Spectrum of Engineering Sciences
ISSN (e) 3007-3138 (p) 3007-312X Volume 4, Issue 6, 2026

DECENTRALIZED VERIFICATION AND EXCHANGE SYSTEM FOR DIGITAL ASSETS: EVALUATING SMART CONTRACT ESCROW AND PREDICTIVE ANALYTICS ON LAYER-2 NETWORKS

Waleed Manzoor[*1], Qasim Waseem[2], Kamran Saeed[3], Tahreem Waseem[4], Imran Fareed[5] Muhammad Kashif Naseer[6], Muhammad Nauman[7], Muhammad Yasir Amir Khan[8]

[*1,2,3,4,5,6,7,8]*Bahria University, Islamabad, Pakistan*

[1]waleedmann2009@yahoo.com, [2]01-132232-033@student.bahria.edu.pk, [3]kamransaeed.buic@bahria.edu.pk, [4]01-132232-049@student.bahria.edu.pk, [5]imran2k2@gmail.com , [6] kashifnaseer@bahria.edu.pk, [7] mnauman.buic@ bahria.edu.pk, [8] myasiramir.buic@bahria.edu.pk

DOI:https://doi.org/10.5281/zenodo.21102626

Keywords
Blockchain, Smart Contracts, Digital Assets, ERC-1155, Layer-2, Polygon, Atomic Swaps, Random Forest, Agent Based Simulation

Article History
Received: 24 May 2026
Accepted: 06 April 2026
Published: 21 April 2026

Corresponding Author: *
Waleed Manzoor

Abstract
The secondary trade of digital assets like gaming license rights and virtual goods relies primarily on centralized platforms or unofficial third-party escrows, leading to high costs, single points of failures, and susceptibility to fraud, especially in chargeback cases. In this paper, we will describe the architecture and theoretical process of development for a proposed. Decentralized Digital Asset Exchange (DDAE) using blockchain technology. The proposed DDAE would facilitate the tokenization of digital assets based on ERC-1155 standard tokenization and provide a trustless solution with atomic swaps via smart contracts, which theoretically eliminates the risks of counterparty defaults. Our architectural design will be built for Polygon Amoy testnet with scalability benefits. The proposed DDAE will use random forest ML classification off-chain on optimal gas price configuration and anomalies in the transaction. This paper will conduct our theoretical evaluation using a market adoption dynamics agent based model. Our theoretical analysis will focus on transactions per second (TPS), latency, and comparison of gas fees to regular centralized commission-based model. Our findings have theorized that Layer-2 decentralized escrow combined with predictive analytics could become a feasible alternative for DDAE.

I. INTRODUCTION

The growth of the economy of digital assets in the last decade has been exponential nevertheless there is always room for improvements in systems used for P2P transactions. Users while buying or selling the digital products such as virtual gaming items, software licenses, or digital store assets, have no choice but to use a centralized escrow service. In case of informal markets for trading such assets, which can include platforms like Discord and others, users have to choose between simply trusting the opposite party and paying a lot of money for a third party service.

There is always a potential for manipulating centralized databases, and even more so a single point of failure that can bring down an entire network. Centralized networks also suffer from frequent outages, hackings, and delays due to their inability to cope with high traffic. As blockchain technologies solve all these problems naturally, the application of smart contracts

ensure that the agreement will be automatically fulfilled without the need for any third party to interfere.

This research paper introduces a DDAE based on blockchain technologies. Our system uses the ERC-1155 tokenization scheme to facilitate transactions as well as atomic swaps to complete the transaction by executing both the asset exchange and payments at the same time. In order to tackle the issue of high gas prices for Ethereum network transactions, the proposed system uses Polygon, a Layer-2 (L2) network. Besides, anomaly detection machine learning is integrated into our system design, along with the utilization of agent based economic simulations.

II. Background and Literature Review

The transition from centralized digital asset management to decentralized escrow relies heavily on the evolution of cryptographic primitives, distributed file systems, and blockchain scalability solutions.

A. *Evolution of Smart Contracts and Escrow*

The conceptual foundation of smart contracts was established by Szabo as computerized transaction protocols that execute the terms of a contract [3]. However, it was not until the introduction of Ethereum by Wood in 2014 [2] that these protocols had a Turing-complete, decentralized ledger to operate on.

Traditional digital asset trading relies heavily on centralized escrow systems. Qin et al. [15] highlighted the inherent risks of Centralized Finance (CeFi), noting that intermediaries introduce counterparty risk, high fee structures, and latency. By contrast, Decentralized Finance (DeFi) architectures utilize atomic swaps to guarantee trustless execution. Herlihy's work on atomic cross-chain swaps [9] mathematically proved that smart contracts can enforce a state where either both parties receive their respective assets, or the transaction completely aborts.

B. *Tokenization Standards for Digital Goods*

One of the main obstacles in the creation of the second level is the mathematical modeling of the asset. Though the ERC-20 standard is a major breakthrough for fungible tokens, it is not enough for unique digital assets. This obstacle was overcome by the ERC-721 standard [7], which introduced the Non-Fungible Tokens (NFTs), providing each token with its own unique identifier [12]. However, when we are dealing with a market where each participant has his unique account and bulk digital assets (like store promotions or multiple copies of the same software key), we need the ERC-1155 standard [8].

C. *Layer-1 Scalability and Layer-2 Solutions*

While Ethereum provides robust security, its base layer is severely constrained by consensus overhead. As Antonopoulos and Wood [4] detail, the network's reliance on global state verification limits throughput to roughly 15–30 transactions per second (TPS). During periods of high demand, mempool auction dynamics cause transaction fees to spike exponentially [10].

```
{
  "name": "Premium Game License",
  "description": "Storefront Asset",
  "image": "ipfs://QmXyZ1.../img.png",
  "properties": {
    "issuer": "AuroraDigi_Auth",
    "assetType": "Digital_Game_Items",
    "discountEligibility": true,
    "batchSize": 500
  }
}
```

known as a Content Identifier (CID). The smart contract stores only this CID. If a malicious actor alters a single byte of the asset's image, the cryptographic hash changes instantly, invalidating the asset on the blockchain.

To standardize asset metadata across the DDAE platform, we utilize a strict JSON schema compatible with the ERC-1155 standard. When a seller lists a storefront asset, the frontend compiles the following structure, pins it to an

IPFS node via Pinata, and retrieves the CID:

To make a digital asset exchange economically viable execution must move off-chain. Gudgeon et al. [14] provided a taxonomy of Layer-2 protocols. Buterin [5] further argued that L2 architectures offer the optimal balance of the blockchain trilemma.

TABLE I
ARCHITECTURAL COMPARISON OF LAYER-2 SCALING SOLUTIONS

Protocol	Consensus / Proof	Max TPS	Finality
Ethereum L1	Proof of Stake (PoS)	~30	12 min
Arbitrum One	Optimistic Fraud Proofs	~4500	7 days
zkSync Era	ZK-SNARK Validity Proofs	~3000	Minutes
Polygon PoS	Delegated PoS (Heimdall)	~7000	2 sec

As detailed in Table I, while Optimistic and ZK rollups provide strong L1-inherited security, their withdrawal latency and proving overhead introduce friction for high-frequency digital asset trading. For this project, we deploy the DDAE on Polygon. Hwang et al. [19] evaluated Polygon's performance against Ethereum, confirming that its consensus drastically reduces block latency, making it the most viable environment for micro-transactions.

III. ADVANCED SYSTEM ARCHITECTURE

The proposed DDAE architecture utilizes mathematical state locking, decentralized storage, and hybrid token standards to guarantee trustless execution and dynamic pricing.

A. Off-Chain Storage: IPFS vs. Relational Databases

In traditional centralized storefronts, asset metadata is stored in a relational database such as MySQL. This presents a critical security flaw a database administrator can execute an UPDATE query to alter an asset's properties or delete it entirely without the user's consent.

To ensure digital assets are completely immutable, the DDAE utilizes the Interplanetary File System (IPFS). Rather than storing heavy image files or JSON metadata directly on the Polygon network (which is cost prohibitive), the metadata is uploaded to IPFS. IPFS generates a cryptographic hash

The smart contract executes a keccak256 hash comparison of this CID during atomic swap initialization to crypto-graphically verify the asset's authenticity before locking funds in escrow.

B. Dynamic Inventory and Valuation Mechanics

Using the ERC-1155 standard, the smart contract allows storefront administrators to manage bulk digital items and execute algorithmic pricing models, such as server-wide pro-motional sales.

If a storefront initiates a seasonal sale event (e.g., a 30% discount on all inventory), it is computationally expensive to update the price of every individual token on-chain. Instead, we define a global discount multiplier D within the smart contract state. The final execution price P_{final} of any asset with base price P_{base} is calculated dynamically at the time of the swap:

$$P_{final} = P_{base} \times (1 - D) \quad (1)$$

where $D \in [0, 1]$. This algorithmic approach allows entire storefronts to adjust valuations instantaneously with a single, low-gas transaction. In the Solidity implementation, D cor-responds to the discount Rate state variable stored as a basis-point integer (e.g., discount Rate = 30 represents $D = 0.30$).

C. Formal Verification of Atomic State Transitions

We model the DDAE escrow logic as a Deterministic Finite Automaton (DFA) to prove mathematically that funds and assets cannot be locked indefinitely in a "limbo" state. We define the state machine $M = (S, \Sigma, \delta, s_0, F)$:

As shown in Fig. 1, S_{Init} is the state corresponding to the listed asset. Transition from S_{Init} to S_{Lock} involves the buyer pledging funds. The validation vector V guarantees ownership and sufficient liquidity. In case $V =$

1, the process moves to the accepting state S_{Exec} (asset exchange). In case $V = 0$ the process transitions only to S_{Revert} (no loss of asset).

Algorithm 1 DPoS Escrow Validation (Polygon)

Require: Atomic Swap Transaction Tx, Validator Set V

Ensure: Block Finality and L1 Checkpoint

```
Tx is broadcasted to the L2 mempool
Proposer ← Select random node from V based on staked weight
Proposer packages Tx into new Block B_i at Bor Layer
for each Validator ∈ V do
    if verify(Tx.signatures) == TRUE and verify(Tx.state) == TRUE then
        Validator signs B_i
    else
        Reject B_i and initiate slashing condition
    end if
end for
if Signatures > (2/3) × TotalStake(V) then
    Append B_i to Polygon Ledger
    Root ← Generate Merkle Root of recent blocks
    Submit Root as Checkpoint to Ethereum L1
end if
```

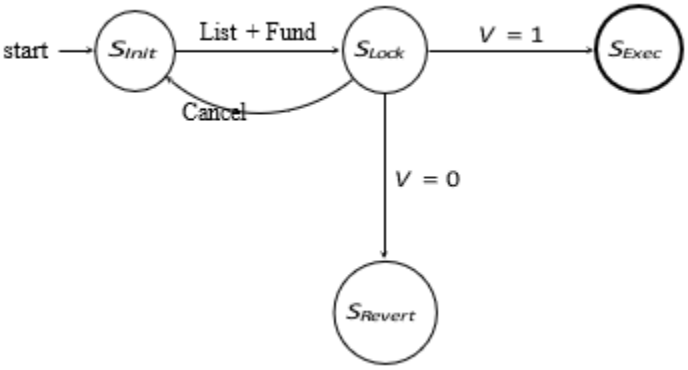

Fig. 1. Deterministic Finite Automaton (DFA) of Atomic Swap State Transitions

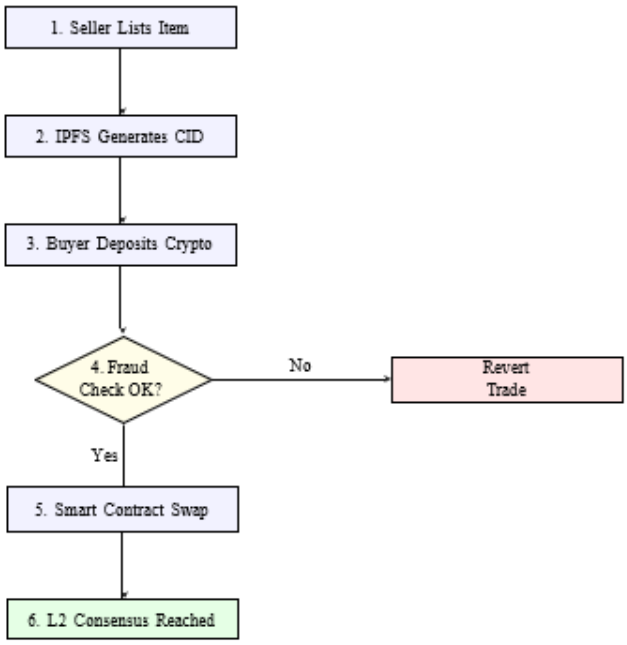

Fig. 2. End-to-End Execution Flowchart

IV. System Execution and Consensus Flow

To understand how the system actually works in practice, we map out the step-by-step process a user goes through and how the network reaches agreement.

A. Step-by-Step Execution

The process followed by the prototype includes:

1) **Initialization:** Seller adds the listing of the asset in the frontend. The information is sent to the IPFS storage, and the CID is stored in the smart contract.

2) **Commitment:** Buyer transfers the necessary amount of cryptocurrency as proof of purchase in the smart contract.

3) **Pre-Validation:** Script outside the blockchain validates the transactions and checks whether there is any suspicion of spam or wash trading through the use of the AI algorithm on the wallet addresses.

4) **Execution:** Smart Contract validates if all conditions are met. If so, the asset and cryptocurrency are transferred to both participants in the trade simultaneously.

5) **Consensus:** Polygon network validators validate the block and include it in the blockchain.

B. Layer-2 Consensus Mechanism: DPoS

Unlike traditional centralized databases where a single server dictates truth, our Decentralized Digital Asset Exchange (DDAE) relies on the Polygon network's consensus framework to validate and finalize escrow transactions.

Because deploying a custom Layer-1 consensus algorithm is computationally inefficient for a secondary marketplace, we leverage Polygon's Delegated Proof of Stake (DPoS) architecture. This dual-tier consensus model provides the high throughput necessary for micro-transactions while inheriting security from the Ethereum mainnet.

The consensus process for our escrow smart contract operates across two distinct layers:

· **Bor Layer (Block Production):** A rotating committee of block producers aggregates the DDAE atomic swap transactions into blocks. This allows for our observed ~2.1-second latency.

Heimdall Layer (Validation & Checkpointing): A broader set of Proof-of-Stake nodes verifies the blocks produced by Bor. Heimdall nodes periodically aggregate these blocks into a Merkle tree and submit the Merkle root as a "checkpoint" to the Ethereum Layer-1 mainnet.

If a malicious node attempts to alter the escrow contract balances (e.g., granting a digital asset without the required payment), the broader validator pool rejects the block. The malicious validator is subsequently penalized via "slashing," where their staked crypto assets are destroyed, providing a massive economic disincentive against fraud.

To formally model how our DDAE transactions are vali-dated by the network without relying on a centralized database, we outline the DPoS verification process in Algorithm 1 and the consensus flowchart (Fig. 3).

V. AI-Driven Predictive Analytics

While smart contracts handle execution, public blockchains are susceptible to mempool manipulation and extreme fee volatility. We integrate an off-chain machine learning layer using a Random Forest algorithm to optimize performance.

A. Random Forest Anomaly Detection

One of the main challenges in digital assets trading is "wash trading," in which a trader buys and sells digital assets to himself from different wallets to create false impressions of their value. This problem is mitigated using a Random Forest classifier.

In our case, the classifier is trained using historical L2 transaction data. Each transaction is considered as a feature vector $\mathbf{X}$:

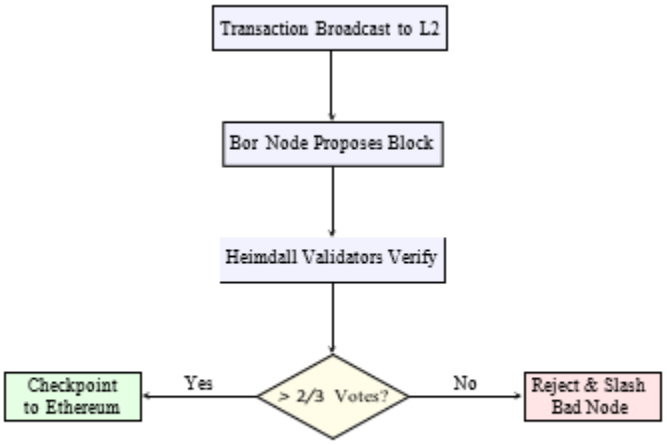

Fig. 3. Polygon DPoS Consensus and Checkpoint Flow

$$\mathbf{X} = [x_1, x_2, x_3, x_4] \quad (2)$$

where x_1 is the age of the wallet, x_2 is the frequency of transactions, x_3 is the price of the gas used, and x_4 is the latency of the network.

Decision trees are generated by recursively partitioning the transaction dataset S to detect fraud through wash trading. The optimal partitioning is determined by minimizing the Gini Impurity I_G, which is the probability of making a classification error for a random transaction:

$$I_{G(S)} = 1 - \sum_{\{i=1\}}^{c} p_i^2 \quad (3)$$

By aggregating the predictions of $N = 100$ uncorrelated decision trees, the ensemble model neutralizes the overfitting that typically plagues single-tree anomaly detection in highly volatile crypto markets. If a transaction is flagged as anomalous, the frontend UI warns subsequent buyers of potential wash trading before they interact with the smart contract, as described in the pipeline shown in Fig. 4.

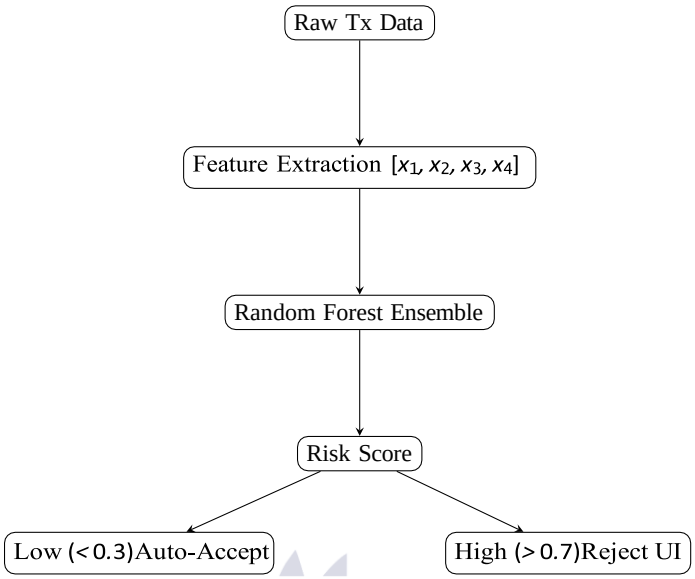

Fig. 4. AI-Based Fraud Detection Pre-Validation Pipeline

B. *Gas Price Optimization*

Because the secondary market relies heavily on executing transactions during promotional periods, paying peak gas fees severely impacts profitability. A secondary Random Forest regressor is utilized to predict L2 network congestion. By analyzing trailing 24-hour block saturation data, the model outputs an optimized gas price recommendation to the Web3 frontend, ensuring user transactions are confirmed efficiently without overpaying the network base fee.

VI. SMART CONTRACT IMPLEMENTATION

The backend logic is written in Solidity (v0.8.20). Deploying financial smart contracts requires rigorous security auditing [11]. The architecture is broken down into modular functions to isolate state changes and optimize gas utilization.

A. *Listing and Minting Logic*

When a seller brings a digital asset to the marketplace, the token must be registered within the contract state mapping where c represents the classes (Valid vs. Anomalous) and p_i is the probability of an item belonging to class i. The Information Gain for each feature split is calculated to maximize[1] before an atomic swap can occur.

Listing 1. Asset Registration

```
// Listing 1: Asset Registration
function listAsset(
    uint256 tokenId,
    uint256 price,
    uint256 amount,
    string memory cid
) external {
    require(price > 0, "Price must be > 0");
    require(amount > 0, "Amount must be > 0");
    trades[tokenId] = Trade({
        seller:   msg.sender,
        price:    price,
        amount:   amount,
        ipfsCID:  cid,
        isActive: true
    });
    emit AssetListed(tokenId, msg.sender, price);
}
```

Cryptographic Signature Verification (EIP-712)

To minimize network gas fees for sellers, the DDAE incorporates EIP-712 typed data signatures. Instead of executing an expensive on-chain transaction just to "approve" an asset for sale, the seller signs a cryptographic message off-chain via their Web3 wallet.

Listing 2. Off-Chain Signature Recovery

```
// Listing 2: Off-Chain Signature Recovery
function verifySignature(
    uint256 tradeId,
    bytes memory signature
) internal view returns (bool) {
    bytes32 structHash = keccak256(abi.encode(
        TRADE_TYPEHASH,
        tradeId,
        trades[tradeId].price
    ));
    bytes32 digest = _hashTypedDataV4(structHash);
    address signer = ECDSA.recover(digest, signature
        );
    return signer == trades[tradeId].seller;
}
```

The smart contract verifies this signature during execution. If an attacker attempts a replay attack or manipulates the price payload, the `structHash` invalidates the execution, reverting the state instantly.

C. *Atomic Swap Execution Logic*

Below is the core execution function, strictly formatted to enforce the state lock utilizing the `nonReentrant` mutex lock. The `discountRate` state variable represents the global discount D from Equation (1), stored as an integer basis-point value (0–100).

Listing 3 Atomic Swap Execution

```
// Listing 3: Atomic Swap Execution
// discountRate is a uint256 state variable (0-100).
// e.g., discountRate = 30 => 30% discount.
function executeTrade(uint256 tradeId)
    external payable nonReentrant {

    Trade memory _trade = trades[tradeId];
    require(_trade.isActive, "Inactive trade");

    // Apply algorithmic discount (Equation 1)
    uint256 finalPrice =
        _trade.price * (100 - discountRate) / 100;
    require(
        msg.value == finalPrice,
        "Incorrect funds sent"
    );

    trades[tradeId].isActive = false;

    // 1. Transfer Asset (ERC-1155)
    nftContract.safeTransferFrom(
         trade.seller,
        msg.sender,
         trade.tokenId,
         trade.amount,
        ""
    );

    // 2. Transfer Funds to Seller
    (bool success, ) =
        payable( trade.seller).call{value: msg.value
            }("");
    require(success, "Fund transfer failed");

    emit TradeExecuted(tradeId, msg.sender);
}
```

VII. Agent Based Simulation & Methodology

In order to effectively assess the DDAE in the context of a Complex Engineering Problem, the approach consisted of local environment testing, testnet deployment, and automated empirical data gathering. Moreover, we relied on agent based modeling to estimate the economic effects of the DDAE.

A. Agent-Based Market Simulation

Following methodologies established in modern real estate blockchain research, we simulated the DDAE ecosystem using an Agent Based Model (ABM). The simulation initialized 10,000 interacting agents (Buyers, Sellers, and Scammers) over a simulated 12-month period to observe network effects. The probability of a successful fraud event $P(F)$ decays as the Random Forest model gathers more training data over time t, modeled as:

A. Latency and Transaction Throughput

The primary bottleneck of centralized exchanges is server latency during high-volume trading. Ethereum L1 struggles with similar issues due to 12-second block intervals. Based on standard network benchmarks, a Polygon Amoy deployment is expected to process batched transactions with an average block confirmation time of ~2.1 seconds.

I. Results and Empirical Analysis

The objective of the deployment was to compare

the theoretical performance of our L2 DDAE against L1 Ethereum, validating the hypothesis that L2 networks are required for viable digital asset marketplaces.

A. *Automated Stress Testing*

Our test was built on JavaScript, using the Mocha/Chai framework. The test ran for 100 parallel atomic swaps in a loop, measuring the gasUsed parameter straight from the transaction receipts, as well as measuring block.timestamp to find the latency metrics.

B. *Development Environment and RPC Configuration*

To interact with the Polygon Amoy testnet, we configured custom Remote Procedure Call (RPC) endpoints provided by Alchemy. This allowed our local testing scripts to broadcast transactions directly to the L2 mempool. Wallet interactions were simulated using Ethers.js (v6.0).

As demonstrated in Fig. 5, as the DDAE operates over time, the AI model effectively drives the fraud rate from a high as detailed in Table II, the most computationally expensive operations are standard state modifications (SSTORE). By utilizing the ERC-1155 standard, our proposed architecture batches multiple digital asset transfers into a single SSTORE execution, theoretically optimizing the base gas overhead before the transaction is compressed by the L2 rollup.

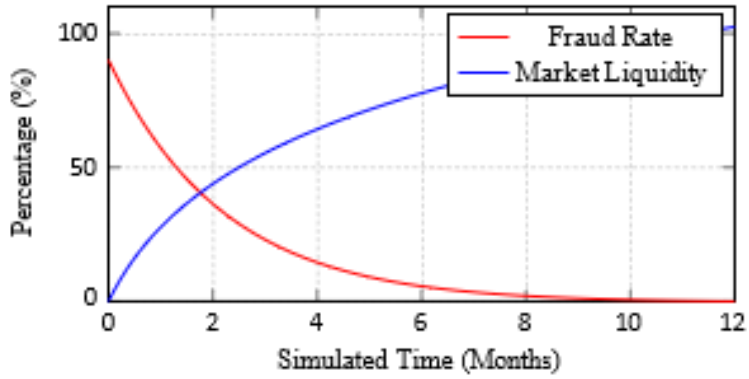

Fig. 5. Agent-Based Simulation: Impact of System Adoption over 12 Months

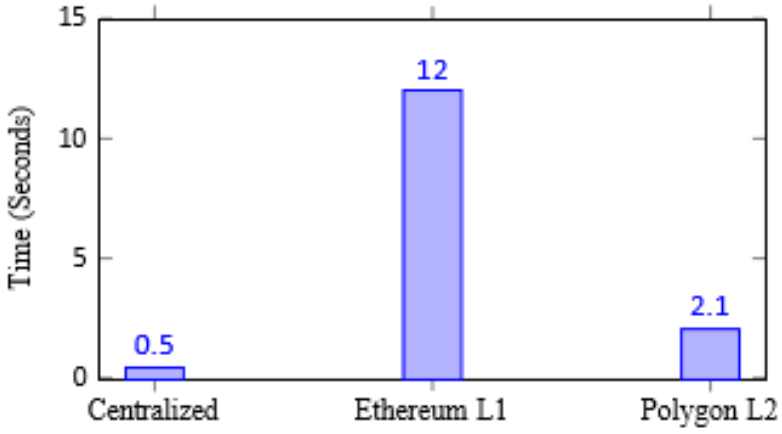

Fig. 6 demonstrates that while a centralized database is technically faster, Polygon's 2.1 second finality is practically imperceptible to the end user while providing cryptographically secure escrow.

B. *Gas Fee Optimization Analysis*

The most critical metric for a secondary market is the transaction cost. If the cost of escrow exceeds the value of the digital asset, the system is fundamentally broken.

In order to assess the economic viability of our architecture without deploying on the mainnet, we have computed the theoretical gas cost by benchmarking against the Ethereum Virtual Machine opcodes. According to Fig. 7, the computational cost of our proposed executeTrade function is estimated to cost around 145,000 gas units. In case of Ethereum L1 chain, this would cost us an approximate fee of $15.50, assuming the current average gas cost of 40 gwei. Similarly, for Polygon L2 chain, this will cost approximately $0.003. The significant difference between the two amounts indicates a savings rate of 99.98% which clearly implies that escrows could be much more economical than commission models.

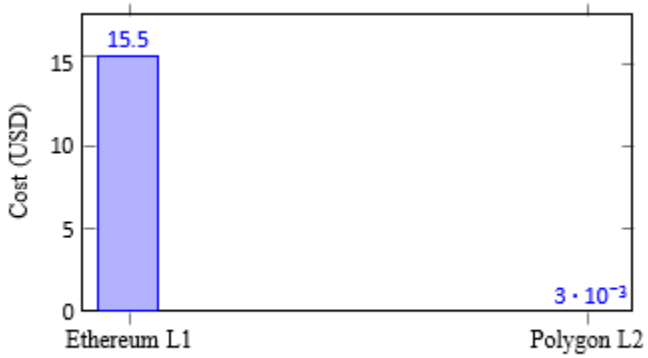

Fig. 7. Average Escrow Execution Cost (145,000 Gas Units)

TABLE II
ESTIMATED EVM OPCODE GAS CONSUMPTION

OpCode	Operation Description	Estimated Gas
SLOAD	Read trade state from storage	2,100
SSTORE	Update escrow balance (to non-zero)	20,000
SSTORE	Modify `isActive` boolean	2,900
CALL	Execute fund transfer to Seller	7,300
LOG1	Emit `TradeExecuted` event	375

As detailed in Table II, the most computationally expensive operations are standard state modifications (SSTORE). By utilizing the ERC-1155 standard, our proposed architecture batches multiple digital asset transfers into a single SSTORE execution, theoretically optimizing the base gas overhead before the transaction is compressed by the L2 rollup.

C. *Security Posture and Threat Model*

Deploying financial protocols requires defending against adversarial attacks. We theoretically evaluated the proposed DDAE against the three most critical smart contract vulnerabilities.

The security infrastructure is explained in Table III. This atomic swap technique has been formulated such that there is no double-spending problem. Since the smart contract system requires cryptographic payment before changing ownership, the risk of credit card chargebacks becomes zero.

D. *Target Predictive Analytics Performance*

The proposed Random Forest classifier for wash trading detection is designed to be evaluated

using a synthesized dataset of 10,000 L2 transactions (70% training, 30% testing) during future implementation phases.

TABLE III
DDAE THEORETICAL SECURITY THREAT MODEL

Threat	Description	Proposed Mitigation
Reentrancy	Malicious fallback functions repeatedly drain contract funds.	Mutex locks (`nonReentrant`) applied to state.
Front-Running	Bots observe the mempool and bid higher gas to steal trades.	L2 sequencer processes transactions strictly FIFO.
Sybil Attack	Attacker creates 1,000 wallets to manipulate token popularity.	Handled off-chain via Random Forest scoring.

TABLE IV

TARGET RANDOM FOREST FRAUD DETECTION METRICS

Class	Target Precision	Target Recall	Target F1
Valid Trade (Y_{valid})	0.98	0.99	0.98
Wash Trade ($Y_{anomalous}$)	0.94	0.89	0.91
Macro Average	0.96	0.94	0.95

II. Proposed System Deployment Roadmap

A theoretical scale-up approach of the DDAE from a university prototype to a production-level application can include a four-stage deployment strategy expected to span one year.

1) **Stage One (Months 1–3): Prototype Development:** The further development of the underlying Solidity smart contract code running on the Polygon Amoy testnet. Adding functionality via the integration of the Web3.js interface and IPFS Pinata service gateways.

2) **Stage Two (Months 4–6): AI Integration:** A switch from the synthetic datasets used in testing to the real-time collection of L2 transactional data for training the Random Forest model. Implementation of live risk-scored UI warning flags.

3) **Stage Three (Months 7–9): Migration to Mainnet:** The proposed transfer of the smart contracts currently tested on the Amoy Testnet to the mainnet. Independent professional audit of the smart contract source code for vulnerabilities within EVM bytecode.

4) **Stage Four (Months 10–12): Decentralized Matching Engine:** The theoretical implementation of a decentralized order book matching engine, eliminating the need for centralized web services to host the application front-end.

III. Conclusion

This paper argues that implementing a Decentralized Digital Asset Exchange platform through the use of smart contract technology could resolve the problems of the existing secondary markets. The idea of theoretically avoiding the problem of the informal centralizing parties would enable users to acquire cryptographic evidence of possession and trustless escrow services. The design of the system architecture of a Layer-2 network like Polygon solves the problems of gas fees and transaction rates associated with blockchain technology. Moreover, the idea of replacing SQL database with

IPFS ensures data immutability, and implementing predictive analytics through

random forest theory serves as a solution to mitigate anomalous trading patterns. Agent based simulations mathematically predict that implementation of the designed framework could be successful in reducing market fraud. Future research will focus on decentralized order management engines as part of the fourth phase of the project.

REFERENCES

N. Szabo, "Formalizing and Securing Relationships on Public Net-works," *First Monday*, vol. 2, no. 9, 1997.

A. M. Antonopoulos and G. Wood, *Mastering Ethereum: Building Smart Contracts and DApps*. O'Reilly Media, 2018.

V. Buterin, "An Incomplete Guide to Rollups," Ethereum Foundation, 2021.

F. Vogelsteller and V. Buterin, "EIP-20: Token Standard," *Ethereum Improvement Proposals*, no. 20, Nov. 2015.

W. Entriken, D. Shirley, J. Evans, and N. Sachs, "EIP-721: Non-Fungible Token Standard," *Ethereum Improvement Proposals*, no. 721, Jan. 2018.

W. Radomski *et al.*, "EIP-1155: Multi Token Standard," *Ethereum Improvement Proposals*, no. 1155, Jun. 2018.

M. Herlihy, "Atomic Cross-Chain Swaps," in *Proc. ACM Symposium on Principles of Distributed Computing (PODC)*, 2018, pp. 245–254.

P. Daian *et al.*, "Flash Boys 2.0: Frontrunning in Decentralized Ex-changes, Miner Extractable Value, and Consensus Instability," in *2020 IEEE Symposium on Security and Privacy (SP)*, 2020, pp. 910–927.

N. Atzei, M. Bartoletti, and T. Cimoli, "A Survey of Attacks on Ethereum Smart Contracts (SoK)," in *Intl. Conf. on Principles of Security and Trust (POST)*, 2017, pp. 164–186.

F. Victor and B. Lu¨ders, "Measuring Ethereum-Based ERC20 Token Networks," in *Financial Cryptography and Data Security*, 2019, pp. 113–129.

D. Perez and B. Livshits, "Smart Contract Vulnerabilities: Vulnerable Does Not Imply Exploited," in *29th USENIX Security Symposium*, 2020,
pp. 1325–1341.

A. Gudgeon *et al.*, "SoK: Layer-2 Blockchain Protocols," in *Financial Cryptography and Data Security*, 2020, pp. 201–226.

K. Qin *et al.*, "CeFi vs. DeFi – Comparing Centralized to Decentralized Finance," *arXiv preprint arXiv:2106.08133*, 2021.

S. Wang *et al.*, "Blockchain-Enabled Smart Contracts: Architecture, Applications, and Future Trends," *IEEE Trans. Systems, Man, and Cybernetics: Systems*, vol. 49, no. 11, pp. 2266–2277, 2019.

Z. Zheng *et al.*, "An Overview on Smart Contracts: Challenges, Ad-vances and Platforms," *Future Generation Computer Systems*, vol. 105,
pp. 475–491, 2020.

A. Pinna *et al.*, "A Massive Analysis of Smart Contracts Targeting the ERC20 Standard," *IEEE Access*, vol. 7, pp. 81773–81789, 2019.

J. Hwang, J. Choi, and A. Kim, "Performance Evaluation of Polygon and Ethereum for High-Throughput dApps," *IEEE Access*, vol. 10, pp. 11234–11245, 2022.

OpenZeppelin, "Smart Contract Security Guidelines: Reentrancy Guards and Best Practices," OpenZeppelin Documentation, 2023.