iJRASET
International Journal For Research in Applied Science and Engineering Technology

# INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

**Volume:** 14 **Issue:** IV **Month of publication:** April 2026

**DOI:** https://doi.org/10.22214/ijraset.2026.80978

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
*ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538*
*Volume 14 Issue IV Apr 2026- Available at www.ijraset.com*

# Venere: Blockchain-Based Escrow Protocol for Luxury Resale

Anjali Dubey[1], Heer Mehta[2], Aryaman Deolia[3], Dr. Ragupathi T[4]
*Department of Data Science and Business Systems, SRM Institute of Science and Technology, Kattankulathur, Chennai, India*

***Abstract:*** ***Venere was designed and implemented as a domain specific blockchain escrow protocol for luxury resale marketplaces. The system consists of three interconnected smart contracts—ProductNFT, Marketplace, and Escrow—deployed on a local Ethereum blockchain using Hardhat. A complete frontend was developed using Next.js, React, Chakra UI, and ethers.js, with MetaMask wallet integration for user interaction. IPFS metadata storage via Pinata enables persistent product information while maintaining on-chain references. Functional validation was performed through comprehensive automated testing, covering all escrow scenarios including successful completion, buyer disputes, and admin resolutions. Gas usage was measured for all core operations, with mintProduct requiring 304,693 gas units and escrow operations ranging from 47,066 to 141,427 gas units. Escrow fairness was demonstrated through testing of all dispute resolution mechanisms, confirming that funds remain locked until proper authorization is received.***

***Keywords: Blockchain, Smart Contracts, Escrow Protocols, Luxury Goods Resale, Fair Exchange, Ethereum, Decentralized Marketplaces.***

## I. INTRODUCTION

The resale market for luxury goods has grown significantly in recent years, driven by increased consumer interest in sustainability, affordability, and access to premium brands. However, trust remains a fundamental challenge in secondhand luxury transactions. Buyers must rely on sellers' claims regarding authenticity and product condition, while sellers must trust that buyers will complete payments without fraudulent disputes. These issues are amplified in high-value transactions where the financial stakes are substantial. Prior studies highlight that perceived risk and lack of trust are among the primary barriers to participation in luxury resale markets [1], [2], [25]. Traditional escrow services have been used to mitigate these risks by acting as intermediaries that hold funds until both parties fulfill transaction requirements. However, centralized escrow systems introduce new limitations. These services rely on trusted third parties to manage funds, resolve disputes, and maintain transaction records. As a result, they can introduce additional transaction costs, processing delays, and potential single points of failure. Moreover, centralized platforms often lack transparent audit trails, making it difficult for participants to independently verify transaction outcomes or dispute resolutions [3]. Blockchain technology provides an alternative approach for implementing trust mechanisms in digital marketplaces. Smart contracts enable programmable agreements that automatically execute predefined rules once specified conditions are satisfied. Early work on smart contracts envisioned automated digital agreements capable of reducing reliance on centralized intermediaries [4], while modern blockchain platforms such as Ethereum provide the infrastructure necessary to deploy such applications in practice [5], [8]. These capabilities have led to increasing research interest in blockchain-based marketplace systems that enable transparent, decentralized transaction management [20], [21].

In parallel, non-fungible tokens (NFTs) have emerged as a mechanism for representing unique digital or physical assets on blockchain networks. NFTs allow ownership and metadata of individual assets to be recorded on-chain, enabling traceable provenance and verifiable asset history [10], [23], [24]. In the context of luxury resale, NFT-based representations can provide buyers with immutable records of product attributes such as brand, production year, and serial number, thereby supporting authenticity verification and ownership tracking. This paper presents Venere, blockchain-based escrow protocol designed specifically for luxury resale marketplaces. The system combines NFT-based product representation with automated escrow mechanisms implemented through smart contracts. Luxury items are represented using ERC-721 tokens containing structured metadata, while marketplace and escrow contracts manage listings, payments, and dispute resolution processes. Funds remain locked within the escrow contract until either the buyer confirms successful delivery or a designated administrator resolves a dispute.

The Venere system was fully implemented using Solidity smart contracts deployed on a local Ethereum blockchain. A web-based frontend built with Next.js and React enables users to mint product NFTs, create listings, participate in auctions, and manage escrow transactions through a MetaMask wallet interface.

©IJRASET: All Rights are Reserved 

Metadata associated with each NFT is stored on the InterPlanetary File System (IPFS), providing persistent off-chain storage while maintaining on-chain references. The implemented system was experimentally validated using automated test suites covering both marketplace operations and escrow workflows. These tests include successful transaction completion, dispute initiation, and administrative resolution scenarios. Gas consumption was also measured for core smart contract functions, providing empirical insights into the computational costs associated with each operation.

The primary contribution of this work is the design and implementation of a domain-specific escrow protocol tailored to luxury resale transactions. By integrating NFT-based asset representation with programmable escrow logic, Venere demonstrates how blockchain technology can support transparent and secure exchange mechanisms for high-value goods while minimizing reliance on centralized intermediaries.

## II. RELATED WORK

Blockchain technology has enabled the development of decentralized marketplaces that facilitate peer-to-peer asset exchange without centralized intermediaries. Research in this area has explored how distributed ledger systems can support transparent transaction records, automated payment mechanisms, and programmable trust through smart contracts [21]. Smart contracts, originally conceptualized as self-executing digital agreements, enable contractual conditions to be encoded directly into software and executed automatically once predefined requirements are satisfied [4], [5], [8]. These capabilities have motivated the development of blockchain based marketplace architectures that aim to improve trust and transparency in online commerce.

Non-fungible tokens (NFTs) have emerged as a mechanism for representing unique assets on blockchain networks. Unlike fungible cryptocurrencies, NFTs allow individual items to be associated with distinct identifiers and metadata, enabling ownership verification and asset provenance tracking. Recent studies have examined the use of NFTs for digital asset marketplaces and ownership management systems [10], [23], [24]. These approaches demonstrate how blockchain-based tokens can represent physical or digital assets while maintaining immutable records of ownership transfers. However, many NFT marketplaces focus primarily on digital collectibles and artwork, providing limited support for transaction mechanisms tailored to high-value physical goods.

Escrow mechanisms have long been studied as a method for enabling fair exchange between mutually distrusting parties. Early cryptographic research introduced protocols designed to ensure that neither party could gain an unfair advantage during digital asset exchange [26], [27]. Later work proposed the use of semi-trusted third parties to mediate transactions while minimizing the need for complete trust in the intermediary [11]. More recent studies have explored blockchainbased implementations of fair exchange protocols that replace centralized mediators with automated smart contract logic [12], [29]. These approaches demonstrate the feasibility of implementing escrow and conditional payment systems directly on blockchain networks.

Despite these advances, blockchain-based escrow implementations face several security challenges. Prior work has identified vulnerabilities in smart contract implementations, including reentrancy attacks, improper access control, and logic errors that can lead to unintended fund transfers [13], [14], [15]. High-profile incidents such as the DAO exploit illustrate the risks associated with poorly designed smart contracts and highlight the importance of secure contract development practices [16]. Tools and frameworks for static analysis and automated verification have therefore been developed to improve the security of deployed contracts [17], [18], while development guidelines such as those provided by OpenZeppelin promote secure contract design patterns [19].

Blockchain technology has also been explored in the context of supply chain management and product authenticity verification. Prior studies demonstrate how distributed ledgers can be used to track product ownership and lifecycle information, enabling transparent provenance records for physical goods [9], [23]. In luxury markets, blockchain-based authentication systems have been proposed to combat counterfeit products and enable digital product passports that record ownership and authenticity data [22], [24]. These approaches highlight the potential of blockchain technology for maintaining verifiable records of product identity and transaction history. Although existing research provides important foundations for decentralized marketplaces, escrow protocols, and NFTbased asset management, relatively few systems integrate these components into a unified architecture tailored for luxury resale transactions. Many NFT marketplaces prioritize digital asset trading, while blockchain escrow protocols often focus on cryptocurrency exchange rather than physical goods markets. As a result, challenges specific to luxury resale—such as authenticity verification, high-value payment protection, and detailed product metadata management—remain insufficiently addressed by existing implementations. The Venere system builds upon these research directions by combining NFT-based asset representation with an integrated escrow protocol designed specifically for luxury resale marketplaces. By incorporating structured product metadata, programmable escrow logic, and automated dispute handling within a unified smart contract architecture, Venere demonstrates how blockchain technology can support secure and transparent exchange mechanisms for high-value physical goods while maintaining decentralized transaction management.

## III. SYSTEM DESIGN

### A. Overall Architecture

The Venere system follows a modular architecture consisting of three primary layers: blockchain-based smart contracts, decentralized metadata storage, and a web-based frontend interface. Smart contracts deployed on an Ethereum-compatible blockchain implement ownership management, marketplace operations, and escrow functionality. Off-chain product metadata is stored using the InterPlanetary File System (IPFS), while a web application enables users to interact with the system through a MetaMask wallet interface.

This architecture separates concerns between on-chain logic and off-chain data storage. Blockchain components maintain the authoritative state of asset ownership and transaction execution, ensuring immutability and transparency through distributed ledger technology [21]. In contrast, large data objects such as product images and descriptive metadata are stored offchain on IPFS, with on-chain references maintained through tokenURI fields. This hybrid design balances cost efficiency with data persistence, reducing the gas costs associated with storing large data structures directly on the blockchain [9], [24].

The frontend layer operates as a client application that communicates with deployed contracts using the ethers.js library. User authentication and transaction signing are handled through Web3React integration with MetaMask, allowing users to interact securely with blockchain-based applications. Through this architecture, the system maintains consistent synchronization between blockchain state and the user interface.

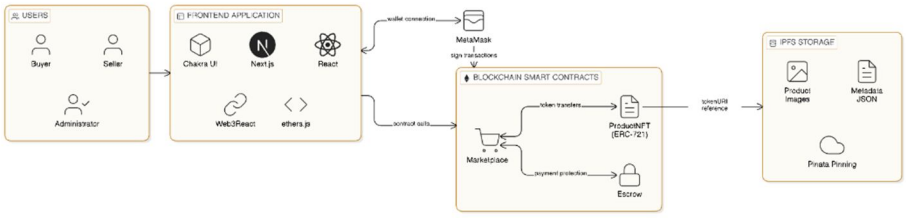



Fig. 1. System Architecture of the Venere Blockchain Escrow Marketplace

### B. ProductNFT Contract

The ProductNFT contract represents luxury items as nonfungible tokens using the ERC-721 standard. NFTs provide a mechanism for representing unique assets on blockchain networks while maintaining verifiable ownership records and immutable transaction histories [10], [23]. In the Venere system, each token corresponds to a specific luxury item and contains structured metadata describing its attributes.

The metadata schema includes fields such as product name, brand, category, production year, condition, and serial number. Serial number uniqueness is enforced through a mapping that prevents duplicate entries across different tokens, reducing the risk of counterfeit product listings. This design enables buyers to verify asset attributes through publicly accessible blockchain records.

The tokenURI field stores references to IPFS content identifiers following the ipfs://CID format. Metadata associated with each token contains additional product details including images and descriptive attributes. Storing metadata off-chain while maintaining on-chain references enables persistent product information without incurring excessive storage costs.

The contract implements standard ERC-721 functions including token minting, ownership transfers, and approval mechanisms, ensuring compatibility with existing blockchain wallets and infrastructure.

### C. Marketplace Contract

The Marketplace contract provides trading functionality for NFT-based luxury items. The contract supports two transaction modes: fixed-price listings and auctions. Fixed-price listings allow buyers to purchase items immediately through a buyProduct function that validates payment and transfers token ownership from seller to buyer.

Auction listings enable competitive bidding within a defined time window. The contract maintains the highest bid and associated bidder information, ensuring that each new bid exceeds the previous one. When a higher bid is placed, the previous bidder's funds are automatically refunded. This mechanism ensures fairness in the auction process and prevents funds from remaining locked after a participant is outbid.

To support platform sustainability, a small transaction fee of 0.25% is deducted from completed sales. Seller proceeds are accumulated in a mapping and can be withdrawn using a dedicated function, enabling sellers to collect earnings from multiple transactions. Access control checks ensure that only the original seller can manage or withdraw earnings associated with their listings.

### D. Escrow Contract

The Escrow contract implements conditional payment mechanisms that protect both buyers and sellers during luxury resale transactions. Escrow protocols are commonly used in digital marketplaces to ensure fair exchange between mutually distrustful parties [11], [26]. In the Venere system, escrow functionality is implemented through a smart contract that locks funds until predefined conditions are satisfied. When a buyer initiates a transaction, the createTransaction function generates a unique transaction identifier derived from transaction parameters using cryptographic hashing. Funds transferred during this operation remain locked within the escrow contract until the transaction reaches a valid resolution state.

The protocol supports three primary operations: transaction completion, dispute initiation, and dispute resolution. Buyers may finalize a transaction by confirming successful delivery, which triggers the release of funds to the seller after deduction of the platform fee. Alternatively, buyers can initiate a dispute if the delivered item does not meet expectations. In disputed cases, an administrator resolves the transaction by either refunding the buyer or releasing funds to the seller.

Event logging ensures that all escrow state transitions are recorded on-chain. These events allow frontend applications to monitor transaction progress and update user interfaces accordingly.

### E. Smart Contract Interaction Model

The three contracts operate through a modular interaction model that separates asset representation, marketplace functionality, and escrow management. The ProductNFT contract manages token ownership and metadata, while the Marketplace contract handles trading operations such as listing creation and bid management. The Escrow contract manages conditional payment flows independently of token ownership. This modular architecture improves maintainability and testing by isolating responsibilities within separate contracts. Contract interactions occur through token transfers and event emissions, enabling coordinated operation without tightly coupling contract logic.

Event-driven communication plays an important role in system coordination. For example, token ownership transfers emit standard ERC-721 Transfer events, while marketplace and escrow contracts emit events describing listing creation, transaction completion, and dispute resolution. These events allow frontend components to retrieve state changes without continuously polling contract storage.

### F. Frontend Architecture

The frontend application was implemented using the Next.js framework with React-based components and the Chakra UI design system. The interface enables users to mint product NFTs, create listings, participate in auctions, and manage escrow transactions through a web-based interface. Blockchain communication is handled through the ethers.js library, which provides interfaces for interacting with deployed smart contracts. Web3React manages wallet connections and account switching, enabling MetaMask integration for secure transaction signing. Frontend components are organized into reusable modules including product display cards, bidding interfaces, purchase dialogs, and escrow dashboards. React state management ensures that user interfaces update dynamically based on blockchain events and transaction status updates.

### G. IPFS Metadata Storage

Product metadata is stored off-chain using the InterPlanetary File System. IPFS enables decentralized storage through content-addressable identifiers, allowing data to be retrieved through distributed nodes without reliance on centralized infrastructure [9], [24]. Metadata files follow a JSON schema containing product attributes, images, and authenticity information. Each file is uploaded to IPFS through the Pinata pinning service, which ensures persistent availability of stored content. The corresponding IPFS content identifier is stored within the tokenURI field of the NFT, linking on-chain tokens to their associated metadata. By separating metadata storage from blockchain state, the system reduces gas consumption while maintaining verifiable references to product information.

## IV. ESCROW PROTOCOL DESIGN

### A. Escrow Motivation

Escrow mechanisms play an important role in enabling secure transactions between mutually distrusting parties. In luxury resale marketplaces, the high monetary value of items increases the risk associated with remote transactions. Buyers must trust that products will be delivered as described, while sellers must rely on buyers to complete payments without fraudulent disputes. Prior research on online resale platforms identifies trust and perceived risk as major barriers to participation in secondary markets for luxury goods [1], [2], [25].

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

Traditional escrow services address these issues by acting as trusted intermediaries that temporarily hold funds until both parties fulfill their obligations. However, centralized escrow systems introduce additional costs, operational delays, and dependency on third-party institutions. Blockchain technology provides an alternative approach in which conditional payment logic can be enforced automatically through smart contracts. Fair exchange protocols and blockchain-based escrow mechanisms have been proposed to ensure that neither party gains an unfair advantage during a transaction [11], [12], [26]. By embedding escrow functionality within smart contracts, decentralized systems can reduce reliance on centralized intermediaries while maintaining transparent and verifiable transaction records.

### *B. Escrow Transaction Lifestyle*

The Venere escrow protocol implements a structured transaction lifecycle that governs how payments are managed during a luxury resale transaction. Each transaction progresses through a set of predefined states that ensure funds remain protected until the transaction reaches a valid resolution.

Transactions begin in an Active state when a buyer initiates an escrow payment. During this phase, funds are locked within the escrow contract and cannot be accessed by either party. The transaction may subsequently transition to one of two states depending on user actions. If the buyer confirms successful delivery of the product, the transaction moves to a Completed state and funds are released to the seller. Alternatively, the buyer may initiate a dispute if the transaction conditions are not satisfied.

Disputed transactions require administrative intervention to determine the final outcome. After review, the transaction transitions to a Resolved state in which funds are either refunded to the buyer or released to the seller. This structured lifecycle prevents unauthorized or ambiguous transaction states while ensuring that each escrow agreement follows a deterministic execution path.

### *C. Transaction Creation*

Escrow transactions are initiated through the createTransaction function, which receives payment from the buyer and establishes a new escrow agreement. The function verifies that the transferred payment amount is valid and that the seller address is properly specified before creating the transaction. To uniquely identify each escrow agreement, the contract generates a transaction identifier using a cryptographic hash derived from key parameters such as buyer address, seller address, transaction amount, timestamp, and product identifier. This approach provides deterministic identification while preventing collisions between transactions.

Once the transaction is created, the transferred funds are locked within the escrow contract. Transaction details are stored in contract storage and associated with the generated identifier. The contract emits a TransactionCreated event that records the participants and transaction amount, enabling external applications to track escrow activity.

### *D. Transaction Completion*

Transaction completion occurs when the buyer confirms successful receipt of the purchased item. The completeTransaction function allows the buyer to finalize the transaction and release the escrowed funds to the seller.

Access control mechanisms ensure that only the buyer associated with the transaction can execute this operation. When the function is called, the contract calculates a platform fee equal to 0.25% of the transaction value and deducts this amount from the escrowed funds. The remaining balance is transferred directly to the seller.

After the transfer is executed, the transaction state is updated to Completed, and a TransactionCompleted event is emitted. This event provides a permanent record of successful transaction settlement and allows the frontend interface to update transaction status accordingly.

### *E. Dispute Mechanism*

In cases where buyers believe that transaction conditions have not been satisfied, the protocol allows disputes to be initiated through the disputeTransaction function. This operation verifies that the caller is the buyer associated with the escrow agreement and that the transaction remains in the Active state.

Once a dispute is initiated, the transaction state transitions to Disputed, and a TransactionDisputed event is emitted. The escrowed funds remain locked during this stage, preventing either party from accessing the payment until the dispute is resolved.

### *F. Dispute Resolution*

Disputed transactions are resolved through the resolveDispute function, which is restricted to an administrator account. Administrative resolution provides two possible outcomes: refunding the buyer or releasing funds to the seller.

If the buyer is refunded, the entire escrowed amount is returned without any platform fee deduction. Alternatively, if the dispute is resolved in favor of the seller, the contract deducts the platform fee before transferring the remaining amount to the seller address. In both cases, the transaction state transitions to Resolved, and a DisputeResolved event records the outcome.

This resolution mechanism provides structured dispute handling while ensuring that funds remain secure until a valid decision is reached.

### *G. Fairness Guarantees*

The escrow protocol enforces fairness through strict access control and deterministic state transitions. Only buyers are permitted to finalize transactions or initiate disputes, preventing sellers from prematurely accessing escrowed funds. Conversely, sellers cannot interfere with dispute initiation or transaction completion.

Administrative dispute resolution is restricted through the Ownable access control pattern, ensuring that only authorized parties can resolve disputes. These constraints ensure that escrowed funds cannot be withdrawn unilaterally and that each transaction reaches a resolution through explicitly defined protocol rules.

By combining state validation, access control, and automated execution through smart contracts, the Venere escrow protocol provides a transparent mechanism for managing conditional payments in luxury resale transactions.

## V. IMPLEMENTATION AND EXPERIMENTAL RESULTS

### *A. Implementation Environment*

The Venere system was implemented using Solidity version 0.8.17 and the Hardhat development framework, which provided compilation, deployment, and automated testing capabilities. Smart contracts were executed on a local Ethereum blockchain (Chain ID 1337), enabling deterministic execution and controlled testing conditions. Using a local blockchain allowed consistent gas measurements and simplified debugging during system development.

User interaction with the blockchain was facilitated through MetaMask, which handled wallet management and transaction signing. The frontend application was implemented using Next.js and React, enabling a modular component-based architecture for decentralized application interfaces. The Chakra UI design system was used to provide consistent user interface components and responsive layouts across application pages.

Blockchain communication was handled through the ethers.js library, which enabled contract interaction, transaction submission, and event monitoring. Contract instances were initialized using deployed addresses and associated ABIs, allowing the frontend to call read and write functions directly on the blockchain. This design ensures that the application retrieves product ownership, listing status, and escrow transaction data directly from on-chain state rather than relying on external services.

Product metadata was stored using the InterPlanetary File System (IPFS) with the Pinata pinning service. Metadata files describing luxury products were uploaded to IPFS and referenced on-chain through the tokenURI field of each NFT. This approach allows persistent storage of product attributes and images while minimizing blockchain storage costs [9], [24].

### *B. Functional Validation*

Functional correctness of the escrow protocol was validated through automated test scenarios executed within the Hardhat testing environment. The following escrow scenarios were designed to evaluate the behavior of the protocol under different transaction conditions.

- Scenario A – Successful Transaction Completion: This scenario verified that a buyer could create an escrow transaction, confirm delivery, and release funds to the seller. After buyer confirmation, the escrow contract transferred the payment to the seller address while deducting the platform fee. The test confirmed that the transaction state transitioned correctly from Active to Completed.
- Scenario B – Buyer Dispute with Refund: In this scenario, the buyer initiated a dispute using the disputeTransaction function. The administrator resolved the dispute by refunding the buyer. The full escrowed amount was returned to the buyer without deducting the platform fee, validating the refund logic implemented in the smart contract.
- Scenario C – Dispute Resolved in Favor of Seller: This test simulated a case in which the buyer initiated a dispute but the administrator determined that the seller had fulfilled the transaction conditions. The contract released funds to the seller after deducting the platform fee, demonstrating that both dispute outcomes were correctly supported.
- Scenario D – Invalid Transaction Transitions: This scenario tested invalid actions such as unauthorized transaction completion and duplicate dispute attempts. The smart contract correctly reverted these operations, confirming that state validation and access con-

trol mechanisms functioned as expected. All escrow scenarios executed successfully, confirming that the protocol correctly enforced transaction state transitions and prevented unauthorized fund movements.

- Scenario E – Direct Marketplace Purchase: This scenario validated the direct purchase flow in the marketplace contract. The buyer executed the buyProduct function, resulting in successful NFT ownership transfer from seller to buyer and automatic deactivation of the listing. The test confirmed correct interaction between the Marketplace and ProductNFT contracts.
- Scenario F – Auction Bidding Flow: This scenario validated the auction mechanism by placing multiple bids on an auction listing. The smart contract correctly enforced the rule that each new bid must exceed the previous highest bid. When a higher bid was submitted, the previous highest bidder was automatically refunded, confirming proper auction behavior.

Table 1. Smart Contract Functional Testing Scenarios

| Scenario | Description | Functions Tested | Result |
|---|---|---|---|
| Scenario A | Successful flow including NFT minting, listing creation, purchase, escrow creation, and buyer confirmation | mintProduct, createListing, buyProduct, withdrawEarnings, createTransaction, completeTransaction | Passed |
| Scenario B | Buyer initiates dispute and administrator refunds buyer | createTransaction, disputeTransaction, resolveDispute (refund buyer) | Passed |
| Scenario C | Buyer dispute resolved in favor of seller | createTransaction, disputeTransaction, resolveDispute (release to seller) | Passed |
| Scenario D | Invalid transaction actions revert correctly | completeTransaction, disputeTransaction | Passed |
| Scenario E | Direct purchase successfully transfers NFT ownership | createListing, buyProduct | Passed |
| Scenario F | Auction listing with multiple bids and automatic refund of previous bidders | createListing, placeBid | Passed |

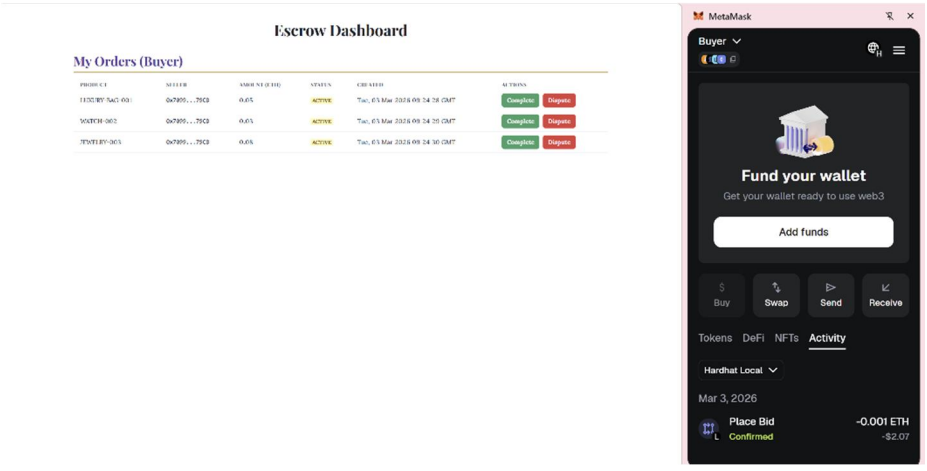



Fig. 2. Escrow dashboard illustrating transaction status and dispute resolution actions

### C. Marketplace Validation Function

Marketplace functionality was validated through six additional test scenarios covering direct sales and auction-based transactions. Direct purchase tests confirmed that NFT ownership transfers correctly when a buyer executes the buyProduct function. The marketplace contract verified payment amounts and transferred token ownership only after successful transaction confirmation.

Auction functionality was tested by creating auction listings and placing multiple bids. The contract correctly enforced the rule that each new bid must exceed the current highest bid. When a new bid was accepted, the previous highest bidder was automatically refunded, ensuring that funds were not locked within the contract.

Additional tests verified that invalid bidding attempts were rejected. Bids lower than the current highest bid and bids placed on non-auction listings were both reverted by the contract logic. Seller earnings accumulation and withdrawal functions were also validated, confirming that sellers could retrieve proceeds from completed sales after platform fee deduction.

These marketplace tests demonstrate that both fixed-price sales and auction mechanisms operate correctly within the integrated marketplace architecture.

### D. Gas Cost Analysis

Gas consumption for core smart contract operations was measured during automated test execution. The measurements provide insight into the computational cost of executing different functions within the system. The mintProduct operation required 304,693 gas, reflecting the computational overhead associated with creating ERC-721 tokens and storing associated metadata references. Marketplace operations demonstrated moderate gas consumption, with createListing requiring 166,234 gas, createAuction requiring 173,393 gas, and buyProduct requiring 122,007 gas.

Auction interactions required additional computation due to bid validation and refund handling. The placeBid operation consumed 163,354 gas, reflecting the logic required to verify bid conditions and transfer funds to previous bidders.

Escrow-related operations demonstrated comparatively lower gas usage. The createTransaction function required 141,427 gas, while the completeTransaction operation consumed 77,712 gas. Dispute operations were relatively inexpensive, with disputeTransaction requiring 50,001 gas and resolveDispute consuming 47,066 gas for buyer refunds and 61,062 gas when releasing funds to the seller.

These values were recorded during test execution on the local blockchain environment and represent baseline computational costs for the implemented operations.

Table 2. Gas Usage of Core Smart Contract Operations

| Operation | Gas Used |
|---|---|
| mintProduct | 304,693 |
| createListing | 166,234 |
| createAuction | 173,393 |
| buyProduct | 122,007 |
| placeBid | 163,354 |
| createTransaction | 141,427 |
| completeTransaction | 77,712 |
| disputeTransaction | 50,001 |
| resolveDispute (refund buyer) | 47,066 |
| resolveDispute (release to seller) | 61,062 |

### E. Observed Security Properties

Several security properties were observed during system testing. Critical functions in the escrow contract were protected using the ReentrancyGuard mechanism to prevent recursive call attacks that could compromise escrow funds. Administrative functions were restricted using the Ownable access control pattern, ensuring that only authorized accounts could resolve disputes.

Permission checks were implemented to ensure that only buyers could complete transactions or initiate disputes, while only administrators could execute dispute resolution functions. Attempted violations of these rules during testing resulted in transaction reverts, confirming that the access control mechanisms operate correctly.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

Additionally, the escrow design prevents unilateral withdrawal of funds. Payments remain locked within the contract until either buyer confirmation or administrative resolution occurs. This ensures that neither party can access escrowed funds without satisfying the protocol's predefined conditions.

## VI. DISCUSSION

### A. *Practical Feasibility of Blockchain Escrow*

The experimental validation demonstrates that blockchainbased escrow mechanisms can be implemented using smart contracts while maintaining security and deterministic transaction behavior. Automated contract execution ensures that all participants interact under predefined rules, eliminating reliance on centralized intermediaries. Because contract state and transaction history are stored on the blockchain, both buyers and sellers can independently verify escrow activity and dispute outcomes.

Smart contracts enforce conditional payment logic directly at the protocol level. Funds remain locked within the escrow contract until predefined conditions are satisfied, preventing premature release and unauthorized withdrawal. This approach aligns with prior work on fair exchange protocols and automated escrow mechanisms [11], [26]. By encoding transaction rules in smart contract logic, the system reduces the need for trusted third parties while maintaining transparent and verifiable transaction processing.

### B. *Suitability of High-Value Luxury Transactions*

Luxury resale markets present unique trust challenges due to the high value of goods and the risk of counterfeit items. Blockchain-based ownership records provide an auditable method for tracking product provenance and transaction history. In the Venere system, luxury items are represented as ERC-721 tokens containing structured metadata such as brand, category, and serial number.

The NFT representation enables verifiable ownership tracking and supports authenticity verification mechanisms that are particularly relevant for luxury goods markets [10], [23]. Because luxury transactions typically involve high-value items, the additional computational cost associated with blockchain transactions becomes economically acceptable. The escrow protocol further strengthens buyer and seller protection by ensuring that payment is released only after delivery confirmation or administrative dispute resolution.

### C. *Trust Minimization Properties*

The system implements trust minimization through protocol-level enforcement of transaction rules. Escrow funds remain locked until either buyer confirmation or administrative dispute resolution occurs, preventing unilateral withdrawal by either party. Buyers cannot reclaim funds after confirming delivery, and sellers cannot access escrowed funds without satisfying completion conditions.

These properties create a balanced environment where participants rely on protocol guarantees rather than interpersonal trust. Blockchain transparency further strengthens accountability by ensuring that transaction histories and dispute outcomes remain publicly verifiable. This design aligns with the broader objective of decentralized systems to reduce reliance on centralized intermediaries while maintaining transaction fairness [21].

### D. *Modular Smart Contract Architecture*

Venere adopts a modular architecture consisting of three independent smart contracts: ProductNFT, Marketplace, and Escrow. Each contract manages a distinct aspect of the system, enabling clearer separation of responsibilities. The ProductNFT contract handles ownership representation, the Marketplace contract manages trading operations, and the Escrow contract implements conditional payment logic.

This modular approach simplifies development and testing because each component can be validated independently before integration. Modular smart contract architectures have been widely recommended in blockchain application design due to their improved maintainability and security isolation [20].

### E. *Integration of On-Chain and Off-Chain Data*

The system balances cost efficiency and data persistence by combining on-chain state management with off-chain metadata storage. Ownership information and transaction logic are maintained on-chain to ensure immutability, while detailed product metadata is stored on IPFS.

Using IPFS reduces blockchain storage costs while preserving persistent access to product information through contentaddressable identifiers [9], [24]. The tokenURI mechanism provides a reliable link between NFT tokens and off-chain metadata resources, enabling efficient retrieval of product details while maintaining blockchain verification capabilities.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

## VII. CONCLUSION

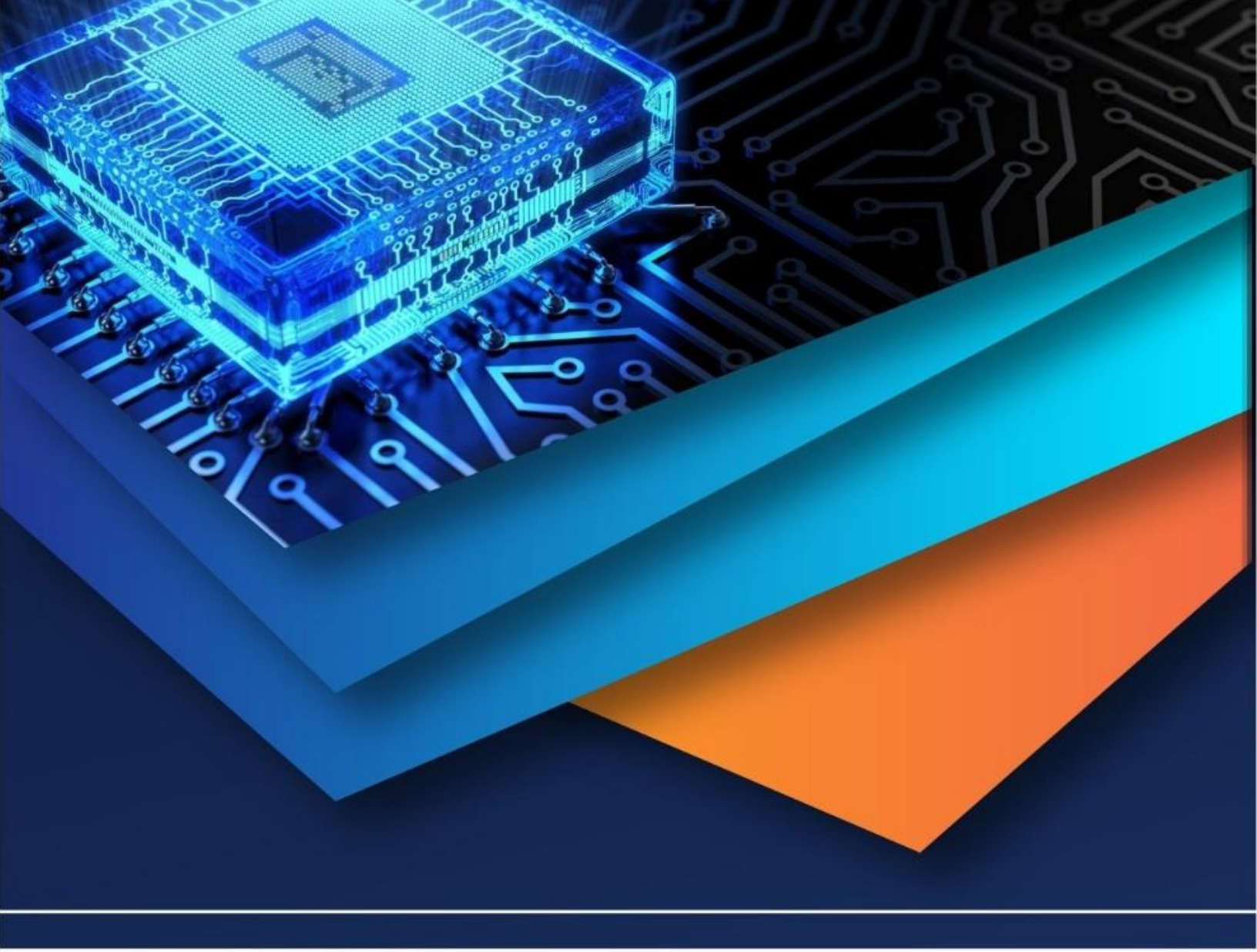
Venere was implemented as a complete blockchain-based escrow protocol specifically designed for luxury resale marketplaces. The system architecture consists of three interconnected smart contracts: ProductNFT for representing luxury goods as ERC-721 tokens with comprehensive metadata, Marketplace for facilitating both fixed-price and auction-based sales mechanisms, and Escrow for implementing conditional payments and dispute resolution. This modular design enables specialized functionality while maintaining clear separation of concerns between ownership representation, trading logic, and payment security.

The system was fully implemented and validated using the Hardhat development environment on a local Ethereum blockchain. Comprehensive automated testing validated all escrow flows, including successful completion, buyer disputes with refunds, and administrative resolutions favoring both buyers and sellers. Marketplace operations were successfully executed across all major functions including direct purchases, auction bidding, and seller earnings withdrawal. Invalid transaction states correctly reverted with appropriate error messages, confirming proper access control and state validation. Gas usage was measured for all core operations, providing empirical data for cost analysis and optimization assessment. Experimental validation confirmed that the escrow protocol behaves correctly under all tested scenarios, demonstrating secure fund locking mechanisms, reliable buyer confirmation processes, and effective dispute handling. The administratorbased resolution system operated as designed, providing fair outcomes for disputed transactions while maintaining proper access controls. All validation scenarios executed successfully, establishing that the smart contract logic operates deterministically and securely across various transaction conditions. The results demonstrate the practical feasibility of implementing domain-specific escrow protocols using smart contracts for luxury resale applications. The combination of NFTbased ownership records, comprehensive metadata storage, and automated dispute resolution provides a foundation for secure high-value transactions. The architecture successfully balances transparency, security, and efficiency while maintaining predictable transaction costs through measured gas consumption patterns. Future work may include deployment to public Ethereum test networks to validate performance under real network conditions, Layer-2 integration for reduced transaction costs, decentralized arbitration models for dispute resolution, and further optimization of smart contract gas usage. These enhancements would build upon the solid foundation established through the current implementation and validation, extending the system's capabilities while maintaining the core escrow protocol principles demonstrated in this research.

## REFERENCES

[1] B. Z. Li and C. Chen, "Second-hand luxury consumption: motivations and risks," Journal of Business Research, vol. 117, pp. 247–259, 2020.
[2] D. Turunen and J. Leipamaa-Leskinen, "Pre-loved luxury: identifying ¨ the meanings of second-hand luxury possessions," Journal of Product & Brand Management, vol. 24, no. 1, pp. 57–65, 2015.
[3] E. K. Clemons, "The future of online intermediaries," Journal of Management Information Systems, vol. 28, no. 3, pp. 43–72, 2011.
[4] N. Szabo, "Smart contracts," 1994. [Online].
[5] V. Buterin, "A next-generation smart contract and decentralized application platform," Ethereum White Paper, 2014.
[6] C. Delmolino et al., "Step by step towards creating a safe smart contract," in Proc. Financial Cryptography, 2016.
[7] M. Bartoletti and L. Pompianu, "An empirical analysis of smart contracts," in Proc. Financial Cryptography, 2017.
[8] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," Ethereum Yellow Paper, 2014.
[9] M. Casino et al., "Blockchain-based applications in supply chain," Sensors, vol. 19, no. 4, 2019.
[10] W. Ante, "Non-fungible token (NFT) markets on the Ethereum blockchain," Blockchain Research Lab, 2021.
[11] M. K. Franklin and M. Reiter, "Fair exchange with a semi-trusted third party," in Proc. ACM CCS, 1997.
[12] S. Dziembowski et al., "FairSwap: How to fairly exchange digital goods," in Proc. ACM CCS, 2018.
[13] N. Atzei, M. Bartoletti, and T. Cimoli, "A survey of attacks on Ethereum smart contracts," in Proc. POST, 2017.
[14] L. Luu et al., "Making smart contracts smarter," in Proc. ACM CCS, 2016.
[15] I. Nikolic et al., "Finding the greedy, prodigal, and suicidal contracts," in Proc. IEEE S&P, 2018.
[16] The DAO Attack Explained, IEEE Security Blog, 2016.
[17] M. Christodorescu et al., "Static analysis of executables," ACM Computing Surveys, 2007.
[18] P. Tsankov et al., "Securify: Practical security analysis of smart contracts," in Proc. ACM CCS, 2018.
[19] OpenZeppelin, "Secure smart contract development guidelines," 2022.
[20] Y. Wang et al., "A survey on blockchain-based marketplace systems," IEEE Access, vol. 8, pp. 12345–12360, 2020.
[21] S. Zheng et al., "An overview of blockchain technology," in Proc. IEEE BigData, 2017.
[22] Aura Blockchain Consortium, "Blockchain for luxury goods authentication," White Paper, 2020.
[23] K. Toyoda et al., "Product ownership management using blockchain," IEEE Access, vol. 5, pp. 17465–17477, 2017.
[24] D. Hughes et al., "Digital product passports and NFTs," Future Internet, vol. 13, no. 10, 2021.
[25] S. Kim and J. Kim, "Trust in online resale platforms," Electronic Commerce Research, 2020.
[26] S. Even et al., "Fair exchange protocols," Journal of Cryptology, vol. 15, no. 1, 2002.
[27] M. Abadi and J. Feigenbaum, "Secure circuit evaluation," Journal of Cryptology, 1990.
[28] J. Poon and T. Dryja, "The Bitcoin Lightning Network," 2016.
[29] T. Ruffing et al., "Optimistic fair exchange," IACR Cryptology ePrint, 2014.
[30] Y. Chen et al., "Gas cost analysis of Ethereum smart contracts," IEEE Access, vol. 8, pp. 144000–144015, 2020.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

# INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)