



Blockchain-Based Smart Contract Escrow for Secure and Transparent Property Transactions

**Omega Joel Patria
Moata***

Universitas Uyulindo,
Indonesia

Febi Elvira Messe

Universitas Uyulindo,
Indonesia

**Fiktor Imanuel
Tanesab**

Universitas Uyulindo,
Indonesia

***Corresponding author:**

Omega Joel Patria Moata, Universitas
Uyulindo, Indonesia. ✉jpatrick30@gmail.com

Article Info:

Article history:

Received: July 26, 2026

Revised: September 11, 2026

Accepted: September 18, 2026

Available Online: October 05, 2026

Keywords:

Blockchain; Property Escrow;
Smart Contract;
Security;
Transparency



This is an open access article under the
[CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.

Copyright © 2026 by Author. Published by
Politeknik Siber Cerdika International

Abstract

Background: Conventional property escrow depends on intermediaries and sequential verification, which can limit transaction visibility and create opportunities for delays, inconsistent records, and disputes. Blockchain-based smart contracts provide an auditable execution environment in which predefined conditions can automatically trigger transactions.

Objective: This study designs and tests a blockchain-based property escrow prototype and evaluates its functional contributions to transaction security and transparency.

Methods: A prototype-oriented design science approach was employed. A Solidity smart contract was developed in Remix IDE, deployed on a local Ethereum environment using Ganache, connected to user wallets through MetaMask, and monitored through transaction logs. Functional scenarios covered contract deployment, role-based interactions, fund-release logic, transaction recording, and verification of transaction hashes. The security assessment was limited to the implemented access-control and ledger-integrity mechanisms; no formal smart-contract security audit was conducted.

Results: The prototype successfully executed the defined escrow functions and generated blockchain records that were traceable through Remix and Ganache logs. Successful execution, reproducible transaction hashes, role-constrained contract calls, and persistent transaction histories were used as functional indicators of automation and transparency.

Conclusion: The prototype demonstrates the feasibility of an end-to-end blockchain-based escrow workflow; however, the findings do not establish absolute security or production readiness. Further work should include formal smart-contract security testing, performance benchmarking, regulatory analysis, and validation in real-world property transactions.

To cite this article: Moata, O. J. P., Messe, F. E., & Tanesab, F. I. (2026). Blockchain-Based Smart Contract Escrow for Secure and Transparent Property Transactions. *Journal of Business, Social and Technology*, 7(4), 2186–2201
<https://doi.org/10.59261/bustechno.v7i4.813>

INTRODUCTION

Property transactions are economic activities involving high financial values and multiple parties, including buyers, sellers, financial institutions, and escrow agents (Gadekallu, 2022). The complexity of these transactions requires a mechanism capable of ensuring fund security, transaction transparency, and the reliable execution of contractual obligations. In conventional practice, escrow services function as trusted third parties that hold funds until all transaction requirements have been satisfied. Although this mechanism has been widely adopted, traditional

escrow systems still face various challenges, including dependence on intermediaries, relatively high operational costs, time-consuming verification processes, and the potential for data manipulation and disputes arising from limited transparency throughout the transaction process. These issues have become increasingly important alongside the digitalization of property transactions and the growing need for systems capable of maintaining data integrity in near real time (Kushwaha et al., 2022a).

The development of blockchain technology offers a new paradigm for digital transaction management through characteristics such as decentralization, tamper resistance, transparency, and consensus mechanisms, which enable transaction records to be stored on a distributed ledger and verified according to the rules of the underlying blockchain network. One of the primary innovations enabled by blockchain technology is the smart contract, a computer program deployed on a blockchain that automatically executes predefined logic when specified conditions are satisfied (Saputra, 2024). Unlike conventional escrow mechanisms, which rely on manual intervention by third parties, smart contracts can automate functions such as conditional fund release and transaction-state recording, while external data or documents can be supplied through appropriate off-chain verification mechanisms. This automation can reduce reliance on manual processing, improve operational efficiency, and strengthen transaction traceability and accountability (Santhanalakshmi et al., 2023).

Figure 1 shows the workflow of the proposed blockchain implementation for addressing issues associated with property sales and purchase transactions.

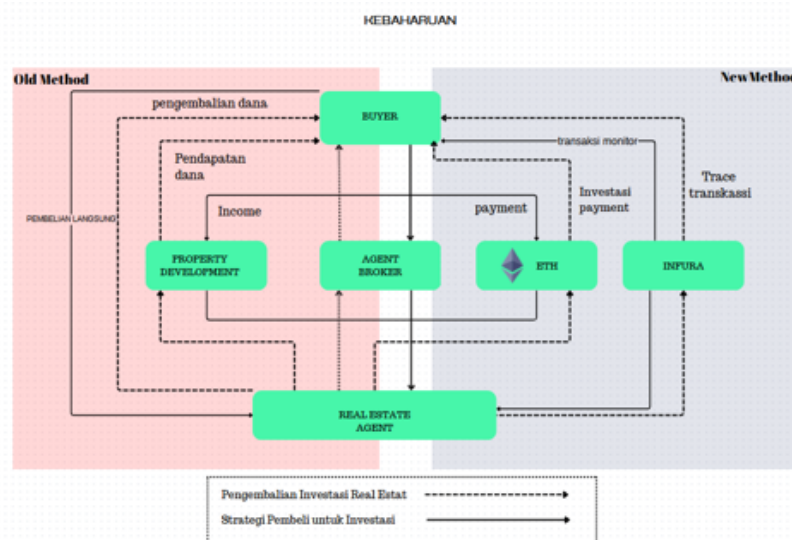


Figure 1. Workflow of the blockchain-based property escrow prototype.

This diagram illustrates the workflow of the smart contract implementation in a property escrow system designed to enhance transaction security and transparency. The stages are described as follows:

- 1) Agent (User): The process begins with an agent, who in this context may be a buyer, seller, or escrow agent. The agent interacts with the smart contract to initiate or manage the transaction according to the assigned role and access permissions.
- 2) Smart Contract: The smart contract is a computer program deployed on the blockchain that encodes the rules and conditions of the escrow agreement. It receives transaction inputs and executes predefined logic in accordance with those rules.
- 3) Analysis: The system analyzes the data submitted by the agent, such as the identities of the involved parties, transaction details, property-document status, and other required information. Because a smart contract cannot independently determine the validity of most real-world documents or identities, such information must be verified through trusted off-chain processes or appropriate external data sources before being used as an input to the contract.

- 4) Data Verification: Following the analysis, the system verifies whether the submitted information satisfies the predefined requirements of the escrow agreement. Where verification depends on off-chain information, the verified result is subsequently supplied to the smart contract as an input or authenticated data value.
- 5) Security: AES-256 may be used as an encryption mechanism to protect sensitive data stored or processed outside the blockchain. However, the security properties demonstrated by the blockchain prototype in this study are limited to role-based contract interaction, integrity of recorded transaction data and hashes, and the resistance of confirmed ledger entries to unilateral modification under the assumptions of the underlying blockchain network. These properties do not establish absolute system security and do not replace a formal smart-contract security audit, penetration test, or comprehensive assessment of off-chain components.
- 6) Blockchain: The blockchain serves as the underlying infrastructure of the system. It provides a distributed ledger for recording and tracking property escrow transactions, enabling transaction records and smart-contract state changes to be independently verified according to the capabilities and access model of the selected blockchain network.

In recent years, blockchain technology has been increasingly applied across various sectors, including financial services, supply chains, healthcare, and public administration (Gadekallu, 2022). Previous research has shown that blockchain can enhance the security and traceability of international business transactions through peer-to-peer mechanisms and the digitization of Letters of Credit (LC) Toorajipour et al. (2022) while other studies have developed Ethereum-based property registration systems to improve the integrity of ownership records (Zhang et al., 2024). Nevertheless, much of the existing research focuses on conceptual blockchain applications, network security, or the digitization of property ownership records (Chen et al., 2020; Chen & Zhang, 2019). Studies that specifically integrate property escrow mechanisms with smart contracts while implementing transaction workflows using the Ethereum ecosystem, Remix IDE, Ganache, MetaMask, and off-chain transaction monitoring systems remain relatively limited. Thus, a research gap remains regarding how blockchain can be operationally applied to automate key stages of the escrow process while simultaneously improving security, transparency, and efficiency in property transactions. The gap is therefore end-to-end rather than tool-specific: existing work often demonstrates property registration, security primitives, or isolated smart-contract functions, but provides less evidence on how escrow roles, conditional fund release, transaction traceability, and operational verification can be integrated into a single reproducible property-transaction workflow.

Besides its academic relevance, this research also has significant practical and social relevance. The high financial value of property transactions means that administrative errors, incomplete verification, or ownership disputes can result in substantial financial losses. At the same time, the increasing digital transformation of the property industry is driving demand for more trustworthy transaction mechanisms that provide auditable records and reduce opportunities for unauthorized data manipulation. Therefore, the development of a blockchain-based escrow system represents a potential technological approach for supporting safer, more transparent, and more efficient property transactions.

Based on this background, this research aims to develop and implement a blockchain-based property escrow system using smart contracts on the Ethereum platform. The implementation involves smart-contract development using Remix IDE, blockchain-network simulation using Ganache, integration with the MetaMask digital wallet, and transaction-recording mechanisms that can be verified through the blockchain. This research not only evaluates the technical implementation of the smart contracts but also analyzes how the application of blockchain can support transaction security, escrow-process transparency, and the efficiency of conditional fund release compared with conventional escrow approaches.

The main contribution of this research lies in the development of an implementation model for property escrow that integrates smart contracts with automated transaction-state verification within the Ethereum blockchain ecosystem. Unlike previous studies that have primarily discussed blockchain technology or smart contracts separately, this research presents an end-to-end implementation encompassing transaction-validation processes, role-based access

control for the involved parties, immutable activity recording, and conditional fund-release simulation through smart contracts. Thus, this research provides a conceptual contribution to the development of blockchain-based digital escrow systems while also providing implementation evidence that can serve as a reference for the future development of digital property-transaction services. Academically, the contribution is a functional escrow architecture that separates actor roles, encodes predefined release conditions, and makes transaction-state transitions auditable through immutable ledger records. This contribution is distinct from novelty based merely on the use of Ethereum, Remix, Ganache, or MetaMask.

METHOD

The research stages comprised: (1) identification of escrow actors and functional requirements; (2) design of the smart-contract architecture and transaction states; (3) Solidity implementation in Remix IDE; (4) deployment and wallet interaction in a local Ethereum network using Ganache and MetaMask; (5) scenario-based functional testing; and (6) interpretation of security and transparency based on access restrictions, contract execution results, and immutable transaction logs. No formal audit for reentrancy, arithmetic vulnerabilities, access-control bypass, front-running, or other smart-contract vulnerabilities was performed; this limitation constrained the security claims.

This study used a prototype-oriented design science research approach. The artifact was a blockchain-based property escrow prototype, and the evaluation focused on whether the artifact satisfied predefined functional requirements: actor-role separation, conditional fund release, transaction recording, hash-based traceability, and reproducible execution in the test environment.

1) Agent

The agent in this context could act as either a buyer or seller in the agreed property transaction mechanism. Escrow itself was a legal agreement in which an item, generally money but potentially any other object, was held by a third party called an escrow agent while awaiting fulfillment of the contract terms (Dimas, 2023). Meanwhile, the buyer and seller conducted transactions mediated by the escrow agent. The seller and buyer provided their digital wallets as transaction tools using MetaMask. MetaMask was used to connect users with Ethereum decentralized applications (Harlian, 2022). By using Ethereum, developers could create various types of applications, including property and real estate applications (Purba, 2024).

2) Ethereum

Ethereum enabled digital assets to be owned and traded and stored transaction records on a public ledger commonly known as a blockchain (Abdilah et al., 2023). By using a local Ethereum blockchain network simulated through Ganache Toorajipour et al. (2022), a simple prototype was created to evaluate how property transactions could be executed in a local network. This research implemented and tested the functionality of the proposed solution using Remix IDE, demonstrating that it could be implemented as a simple transaction prototype.

3) Remix Tools

A smart contract was developed in Remix IDE using Solidity and deployed in the Ethereum test environment. The prototype represented the principal escrow roles and transaction logic, while wallet interaction and transaction execution were observed through the configured test network. This impersonal formulation reflected the artifact-development process rather than the actions of an individual author.

4) Smart Contract

A smart contract was a computer program that ran on a blockchain and was executed in a decentralized manner through network nodes. In a software development environment, the concept of a smart contract referred to a program that enabled the automatic execution of specific predefined functions on a blockchain. In this sense, smart contracts had characteristics that distinguished them from other types of software (Bassan & Rabitti, 2024). For example, a segment of the escrow contract and its function related to the proposal for fund release to the recipient were presented as part of the blockchain implementation. This escrow smart contract included several main elements and mechanisms intended to support transaction security and

efficiency:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract PropertyEscrow {
    address public depositor;
    address public beneficiary;
    address public escrowAgent;

    struct SourceMaterial {
        string description;
        bool isReleased;
        address owner;
    }

    SourceMaterial public material;

    // Struktur untuk biaya escrow
    struct EscrowFee {
        uint256 amount;
        address payable recipient;
        bool isPaid;
    }

    EscrowFee[] public escrowFees;

    // Event untuk mencatat penambahan biaya
    escrow
    event EscrowFeeAdded(uint256 amount,
        address recipient);

    // Event untuk mencatat pembayaran biaya
    escrow
    event EscrowFeePaid(uint256 amount,
        address recipient);

    constructor(address _depositor, address
    _beneficiary, string memory _description) {
        depositor = _depositor;
        beneficiary = _beneficiary;
        escrowAgent = msg.sender;
        material = SourceMaterial({
            description: _description,
            isReleased: false,
            owner: _depositor
        });
    }

    modifier onlyEscrowAgent() {
        require(msg.sender == escrowAgent, "Only
        the escrow agent can perform this action.");
        _;
    }
}
```

```
modifier onlyDepositor() {
    require(msg.sender == depositor, "Only the
depositor can perform this action.");
    _;
}

modifier onlyBeneficiary() {
    require(msg.sender == beneficiary, "Only
the beneficiary can perform this action.");
    _;
}

// --- Fungsi untuk mengelola materi sumber
---

function returnMaterial() public
onlyEscrowAgent {
    require(material.isReleased == false,
"Material has already been released.");
    material.owner = depositor;
    emit MaterialReturned(depositor);
}

function releaseMaterial() public
onlyEscrowAgent {
    require(material.isReleased == false,
"Material has already been released.");
    material.isReleased = true;
    material.owner = beneficiary;
    emit MaterialReleased(beneficiary);
}

// --- Fungsi untuk menangani aspek hukum
---

event LegalNotification(address indexed
from, string message);

function notifyLegalAction(string memory
message) public onlyEscrowAgent {
    emit LegalNotification(msg.sender,
message);
}

event Indemnification(string message);

function indemnify(string memory message)
public {
    emit Indemnification(message);
}

event DisputeResolution(address indexed
from, string message);

function resolveDispute(string memory
```

```
message) public {
    emit      DisputeResolution(msg.sender,
message);
}

// --- Fungsi untuk mengelola biaya escrow ---
-

function  addEscrowFee(uint256  amount,
address    payable    recipient)  public
onlyEscrowAgent {
    escrowFees.push(EscrowFee(amount,
recipient, false));
    emit EscrowFeeAdded(amount, recipient);
}

function  payEscrowFee(uint256  index)
public payable {
    require(index < escrowFees.length, "Invalid
escrow fee index.");
    EscrowFee storage fee = escrowFees[index];
    require(fee.isPaid == false, "Escrow fee
already paid.");
    require(msg.value == fee.amount, "Incorrect
payment amount.");

    fee.recipient.transfer(fee.amount);
    fee.isPaid = true;
    emit      EscrowFeePaid(fee.amount,
fee.recipient);
}

// --- Fungsi untuk mencatat verifikasi materi
sumber ---

event  VerificationReportSubmitted(string
report);

function  submitVerificationReport(string
memory report) public onlyEscrowAgent {
    emit VerificationReportSubmitted(report);
}

// --- Fungsi untuk menandatangani kontrak
---

struct ContractSignature {
    string name;
    string title;
    string date;
    bool isSigned;
}

mapping(address => ContractSignature)
public signatures;
```

```

event ContractSigned(address indexed from,
string name, string title, string date);

function signContract(string memory name,
string memory title, string memory date)
public {
    require(msg.sender == depositor ||
msg.sender == beneficiary || msg.sender ==
escrowAgent, "Only parties involved can sign
the contract.");
    signatures[msg.sender] =
ContractSignature(name, title, date, true);
    emit ContractSigned(msg.sender, name,
title, date);
}

// --- Event untuk mencatat kejadian penting
---

event MaterialReturned(address depositor);
event MaterialReleased(address
beneficiary);
}

```

Figure 2. Property Escrow Smart Contract Implementation

After the smart contract implementation had been explained, a comparison between the conventional escrow mechanism and the blockchain-based escrow mechanism was presented through the sequence diagram shown in Figure 3.

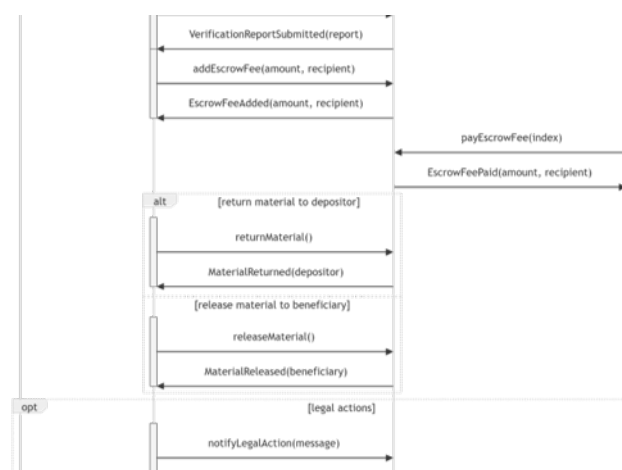


Figure 3. Sequence Diagram

The transaction in Figure 3 represented the mechanism of the proposed method for designing the smart contract, in which the main goal of the escrow contract was to ensure that transactions between two or more parties took place securely and efficiently. The escrow contract controlled the assets involved in the transaction and delivered them to the appropriate party according to the established requirements (Preethi et al., 2026). The Property Buyer Agent

created a goods transaction tender containing the material ID or the name and location of the house received from the Escrow Agent or the Ethereum ID. The focus of this dissertation was on developing applications on the Ethereum blockchain to address issues associated with outdated property ledgers. After that, both parties signed the contract according to the agreed terms, and the Escrow Agent reviewed the goods according to the applicable provisions before releasing the funds or property-related assets, such as land and building areas, facilities, and other terms specified by the seller. After confirming that the property met the specified requirements, it was transferred by the escrow agent to the developer or used to settle obligations based on the loan agreement, either during the contract period or as part of a permanent purchase. Both parties agreed to use a trusted escrow agent or service, such as Ethereum, to facilitate the transfer to the seller's account, such as a MetaMask account.

Since the transaction blockchain used in the prototype was Ethereum, the homeowner, buyer, and agent were required to have MetaMask wallets to proceed with the transaction. After these wallets had been configured, the funds provided could be used for the final fund disbursement, converted into Indonesian Rupiah, or transferred among Ethereum users such as sellers and buyers, while transaction fees could be deducted as gas fees for the selling and purchasing processes. Based on the implementation of the escrow-agent smart contract, an alternative flow was available in which the Escrow Agent returned the goods to the Depositor if a discrepancy with the agreement was identified by the buyer; however, such a process had to proceed through the appropriate legal channels. The debtor's activity in imposing mortgage rights on land in favor of the creditor was considered valid if it was carried out with the knowledge and approval of the landowner. This was in accordance with the provisions of Article 7 of the Mortgage Law (UUHT), which stated that mortgage rights would continue to attach to the object wherever the object was located.

The Escrow Agent was obliged to notify the Depositor of any legal violations or notify the relevant authorities, supported by transaction logs and descriptions of the results of the ongoing activities. This transparency was useful for participants exchanging documents and data through nodes in the decentralized blockchain system, where transaction records could be traced and protected against unauthorized alteration. The blockchain used timestamps to identify and record transactions, thereby providing a temporal dimension to transaction data. In terms of credibility, blockchain data transmission relied on participating nodes to maintain the integrity of the distributed system and provide resistance to external attacks without requiring centralized human intervention. Responses and indemnity agreements were handled according to the applicable transaction arrangements. Disputes were resolved according to the agreement governing the Property Escrow transaction. Smart-contract concepts had become increasingly influential in blockchain applications. InterLedger created a system that enabled value transfers between different blockchains within the system. This project primarily built a blockchain platform on which users could develop blockchain applications and create their own blockchain transactions.

Letters of Credit had been used for centuries to facilitate payments, particularly in international business transactions. An LC (Letter of Credit) was a relatively established mechanism consisting of an agreement between a seller and a buyer under terms agreed upon by both parties. This mechanism provided a basis for comparison with smart contracts on blockchain technology because both involved predefined contractual conditions. Such an agreement mechanism could be implemented in an off-chain system to monitor whether transactions complied with the agreed terms. The use of Ethereum and IPFS supported trust and security in the digital environment (Onwubiko et al., 2023), while Ethereum functioned as a blockchain platform (Chen et al., 2020; Onwubiko et al., 2023). According to information from the digital financial technology platform Coinbase, Ethereum ranked second in popularity and had a large and active community. This technological approach could potentially be utilized in the future to support large-value transactions between individuals and facilitate secure and modern transaction processes, as illustrated in Figure 4.

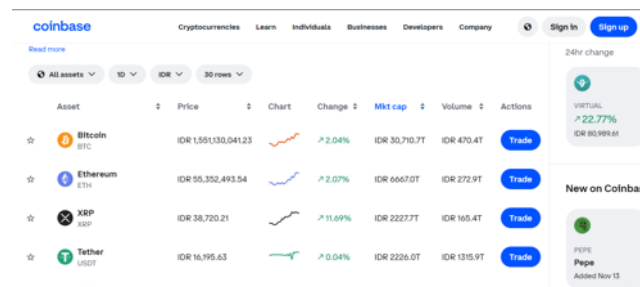


Figure 4. <https://www.coinbase.com/explore>

5) Security

The Ethereum blockchain had two main security characteristics relevant to this discussion: first, its permissionless nature, which allowed anyone to interact with the network and potentially attempt to exploit application-level vulnerabilities; and second, the immutability of confirmed blockchain records, which meant that erroneous or vulnerable smart-contract code could not simply be patched in place in the same manner as conventional software.

Although attacks at the application layer had caused significant financial losses, the resulting damage did not necessarily spread to host computers because of the isolation provided by the Ethereum Virtual Machine (EVM). Code reuse in smart contracts could also pose greater risks than in some traditional systems, emphasizing the importance of security audits for smart contracts and reused libraries. Figure 5 shows the Ethereum blockchain architecture and the environment in which the Ethereum blockchain operated. The environment consisted of four layers of the Ethereum architecture: web user interfaces for interacting with applications, databases for storing blockchain data, cryptographic mechanisms supporting consensus protocols, and Internet services supporting the network layer.

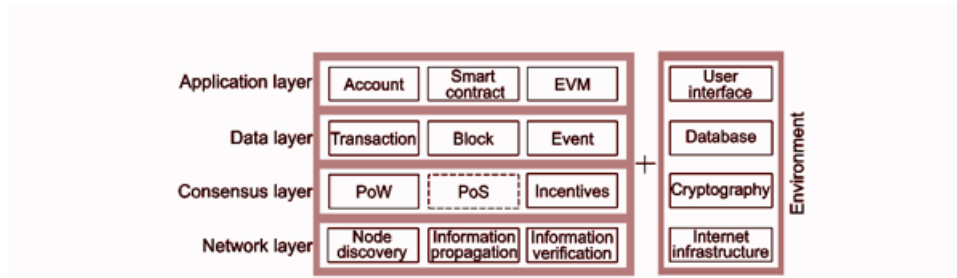


Figure 5. Ethereum blockchain architecture and the environment in which the Ethereum blockchain operates

Security analysis of the executed transactions was conducted using the final transaction log file, which served as evidence that the executed transactions had been recorded. Through these records, it was possible to identify the transactions that had been executed. This analysis was relevant to financial security within the blockchain ecosystem (Chen et al., 2020; Kushwaha et al., 2022b). With various updated and applied security measures, the study evaluated whether these mechanisms provided an adequate security layer for the escrow transactions under investigation. In addition, research on “Leveraging complex event processing for monitoring and automatically detecting anomalies in Ethereum-based blockchain networks” by Jesús Rosa-Bilbao, Juan Boubeta-Puig, Jesús Lagares-Galán, and Mark Vella in 2024 indicated that bugs in smart-contract programming could lead to security breaches and significant economic losses. Furthermore, attacks attempting to manipulate blockchain data could cause errors and other risks within the blockchain network. For this reason, monitoring blockchain data through off-chain systems was considered necessary for detecting anomalies and reducing risks associated with blockchain technology. Such systems could analyze transaction flows and interact with off-chain infrastructure such as Infura (Rosa-Bilbao et al., 2025).

6) Infura

Infura Ethereum RPC provided a standard RPC interface that allowed uPort to communicate with the Ethereum network. Infura IPFS provided a standard interface that allowed uPort to communicate with the IPFS network (Babel et al., 2025; Naik & Jenkins, 2020). In this study, Infura was used as an infrastructure service for interacting with the Ethereum network and IPFS. It provided access to blockchain and IPFS infrastructure for application interactions involving both mainnet and testnet environments.

RESULTS AND DISCUSSION

The prototype was evaluated using four functional indicators: (1) successful deployment and execution of the defined escrow functions; (2) enforcement of the roles and conditions encoded in the smart contract; (3) generation of transaction hashes that can be independently traced in the test environment; and (4) persistence of the transaction history in the local blockchain ledger. These indicators demonstrate functional operation and auditability, but they should not be interpreted as a comprehensive cybersecurity assessment.

The results of the transaction mechanism executed on the testnet are shown in Figures 6, 7, and 8.



Figure 6. Deployment of the property escrow smart contract in Remix IDE



Figure 7. Executed smart-contract transaction and deployment status in Remix IDE

Figure 7 displays the smart-contract deployment. Within the Remix IDE development environment, several transaction trials were conducted using the smart-contract mechanism described earlier. The transactions executed successfully, and the transaction results can be viewed in the Remix IDE logs.

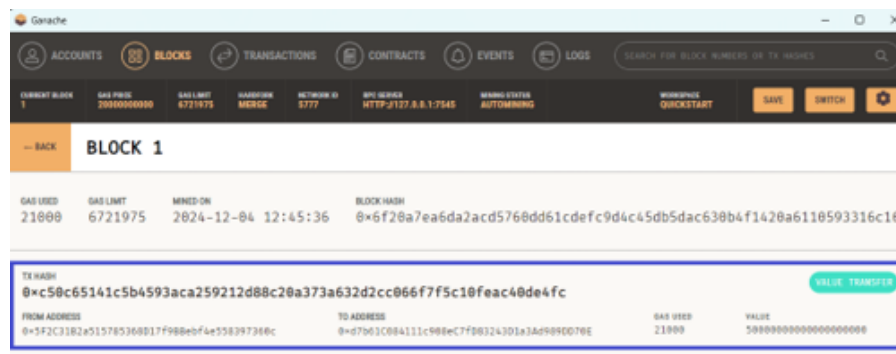


Figure 8. Ganache Transaction Log

Figure 8 displays the Ganache working environment. Ganache records transactions in a local blockchain environment used for testing and development. The transaction log shows the transaction hashes associated with transactions between the previously created wallet accounts, as shown in the results in Figure 9.

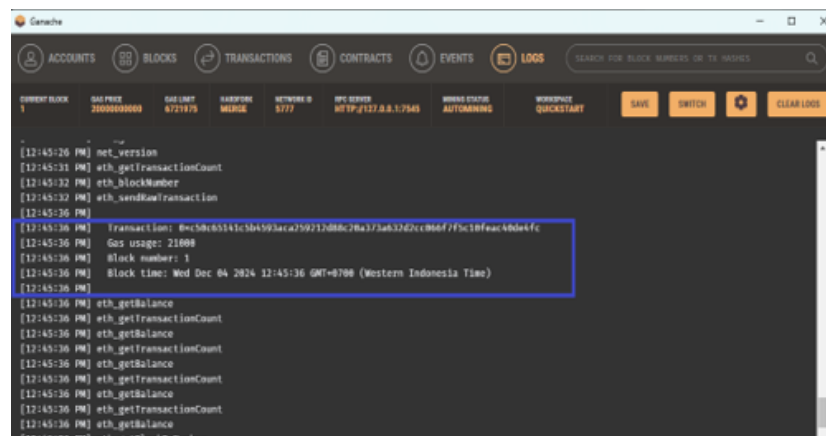


Figure 9. Ganache Log

The transaction log shows the transaction hash associated with transactions between the wallet accounts, demonstrating that the transactions were recorded in the test blockchain and can be verified using their corresponding transaction hashes.

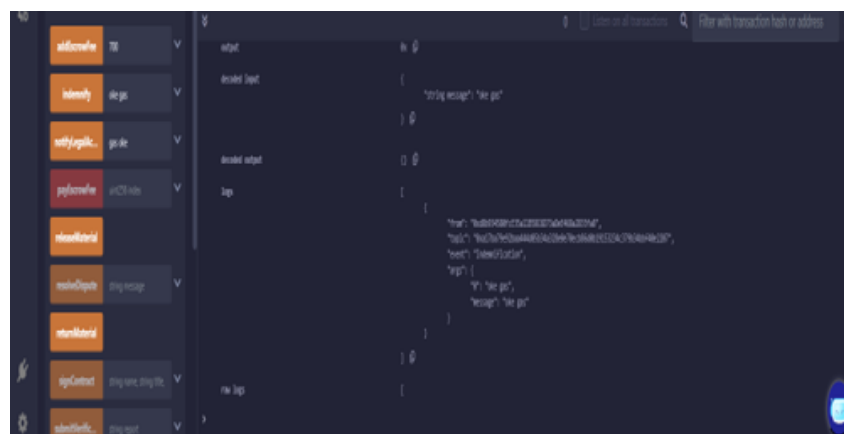


Figure 10. Log on Remix IDE

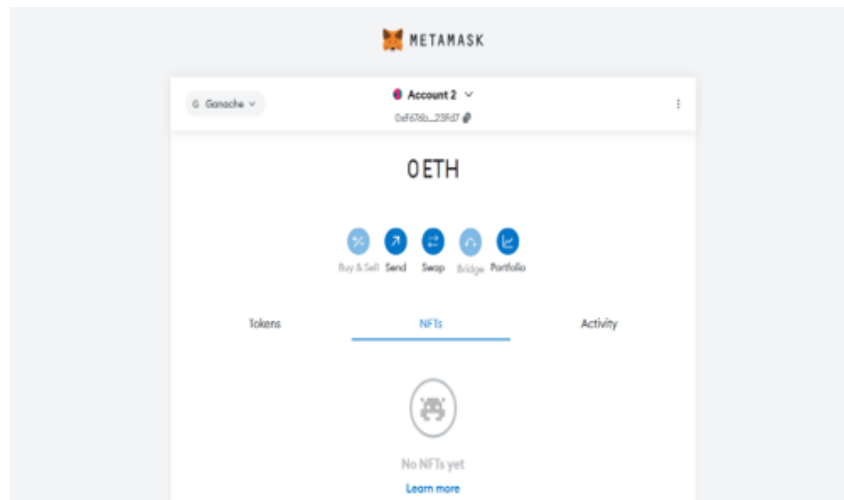


Figure 11. Seller Account

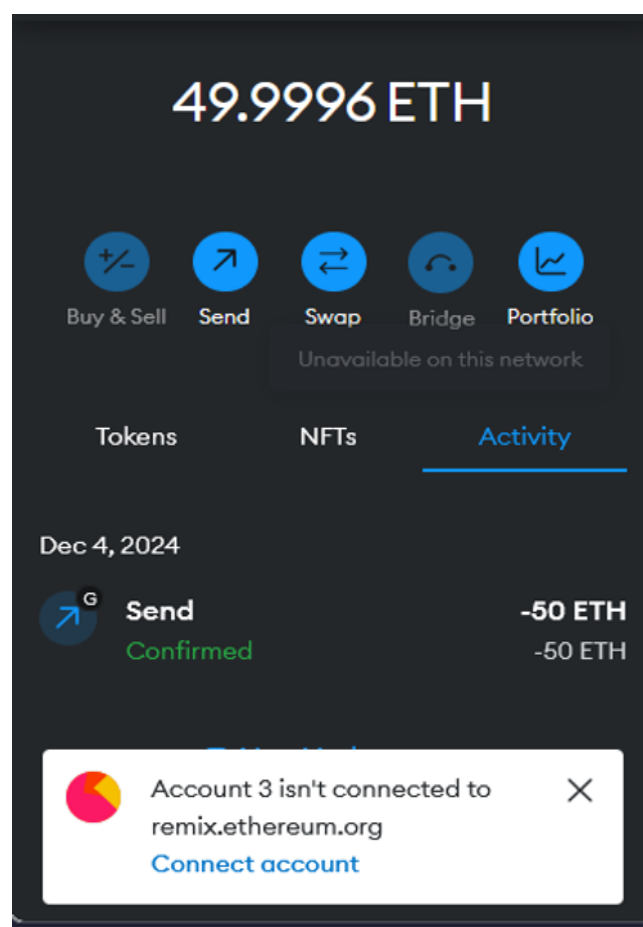


Figure 12. Escrow Transaction Result

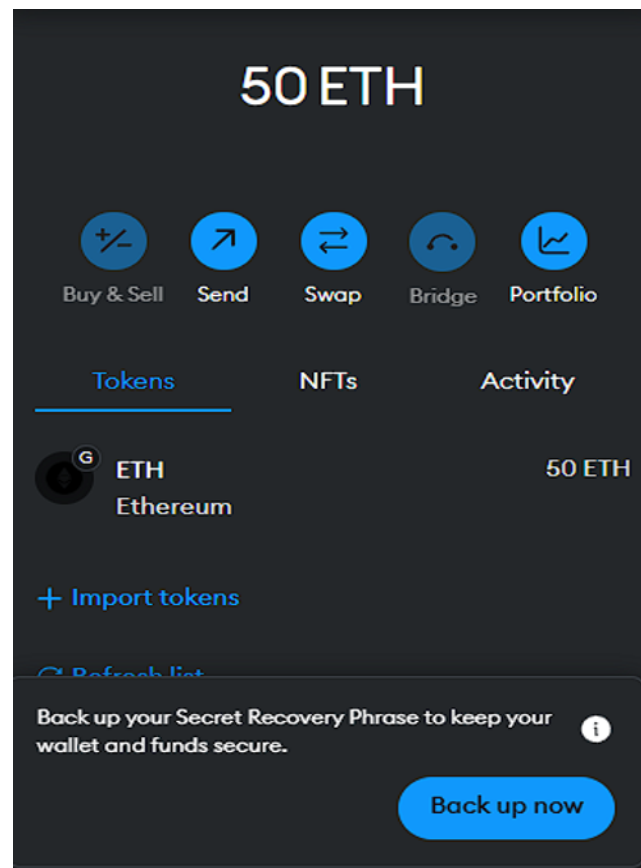


Figure 13. Seller Account Result

Discussion

Compared with conventional escrow, the prototype reduces the number of discretionary processing steps by encoding selected release conditions in a smart contract and providing a shared, time-ordered record of executed transactions. The principal transparency gain is therefore traceability: authorized participants can verify whether a transaction occurred and associate it with a specific transaction hash and contract state. The principal security contribution is narrower than the inherent security implied by the term “blockchain”; rather, it lies in the combination of role-restricted contract calls and tamper-evident transaction records within the tested prototype.

These benefits are accompanied by material limitations. Smart contracts can contain logic errors and security vulnerabilities, and an immutable ledger can preserve erroneous or maliciously triggered transactions as effectively as valid ones. The current study did not conduct a formal code audit, adversarial vulnerability testing, throughput benchmarking, or legal validation of automated escrow conditions. Consequently, the artifact should be regarded as a functional proof of concept rather than a production-ready substitute for regulated escrow services.

For real property transactions, deployment readiness also depends on off-chain identity verification, authoritative property-title records, dispute-resolution mechanisms, consumer protection, and the legal recognition of digital signatures and automated contractual performance. Future systems should therefore combine blockchain auditability with institutional governance, formal security assurance, and mechanisms for handling exceptions that cannot be resolved by code alone.

CONCLUSION

The developed prototype demonstrates that a property escrow workflow can be encoded in a smart contract and executed in an Ethereum-based test environment with traceable transaction records. The evidence supports functional feasibility, role-based execution, and transparent recording of contract activity; it does not demonstrate absolute security, reduced

transaction time, or legal effectiveness in real transactions because these outcomes were not directly measured.

The study contributes an end-to-end escrow architecture linking actor roles, conditional execution, and auditable blockchain records. Its main limitations are the local test environment, the absence of formal vulnerability auditing and performance benchmarking, and the lack of empirical legal or user validation. Future research should include smart-contract security audits, adversarial testing, cost and latency measurements, comparison with conventional escrow using predefined indicators, and regulatory validation before field deployment.

ACKNOWLEDGEMENT

The authors would like to express their sincere gratitude to Universitas Uyelindo for providing academic support, research facilities, and an environment that enabled the completion of this study. The authors also extend their appreciation to colleagues, researchers, and all individuals who contributed through valuable discussions, technical insights, and constructive feedback during the development and evaluation of the blockchain-based smart contract escrow prototype. Their support and encouragement were instrumental in completing this research.

AUTHOR CONTRIBUTION STATEMENT

Conceptualization, Omega Joel Patria Moata, Febi Elvira Messe, and Fiktor Imanuel Tanesab; methodology, Omega Joel Patria Moata and Fiktor Imanuel Tanesab; software development and smart contract implementation, Omega Joel Patria Moata; validation, testing, and analysis, all authors; investigation and visualization, Febi Elvira Messe and Fiktor Imanuel Tanesab; writing—original draft preparation, Omega Joel Patria Moata; writing—review and editing, all authors. All authors have read and agreed to the published version of the manuscript.

REFERENCES

- Abdilah, A., Sari, P. A., & Sasongko, A. T. (2023). Sistem Otomasi Lazy Minting NFT di Marketplace Rarible pada Blockchain Ethereum Menggunakan Autoit. *Jurnal Teknologi Dan Sistem Informasi Bisnis*, 5(3), 362–370. <https://doi.org/10.47233/jteksis.v5i3.915>
- Babel, M., Willburger, L., Lautenschlager, J., Völter, F., Guggenberger, T., Körner, M. F., Sedlmeir, J., Strüker, J., & Urbach, N. (2025). Self-sovereign identity and digital wallets. *Electronic Markets*, 35(1). <https://doi.org/10.1007/s12525-025-00772-0>
- Bassan, F., & Rabitti, M. (2024). From smart legal contracts to contracts on blockchain: An empirical investigation. *Computer Law & Security Review*, 55, 106035. <https://doi.org/10.1016/j.clsr.2024.106035>
- Chen, H., Pendleton, M., Njilla, L., & Xu, S. (2020). A survey on ethereum systems security: Vulnerabilities, attacks, and defenses. *ACM Computing Surveys*, 53(3), 1–43. <https://doi.org/10.1145/3391195>
- Chen, W., Guo, X., Chen, Z., Zheng, Z., & Lu, Y. (2020). Phishing scam detection on Ethereum: Towards financial security for blockchain ecosystem. *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI)*, 4456–4462. <https://www.ijcai.org/proceedings/2020/0621.pdf>
- Chen, Y., & Zhang, H. (2019). Outlier detection in student performance data. *Expert Systems with Applications*, 126, 217–229.
- Dimas, M. (2023). *Trend Penerapan Blockchain dalam Meningkatkan Keamanan informasi Digital*. Universitas Komputer Indonesia. https://www.researchgate.net/profile/Muhamad-Dimas/publication/370073889_Trend_Penerapan_Blockchain_dalam_Meningkatkan_Keamanan_informasi_Digital/links/643e73bf39aa471a52487e6c/Trend-Penerapan-Blockchain-dalam-Meningkatkan-Keamanan-informasi-Digital.pdf
- Gadekallu, T. R. (2022). Blockchain for the metaverse: A review. *ArXiv Preprint ArXiv:2203.09738*. <https://doi.org/10.48550/arXiv.2203.09738>
- Harlian, T. (2022). *Implementasi Blockchain Untuk Pendataan Dokumen Digital*. <https://repositori.telkomuniversity.ac.id/home/catalog/id/178454/slug/implementasi-blockchain-untuk-pendataan-dokumen-digital.html>
- Kushwaha, S. S., Joshi, S., Singh, D., Kaur, M., & Lee, H.-N. (2022a). Ethereum Smart Contract

- Analysis Tools: A Systematic Review. *IEEE Access*, 10, 57037–57062. <https://doi.org/10.1109/ACCESS.2022.3169902>
- Kushwaha, S. S., Joshi, S., Singh, D., Kaur, M., & Lee, H. N. (2022b). Systematic Review of Security Vulnerabilities in Ethereum Blockchain Smart Contract. In *IEEE Access* (Vol. 10). <https://doi.org/10.1109/ACCESS.2021.3140091>
- Naik, N., & Jenkins, P. (2020). Self-Sovereign Identity Specifications: Govern Your Identity Through Your Digital Wallet using Blockchain Technology. *2020 8th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*, 90–95. <https://doi.org/10.1109/MobileCloud48802.2020.00021>
- Onwubiko, A., Singh, R., Awan, S., Pervez, Z., & Ramzan, N. (2023). *Enabling Trust and Security in Digital Twin Management: A Blockchain-Based Approach with Ethereum and IPFS*. <https://doi.org/10.3390/s23146641>
- Preethi, Ulla, M. M., Sapna, R., Devadas, R. M., Hiremani, V., & Pavithra, N. (2026). Automated Escrow Services in Real Estate: Enhancing Trust and Efficiency with Smart Contracts. In N. K. Atulya, J. Dharm Singh, M. Durgesh Kumar, & A. Joshi (Eds.), *Intelligent Sustainable Systems* (pp. 226–235). Springer Nature Switzerland.
- Purba, C. (2024). *Implementasi Teknologi Blockchain pada Real Estate Transaction*. Sekolah Teknik Elektro dan Informatika. [https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi-dan-Koding/2023-2024/Makalah/Makalajh-II4031-Kriptokoding-2024\(23\).pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi-dan-Koding/2023-2024/Makalah/Makalajh-II4031-Kriptokoding-2024(23).pdf)
- Rosa-Bilbao, J., Boubeta-Puig, J., Lagares-Galán, J., & Vella, M. (2025). Leveraging complex event processing for monitoring and automatically detecting anomalies in Ethereum-based blockchain networks. *Computer Standards & Interfaces*, 91, 103882. <https://doi.org/10.1016/j.csi.2024.103882>
- Santhanalakshmi, M., Lakshana, M., & GM, M. S. (2023). Enhanced AES-256 cipher round algorithm for IoT applications. *Scientific Temper*, 14(01), 184–190. <https://doi.org/10.58414/SCIENTIFICTEMPER.2023.14.1.22>
- Saputra, I. (2024). Enhancing Environmental Data Management through Dynamic Smart Contract Interaction on Blockchain. *2024 International Conference on Information Technology Research and Innovation (ICITRI)*, 341–346. <https://doi.org/10.1109/ICITRI62858.2024.10698936>
- Toorajipour, R., Oghazi, P., Sohrabpour, V., Patel, P. C., & Mostaghel, R. (2022). Block by block: A blockchain-based peer-to-peer business transaction for international trade. *Technological Forecasting and Social Change*, 180, 121714. <https://doi.org/10.1016/j.techfore.2022.121714>
- Zhang, L., Ci, L., Wu, Y., & Wiwatanapataphee, B. (2024). The real estate time-stamping and registration system based on Ethereum blockchain. *Blockchain: Research and Applications*, 5(1), 100175. <https://doi.org/10.1016/j.bcra.2023.100175>