, enabling more financial institutions to maximize capital efficiency through integration with DeFi protocols while maintaining regulatory compliance through simple deployment.

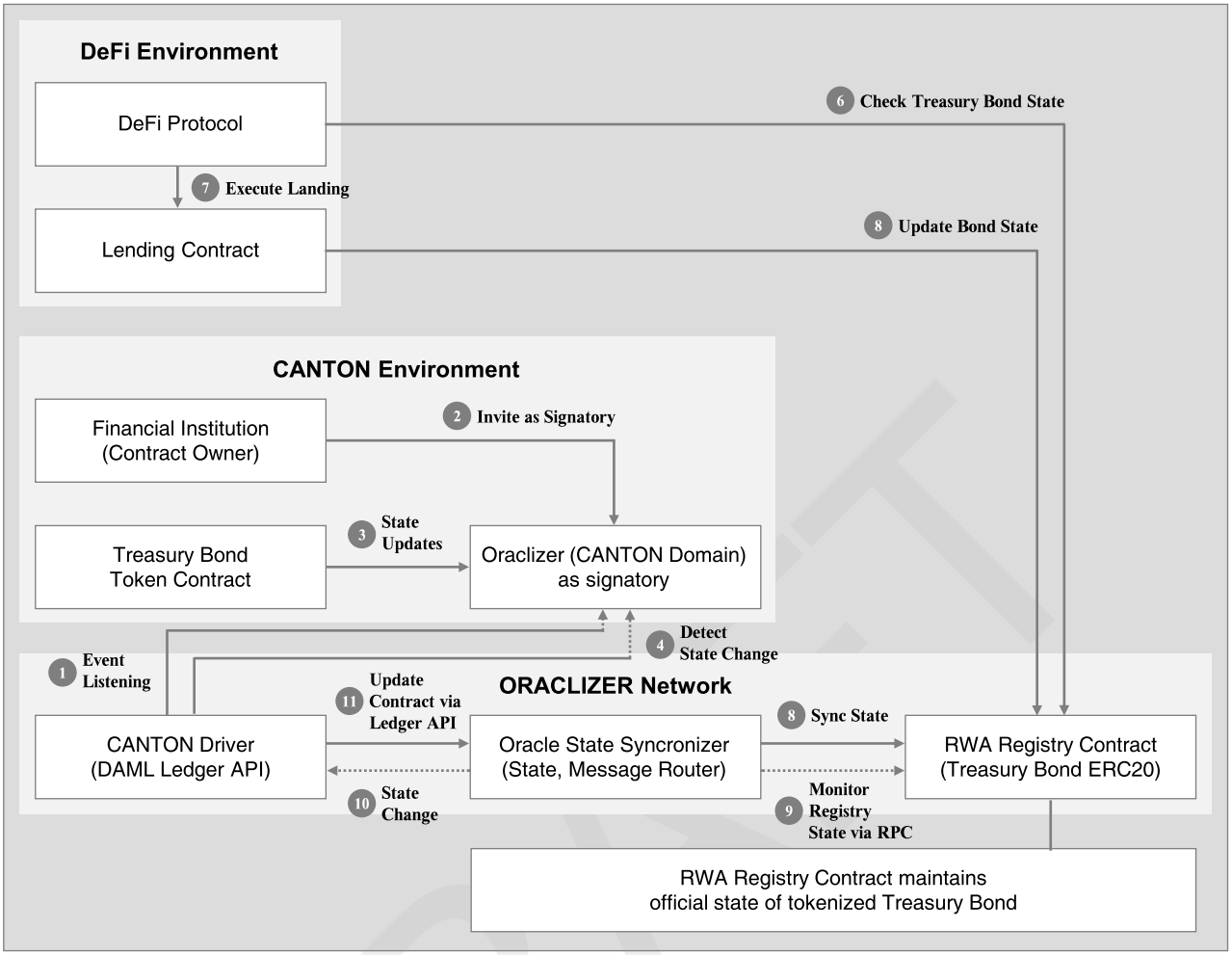

Figure 11. Oracle Contract Mechanism via CANTON Driver

5. System Architecture

Traditional oracle systems have shown limitations in solving cost issues due to per-call fee structures and fundamental network scalability constraints. To overcome these limitations, Oraclizer introduces innovative scaling strategies including L3 architecture, DA modular, and zkVerify integration, enabling **automated state synchronization without additional costs after initial oracle calls**, resulting in logarithmic reduction of Gas fees compared to existing oracles. This dramatic cost reduction goes beyond mere efficiency improvements, forming the foundation for a new paradigm of on-chain state synchronization that was previously impossible. In other words, **on-chain state synchronization implementation is fundamentally impossible without Gas fee reduction.** Through these practically feasible Gas fees, Oraclizer can achieve comprehensive state synchronization across DeFi protocols, traditional finance, and the gaming industry.

DeFi Protocol Integration

Integration with EVM chain DeFi protocols is implemented through an architecture based on RWA Registry and bridge contracts. The RWA Registry contract on Oraclizer(L3) serves as the Single Source of Truth (SSOT) for oracles RWAs, recording current states and ownership (pseudonymous identities) from external systems, and including contract locking mechanisms to prevent double-spending across chains. This SSOT ensures reliability and liveness for EVM chain DeFi protocols by synchronizing states from immutable external ledgers through OSS to final L1 records, enabling verification and trading of consistent, updated RWA types, balances, identities, and contract states.

Oraclizer's bridge contracts deployed on each EVM chain interface between that chain's DeFi protocols and the RWA Registry contract. These bridge contracts relay requests from DeFi to the Registry and subscribe to oracle RWA state changes for immediate DeFi updates. For example, when a lending protocol on Arbitrum wants to use tokenized treasury bonds as collateral, the bridge contract performs the following tasks:

1. **Registry Query:** Verify current state, validity, and valuation information of the treasury bonds
2. **State Lock Request:** Request asset locking when executing loan contracts
3. **State Subscription:** Monitor and relay Registry state changes
   (e.g., collateral value changes, regulatory actions) to DeFi
4. **Contract Completion Notification:** Update final state to Registry upon loan contract completion

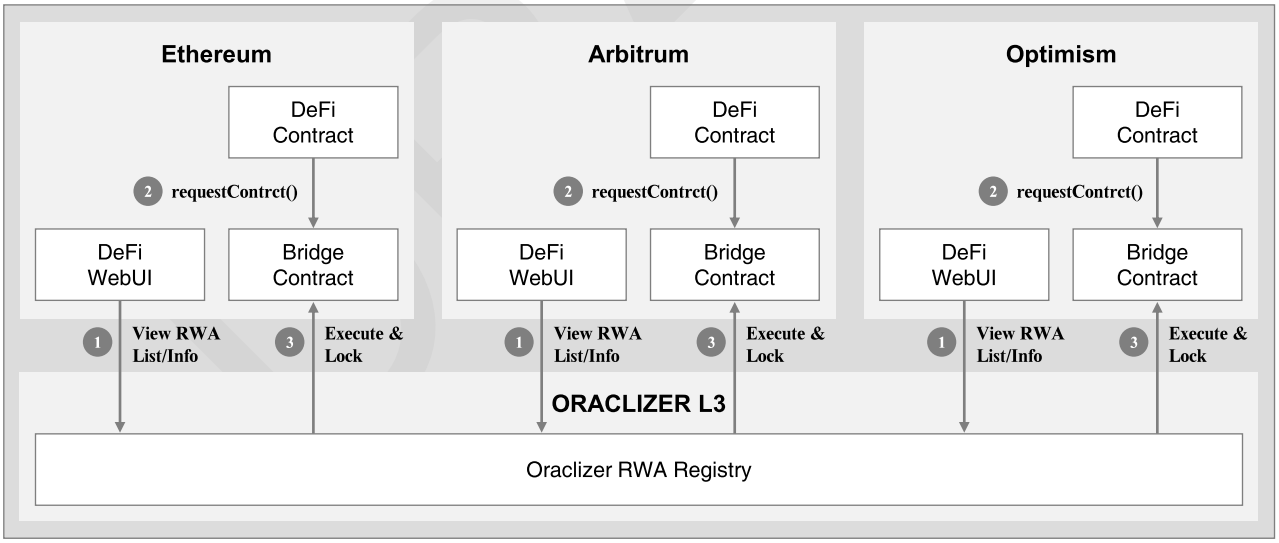

Figure 12. DeFi Protocol Integration

Through this simple deployment structure, DeFi protocol developers can integrate oracle RWAs from external systems with DeFi by simply referencing bridge contract addresses and calling standardized interfaces defined by OIP specifications, without needing to implement cross-chain logic directly. This significantly reduces the integration barrier for DeFi protocols with Oraclizer, enabling new financial business opportunities such as developing low-risk products based on oracle state synchronization contracts with minimal changes to legacy protocols.

Traditional Finance Connection

Leveraging the cross-domain interoperability embedded within CANTON, we efficiently address the silo challenges of tokenized RWA in traditional financial institutions. Each financial institution operates within its own CANTON domain and can bridge assets on-chain through the Oraclizer-Finance CANTON domain. This architecture transcends mere system integration, **providing a framework that selectively maximizes asset liquidity while ensuring institutional privacy and regulatory compliance**.
For instance, when Bank A on CANTON.network seeks to utilize its government bonds as collateral in Arbitrum's DeFi protocols through Oraclizer, the implementation can be initiated with the following DAML code:

DAML

```
module FinanceIntegration where

import DA.Date

template TreasuryBond
  with
    issuer: Party
    owner: Party
    bondId: Text
    amount: Decimal
    maturity: Date
    observers: [Party]
  where
    signatory issuer, owner
    observer observers

    choice TransferToOraclizer : ContractId TreasuryBond
      with
        newOwner: Party  -- Oraclizer
      controller owner
      do
        create TreasuryBond with
          issuer = issuer
          owner = newOwner
          bondId = bondId
          amount = amount
          maturity = maturity
          observers = owner :: observers

    choice UpdateObservers : ContractId TreasuryBond
      with
        newObservers: [Party]
      controller owner
      do
        create TreasuryBond with
          observers = newObservers

template OraclizerIntegrationRequest
  with
    requester: Party
    oraclizer: Party
```

```
    bond: ContractId TreasuryBond
  where
    signatory requester
    observer oraclizer

    choice Accept : ContractId TreasuryBond
      controller oraclizer
      do
        exercise bond TransferToOraclizer with
          newOwner = oraclizer

    choice Reject : ()
      controller oraclizer
      do
        pure ()
```

CANTON configuration and routing(bash):

```
canton {
  domains {
    bankA {
      domain-id = "BankA.canton.domain"
      public-api {
        port = 5018
        address = "localhost"
      }
    }
    oraclizer-finance {
      domain-id = "Oraclizer-Finance.canton.domain"
      public-api {
        port = 5019
        address = "localhost"
      }
    }
  }

  participants {
    bank {
      domains = ["BankA.canton.domain", "Oraclizer-Finance.canton.domain"]
    }
    oraclizer {
      domains = ["Oraclizer-Finance.canton.domain"]
    }
  }
}
```

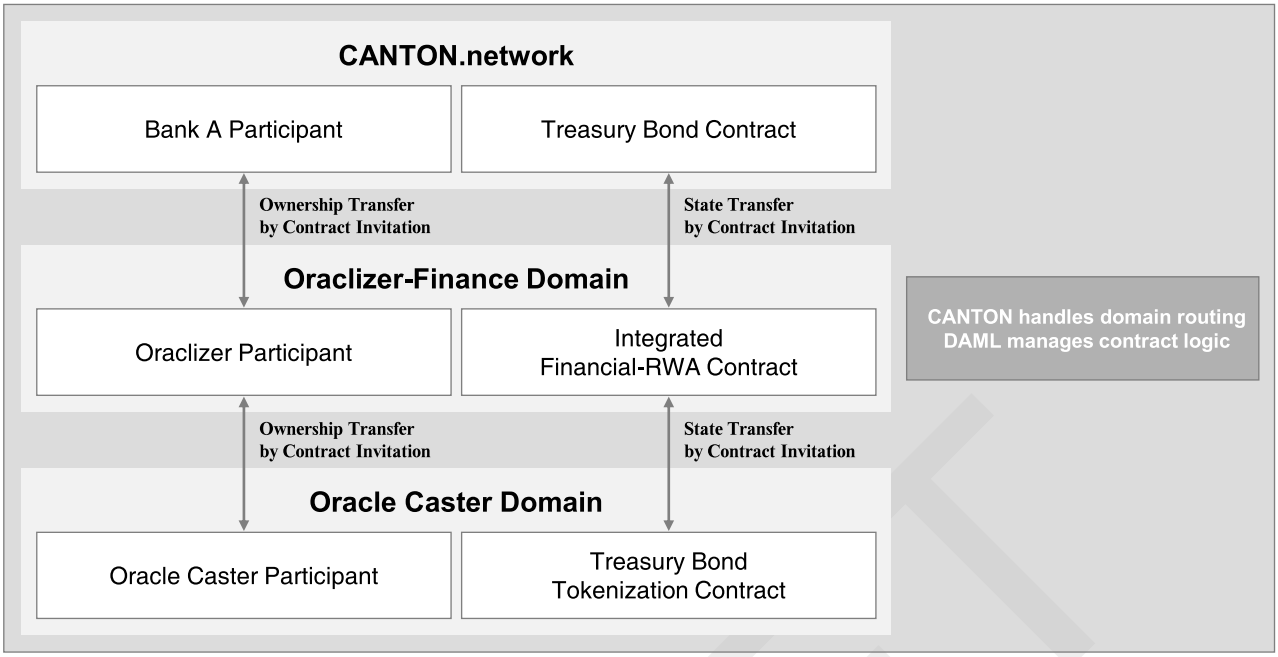

Figure 13. CANTON Driver Architecture

Gaming RWA Integration

Unlike existing RWA tokenization solutions that primarily focused on financial assets, **Oraclizer presents a new vision for comprehensive RWA that enables game assets to be treated as tokenized RWA within a secure regulatory-compliant environment.** Game companies face constraints in registering games with tokens on application platforms (e.g., App Store, Play Store). However, tokenization of game assets through Oraclizer allows them to leverage blockchain benefits without regulatory constraints, similar to how tokenized treasury bonds are not classified as virtual assets. Since the Oraclizer-Gaming CANTON domain provides the same level of regulatory compliance and oracle interoperability as the Oraclizer-Finance CANTON domain, **game companies gain the opportunity to safely advance their in-game economics without issuing virtual assets.**

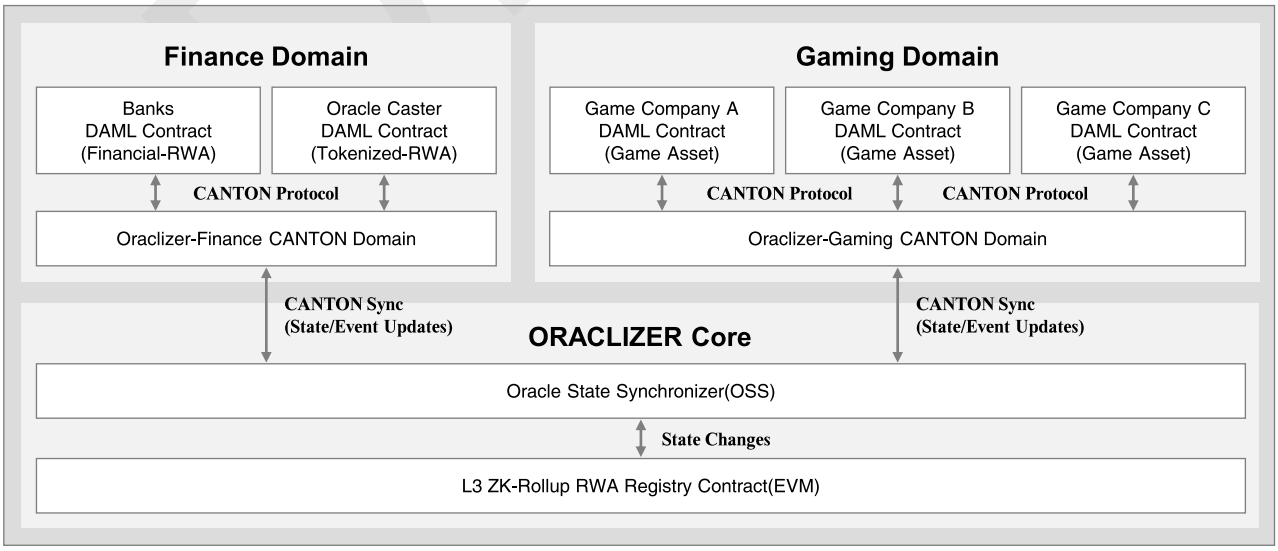

Figure 14. Oracle Contract for Gaming RWA via Oraclizer

The core limitations faced by traditional GameFi were regulatory risks from virtual asset issuance and unsustainable gameplay overly dependent on token economics. This not only hindered entry into traditional application platforms but also compromised the inherent fun factor of games. Oraclizer fundamentally resolves these issues by treating game assets as RWA. While game companies can maintain their existing game systems while leveraging expanded economic elements through tokenization, **players can maintain their original gaming experience without conscious interaction with complex token economics, naturally utilizing asset value, thus enabling the true fusion of gaming and finance that GameFi envisioned.**

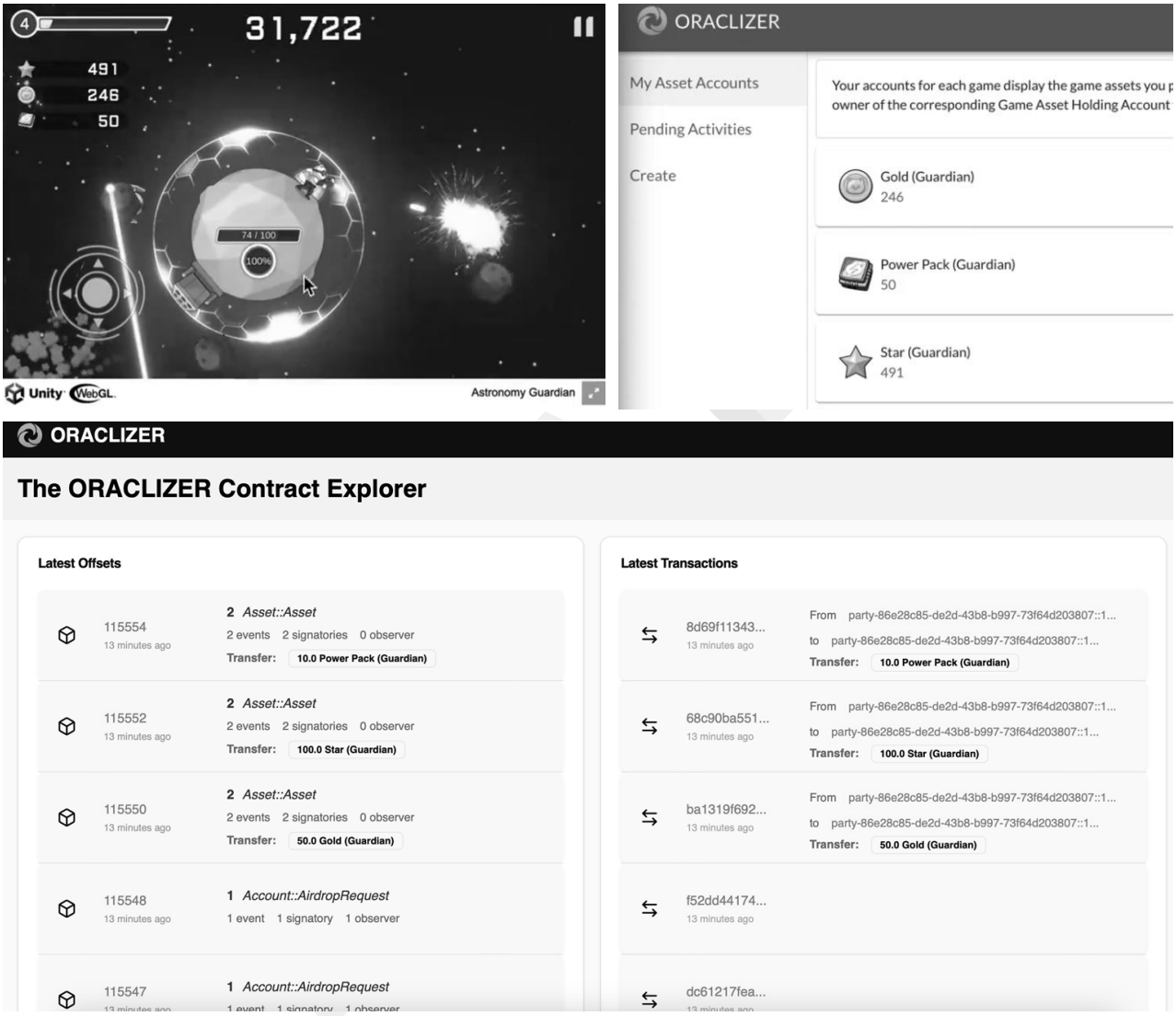

Figure 15. Tokenized Gaming RWA Demo and RWA Contract Explorer in Oracle Caster

6. Performance Analysis

While on-chain and oracle state synchronization has been a long-standing dream in the blockchain industry, it remained unrealized due to technical issues and limitations in scalability and Gas fees. For an oracle state machine to succeed, a practical scaling strategy is essential to address cross-chain latency and performance overhead during continuous state reflection. If the cumulative costs of cascading transactions triggered by a single oracle call exceed the expected benefits, it ultimately threatens the system's viability.

Moreover, without ensuring oracle data reliability from untrusted external systems, a decentralized state machine at Stage 1 level, and a robust regulatory compliance framework, both implementation and operation of such an innovative system are impossible. **Implementing an oracle state machine requires demonstrable results for scalability, economic efficiency, security, and regulatory compliance.**

Scalability Metrics

Oraclizer's scalability has unique architectural advantages specific to state synchronization oracles. While traditional oracles must process new data independently end-to-end for each call, with verification overhead increasing linearly with call frequency, state synchronization oracles maximize scalability through incremental proving and modular layer scaling strategies as transactions accumulate. This is achieved through state transition mechanisms that only prove state differences (state diffs) to SSOT, high compression via Blob-hash DA layer, and zkVerify's verification optimization.
For example, when processing treasury bond price changes through oracle:

1. **Traditional Oracles:** Require new independent transactions for each price change, necessitating complete verification for each

2. **Oraclizer:** Stores only incremental proofs of price differences as SSOT, with enhanced efficiency through L3 incremental proving, off-chain DA, and zkVerify utilization

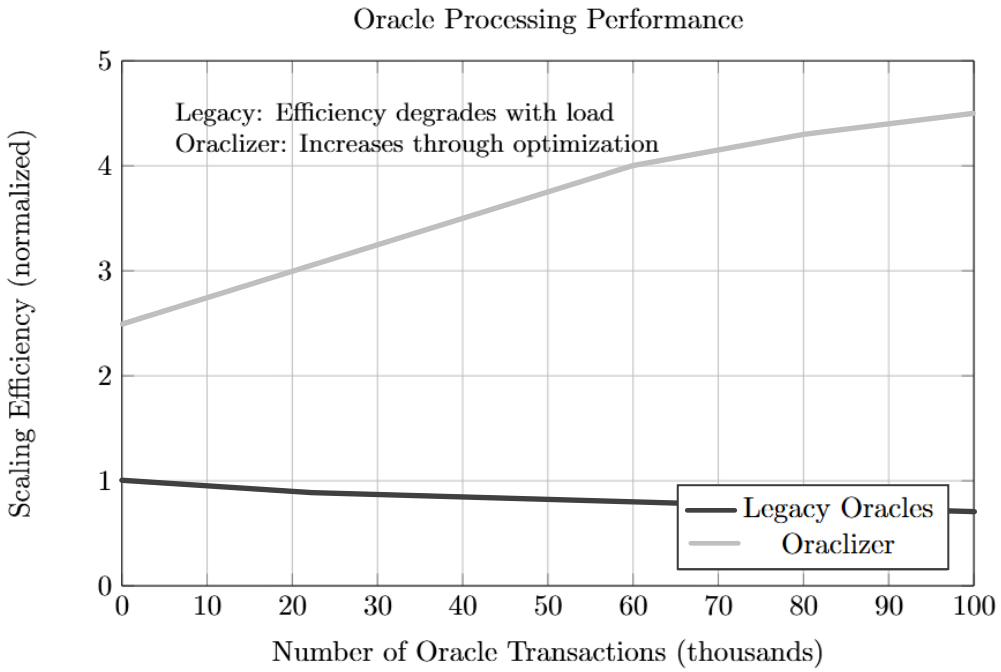

Figure 16. State Synchronization Oracle Specificities
that Increase Performance as Oracle Tx Demand Increases

Oraclizer's incremental proving approach is similar to Git's differential storage method. Just as Git efficiently manages only changed portions instead of storing the entire codebase repeatedly, **Oraclizer achieves exceptional efficiency by proving only state differences and synchronizing them through various scaling strategies.**

Cost Analysis

Gas fees present the greatest technical challenge in implementing state synchronization oracles. From a pricing policy perspective, implementing state synchronization with existing oracle solutions requires new oracle calls each time, incurring independent costs for each call. **Traditional oracles operate inefficiently, similar to making and paying for new API calls each time to track real-time stock prices.**

In contrast, Oraclizer provides an innovative cost structure enabling continuous state synchronization without additional costs after the initial oracle contract. This, combined with its unique characteristic of logarithmically decreasing Gas fees as transactions increase through higher-order layers and computational offload modulars with DA layer for reduced zk verification costs, dramatically lowers user costs. This is achieved through the combined effect of the following technical factors:

1. **Incremental State Processing Optimization**
   In L3 structure, consecutive transactions can reuse proofs of previous states. For example, when treasury bond prices change from 100→110→120, proving the 120 state requires only an incremental proof of the 110→120 change, not a complete proof from 100. This enables optimized proof generation utilizing previous proofs, beyond simple state diffs.

2. **Cumulative Data Optimization**
   Off-chain DA layer discovers and optimizes redundancies in consecutive transaction data structures. As transactions accumulate, data patterns become more predictable, leading to more efficient data compression. This is similar to how ZIP compression achieves higher compression rates with more similar data.

3. **Verification Cost Reduction**
   Consecutive transaction verification in zkVerify can utilize previous verification results. For instance, verification of the nth transaction builds on the (n-1)th verification result, eliminating the need to re-verify the entire state. This leads to optimization of the verification logic itself, beyond simple incremental verification.

These technical optimizations enable practical implementation of state synchronization oracles. While traditional oracles' repetitive calls and accumulating cost structure made state synchronization implementation difficult, Oraclizer's approach fundamentally resolves this issue. **For example, in cases requiring asset price synchronization at one-minute intervals over 24 hours, traditional oracles need 1440 individual calls with associated costs, while Oraclizer can process this with no additional costs beyond the initial contract fee.**

In particular, Oraclizer's RWA Registry maintains the latest state until actual state changes occur in external systems or blockchains. This eliminates unnecessary oracle calls and enables access to current information through simple on-chain state queries. By reducing the number of transactions needed for state updates through on-chain SSOT, it further enhances the overall system's cost efficiency.

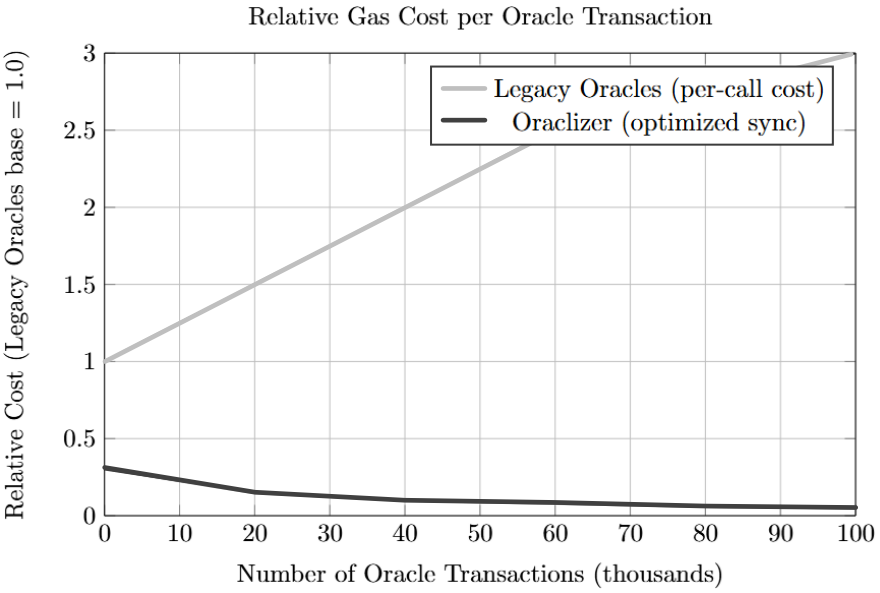

Figure 17. Analyzing Gas Cost for Oraclizer vs. Legacy Oracles

Gas fees reducing on a logarithmic scale by over 93% per Tx translate to service costs, establishing the foundation for an innovative pricing policy that enables continuous state synchronization services with just a single oracle call fee.

Security Evaluation

Implementing state synchronization with external systems while maintaining blockchain's decentralized nature entails complex security challenges. An oracle state machine must ensure reliability of both blockchain and non-blockchain systems, while addressing various security threats that may arise in cross-chain environments after states transition on-chain. Oraclizer achieves Stage 1 rollup through the D-quencer consensus algorithm, serving as a key element that simultaneously ensures system decentralization and security.

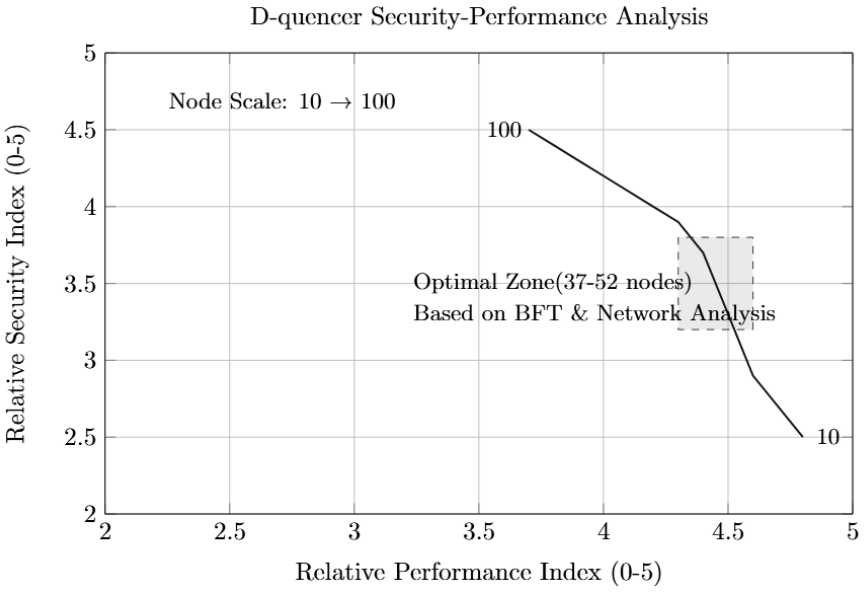

Figure 18. Evaluation of Security and Performance Indexes based on The Number of D-quencer Nodes

Through simulations and testing of how the D-quencer consensus algorithm affects BFT security thresholds, consensus delay overhead, network message complexity, and VRF entropy (random function unpredictability) based on node fluctuations, we found that minimum security standards could be met with at least 10 nodes, relative efficiency indices were optimal in the 37-52 node distribution range, and marginal utility in security-performance relative indices decreased beyond 90-100 nodes, leading to an evaluation of 100 nodes as the maximum node threshold.

Compliance Verification

RCP, as the blockchain industry's first regulatory research comprehensively analyzing financial regulatory authorities' recommendations and financial product guidelines, maintains value-neutral public interest rather than representing specific group or individual claims. Solutions related to tokenized RWA in capital markets must comply with the core value of maintaining financial market order. RCP's 31 regulatory items provide objective criteria for measuring these solutions' regulatory compliance levels. This verification evaluates system regulatory compliance according to five core principles: completeness, traceability, confidentiality, enforceability, and tokenization.

Property	ORACLIZER	Legacy Oracle Solutions
Finality	✓ Bidirectional state sync ✓ Cross-chain atomicity guarantee ✓ Complete state transition	✗ Simple data delivery only ✗ Lack of state synchronization ✗ No atomicity guarantee
Traceability	✓ KYC integration with OCID (System-wide AML) ✓ Complete asset movement tracking ✓ Regulatory supervision support	✗ Limited transaction tracking ✗ No unified identity management ● Partial supervision capability
Confidentiality	✓ Need-to-know based disclosure ✓ ZK-based privacy protection ✓ Selective information sharing	✗ Limited privacy protection ✗ No selective disclosure
Enforceability	✓ Asset freeze/forced liquidation ✓ Immediate regulatory action ✓ Cross-chain enforcement	✗ No regulatory mechanisms ✗ Limited asset control ✗ Unable to enforce regulations
Tokenizability	✓ Clear rights relationship ✓ Legal binding framework	✗ Lack of standardization ✗ No legal binding power

Table 1. Regulatory Compliance Analysis: Oraclizer vs Legacy Oracle Solutions

RCP compliance goes beyond mere technical differences to establish new standards for operationally viable oracle solutions in capital markets. Oraclizer's high-level regulatory compliance principles enable it to serve as a reliable bridge between TradFi and DeFi and represent essential prerequisites for state synchronization oracles to function in real financial environments.

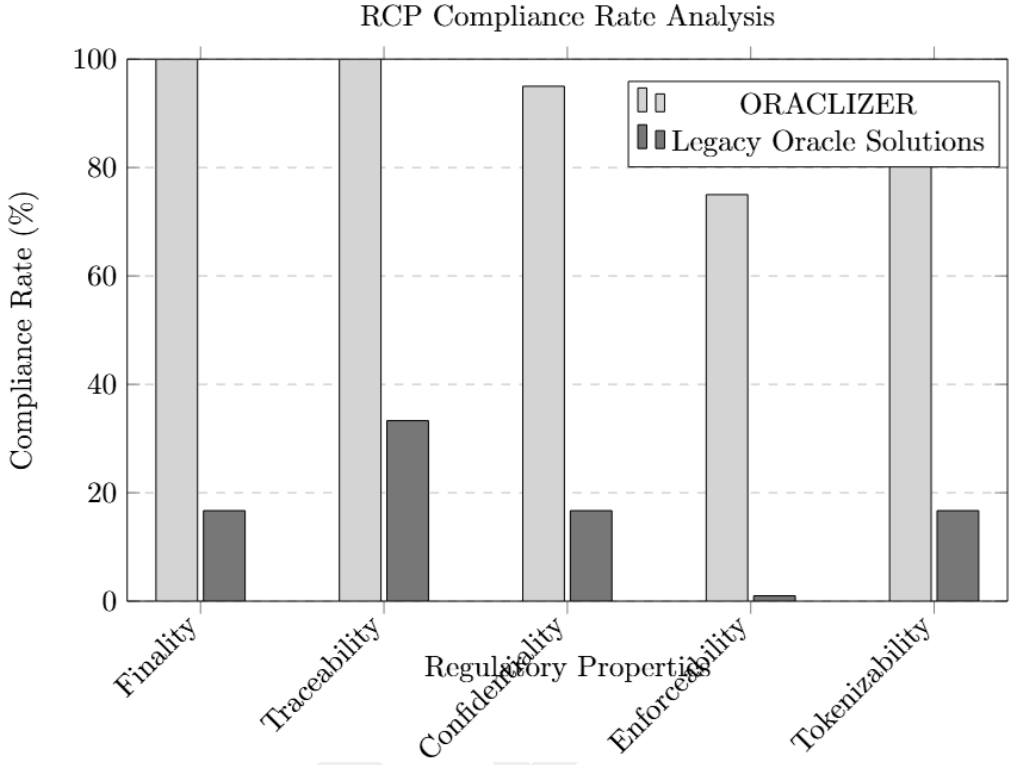

Figure 19. Quantitative Analysis of RCP Fulfillment

7. Future Development

We have opened new horizons for oracle state synchronization, but this is just the beginning. Our greatest challenge was first achieving complete state synchronization in cross-chain and oracle environments, for which we presented solutions from the perspective of proper direction, including regulatory research, for various technical strategies and viable services. However, more important is considering how this technology will be used in the real world.

For example, our planned integration with Intent Standard Protocol[10] is not merely adding functionality, but rather our effort to reinterpret oracle contracts from a user perspective. Our pursuit of bridgeless architecture represents an attempt to overcome fundamental limitations in the current cross-chain ecosystem, while SMT smart merging[11] research expresses our commitment to continuously improving system efficiency.

Particularly, expanding from the traditional financial RWA-focused tokenization market to the gaming industry holds significant meaning in pioneering new markets. Overcoming the limitations of existing GameFi and realizing true game asset liquidity will be an important milestone in proving blockchain technology's practical value.
These challenges will be realized one by one through systematic roadmap execution and continuous research and development.

10. Enhances cross-chain interoperability by enabling declaration of intentions such as token swaps and governance voting across multiple chains
11. Technology for efficiently merging multiple Merkle trees. Improves verification speed and storage efficiency while maintaining data consistency through tree structure optimization

Research Directions

While we have presented a new paradigm of bidirectional synchronization contracts through oracle state machines, there remain areas requiring improvement. We continue to analyze the system's limitations and conduct research to find better solutions.

1. **Intent-based Oracle Contract Research:** Current oracle contracts remain structurally complex, limiting user experience. We aim to improve this through integration of ERC-7683[13] standard and OIP.

    - Abstracting complex cross-chain oracle requests into single intents
    - Automatic derivation of optimal execution paths based on user intentions
    - Enhanced automation level of state synchronization processes

2. **Bridgeless Architecture Expansion:** Oraclizer already achieves partial bridgeless functionality through integrated zk bridges. However, this needs to be expanded to broader cross-chain environments.

3. **Privacy-Compliant API Drivers:** Beyond DLTs like DAML that provide privacy from ledger infrastructure, privacy protection in more diverse environments is needed for integrating general external systems.

    - Developing new API communication protocols based on zk proofs
    - Data processing utilizing Fully Homomorphic Encryption

4. **SMT Smart Merging Optimization:** Given the appchain characteristics, current state management systems have room for optimization.

    - Improving incremental state update efficiency
    - Researching integration possibilities with Verkle Trees[12]
    - Enhancing Merkle proof generation speed through parallel processing

While these research initiatives are conducted independently, they will ultimately converge into a unified system. In particular, we anticipate that Intent-based contracts and expanded bridgeless architecture will open new horizons for cross-chain state synchronization.

Roadmap

2025.Q1

- Integration of OZ token as Oraclizer's native gas token
- Initiation of ERC-7683 Intent protocol integration development
- Launch of Oraclizer testnet (with zkEVM support)
- NEW-EIP: RCP (informational-EIP), regulatory enforcement-related (Standard-EIP)

12. Data structure for state storage optimization, generating smaller proofs than Merkle trees

2025.Q2

- Configuration of Oraclizer CANTON network and driver integration testing
- Integration of zkVerify and DA modular
- KYC integration
- Oracle Caster (β)
- Gaming RWA expansion (Oraclizer-Gaming)

2025.Q3-Q4

- Mainnet launch
- New chain expansion: Arbitrum, Optimism, Base
- Real estate RWA expansion (Oraclizer-RE)

2026.H1

- Development of SMT smart merging prototype
- Development of Intent-based oracle contract prototype
- Expansion of privacy-compliant API drivers

2026.H2

- Expansion of bridgeless architecture
- Integration of SMT smart merging (performance scaling)
- Enhancement of cross-chain scalability

2027

- Implementation of multi-state (multiple timepoint) rollback system (fault tolerance)
- Digital asset regulatory standardization framework (open source)

Ecosystem Expansion

We aim to initiate a new oracle ecosystem centered on financial RWA. This is a deliberate choice to validate system stability and reliability by starting in the most stringent regulatory environment. However, the true value of state synchronization oracles will be realized in connecting with broader ecosystems.

Collaboration with DeFi protocols carries meaning beyond simple partnerships. We have deeply understood each protocol's unique requirements and undergone the process of organically integrating them into our architecture. Collaboration with global financial institutions requires an even more cautious approach. While respecting each institution's regulatory requirements and operational methods, we will jointly build a framework for efficient tokenization and utilization of RWA.

Expansion into the gaming industry requires particularly careful consideration. The Oraclizer-Gaming domain will evolve toward realizing true value of game assets while preserving the essential fun of gaming. A new challenge in the real estate market also awaits. While real estate RWA must consider complex elements such as SPCs, valuation, legal structures, and liquidity, it represents a challenge we must undertake for efficient utilization of physical assets.

The developer ecosystem will form the foundation for all these expansions. We will provide necessary tools and support for developers to realize their innovative ideas through Oraclizer. This means building a collaborative ecosystem that develops the new paradigm of state synchronization oracles together, beyond mere technical support.

8. Conclusion

The 'Oracle Problem,' which has been viewed as a fundamental limitation of blockchain, has evolved from its simple beginnings of connecting external data into increasingly complex and sophisticated forms. However, oracles enabling true state synchronization have remained unattainable due to technical limitations, network costs, and regulatory compliance challenges. As the first oracle state machine to overcome these limitations and achieve complete state synchronization, Oraclizer presents a new thesis to the blockchain industry.

Oraclizer serves as a new 'knowledge layer' in the EVM ecosystem that can directly understand and process the complete state and changes of RWAs on-chain. Through this, smart contracts can comprehensively handle not just simple oracle data, but asset states, regulatory compliance status, and contract conditions, expanding the application scope of oraclized information.

The RWA tokenization market is projected to grow from a minimum of $4 trillion to a maximum of $30 trillion by 2030[14]. As an oracle state machine, **Oraclizer creates network value equivalent to the combined scale of the RWA tokenization market and DeFi protocols attempting oracle requests.** Moreover, potential expansion into new RWA domains beyond financial assets, such as the gaming industry and real estate market, signifies the potential for continuous growth in the Oraclizer ecosystem's value.

When massive capital markets become tokenized and connected to DeFi through Oraclizer, the resulting innovation in capital efficiency will bring us one step closer to realizing the ultimate goal that blockchain technology has been pursuing.

References

[1] IntoTheBlock and Trident Digital. (2024). DeFi's Next Frontier: In-Depth Report on Institutional DeFi. Globe Newswire.

[2] Financial Action Task Force (FATF). (2023). Updated Guidance for a Risk-Based Approach to Virtual Assets and Virtual Asset Service Providers.

[3] Digital Asset. (2024). DAML: The Contract Language for Building Composable Multi-party Applications. https://www.digitalasset.com/daml

[4] Financial Stability Board. (2023). Regulation, Supervision and Oversight of Crypto-Asset Activities and Markets. https://www.fsb.org/2023/07/regulation-supervision-and-oversight-of-crypto-asset-activities-and-markets-2023-progress-report/

[5] Horizen Korea, & Oraclizer Core. (2024). Regulatory Compliance Protocol (RCP) for Tokenized Capital Markets. https://www.slideshare.net/slideshow/embed_code/key/kAZFXrBLjtkinq

[6] Ethereum Foundation. (2023). EIP-1400: Security Token Standard. https://eips.ethereum.org/EIPS/eip-1400

[7] Ethereum Foundation. (2023). EIP-3643: T-REX Token for Regulated EXchanges. https://eips.ethereum.org/EIPS/eip-3643

[8] Chainlink. (2023). The Blockchain Oracle Problem. https://chain.link/education-hub/oracle-problem

[9] Caldarelli, G., & Ellul, J. (2021). The Blockchain Oracle Problem in Decentralized Finance—A Multivocal Approach. Applied Sciences, 11(16), 7572. https://doi.org/10.3390/app11167572

[10] Ethereum Foundation. (2023). EIP-3770: Chain-specific addresses. https://eips.ethereum.org/EIPS/eip-3770

[11] Digital Asset. (2024). DAML: The Contract Language for Building Composable Multi-party Applications. https://www.digitalasset.com/daml

[12] Matter Labs. (2023). zkSync: Scaling and privacy engine for Ethereum based on ZK Rollup. Retrieved from https://zksync.io/zksync.pdf

[13] Ethereum Foundation. (2023). EIP-7683: Intent-based Transactions. Retrieved from https://eips.ethereum.org/EIPS/eip-7683

[14] Tren Finance. (2023). Real World Asset (RWA) Tokenization Market Forecast 2030. Retrieved from https://www.chaincatcher.com/en/article/2147534