Tokenization of a Property on the Ethereum and Solana Blockchains: A Comparative Study

NIKOS E. MASTORAKIS
Technical University of Sofia,
English Language Faculty of Engineering
8 Kliment Ohridski Blvd, Building 12, Sofia 1000
BULGARIA

Abstract: This paper presents the tokenization of a 100 m² property into 100,000 tokens, which are then deployed on the Ethereum and Solana blockchains. Each token represents 0.001 m² of the property (or 0.00001 of its total value). The following is a step-by-step overview, along with sample code, demonstrating how to perform this tokenization on Ethereum using the ERC-20 token standard—the most commonly used standard for Ethereum-based tokens. Additionally, we show that tokenizing the property on the Solana blockchain is entirely feasible, although the process differs from Ethereum. Solana uses the SPL Token standard, and its programs are written in Rust, unlike Ethereum's Solidity-based development.

Key-Words: - Real Estate Tokenization, Blockchain Technology, Ethereum Blockchain, Solana Blockchain, ERC-20 Token, Standard SPL, Token Standard, Fractional Ownership, Digital Assets, Smart Contracts, Decentralized Finance (DeFi), Property Tokenization, Rust Programming Language, Solidity Programming, Language Token Economics

Received: April 15, 2024. Revised: January 29, 2025. Accepted: March 14, 2025. Published: May 23, 2025.

1 Introduction

The tokenization of real-world assets is emerging as a transformative application of blockchain technology, with the potential to revolutionize the way assets are owned, transferred, and managed. Among these assets, real estate stands out due to its high value, illiquidity, and often cumbersome transfer processes. By converting ownership rights of physical properties into digital tokens that are securely recorded on a blockchain, tokenization introduces unprecedented levels of accessibility, transparency, and efficiency to property transactions. Blockchain-based tokenization allows for the fractional ownership of real estate, lowering the traditional barriers to entry for individual investors. Instead of requiring substantial capital to purchase an entire property, investors can own small fractions through tokens representing proportional shares of the asset. This not only democratizes access to real estate investments but also enhances liquidity in an otherwise illiquid market by enabling tokens to be traded on secondary markets. Among the various blockchain platforms available, Ethereum and Solana are two of the most prominent and widely adopted for asset tokenization. Ethereum, the pioneering platform for decentralized applications, offers a well-established ecosystem and the widely-used ERC-20 token standard, which provides a consistent and interoperable framework for digital assets. However, Ethereum's network can be constrained by scalability issues and high transaction fees, particularly during periods of high demand. In contrast, Solana offers an alternative that emphasizes high throughput and low transaction costs, making it an attractive option for applications that require fast and economical operations. Solana employs the SPL (Solana Program Library) token standard and leverages a unique hybrid consensus mechanism combining Proof of History (PoH) and Proof of Stake (PoS), which enables significantly higher transaction speeds compared to Ethereum. However, Solana's ecosystem is newer, and its programming

model, based on the Rust language, poses different development challenges. This paper investigates the practical implementation of real estate tokenization on both the Ethereum and Solana blockchains. We focus on a case study where a 100 m² property is subdivided into 100,000 digital tokens, each representing 0.001 m² or 0.00001 of the property's value. Through this case study, we demonstrate the creation and deployment of tokens on both platforms, compare the technical and economic implications of each approach, and highlight the key differences in development workflows, token standards, and user experience. By analyzing and contrasting the two platforms, this work aims to provide a comprehensive understanding of the opportunities and challenges associated with real estate tokenization using blockchain technology. The results contribute to the growing body of research in decentralized finance (DeFi) and property technology (PropTech), and offer practical insights for developers, investors, and policymakers interested in harnessing blockchain for asset management and investment democratization.

2 Tokenizing Real Estate on the Ethereum Blockchain: A Case Study for a 100 m² Property

In this session, we present a comprehensive methodology for tokenizing a physical asset — specifically, a 100 square meter property — on the Ethereum blockchain. The objective is to fractionalize the property into 100,000 tokens, with each token representing 0.001 m² or 0.001% of the asset's value. We provide technical explanations, Solidity smart contract code, deployment steps, and security considerations, enabling real estate tokenization to be adopted securely, transparently, and legally. Blockchain technology enables the transformation of traditionally illiquid assets into fractional, easily tradeable digital units. In this context, real estate tokenization offers a revolutionary method for increasing liquidity, transparency, and accessibility in property ownership and investment. This paper focuses on Ethereum, the most established smart contract platform, to illustrate how a 100 m² property can be divided into 100,000 tokens.

Tokenizing real estate involves converting ownership or financial rights into digital tokens. Our key considerations include:

a) Regulatory Compliance, i.e. Local laws often classify fractional property tokens as securities.

b) Ownership Documentation. That means that the smart contract must be backed by legal documents recognizing token holders' rights.

c) Custody and Management What we mean is that if rental income is generated, a smart contract can redistribute earnings.

Now we will proceed with the Token Design. We opt for the **ERC-20 standard**, the most commonly supported token interface in Ethereum-compatible wallets and exchanges.

As token, we use the Mastor-Coin
The token symbol is MASTOR
Mastor Decimals: 18 (standard)
Total Supply: 100,000

We present now the Smart Contract Code (Solidity)

```
// SPDX-License-Identifier: MIT
```

```
pragma solidity ^0.8.0;

import "https://github.com/OpenZeppelin/openzeppelin-
contracts/blob/master/contracts/token/ERC20/ERC20.sol";

contract MyPropertyToken is ERC20 {
    constructor() ERC20("MyPropertyToken", "MASTOR") {
        _mint(msg.sender, 100000 * 10 ** decimals());
    }
}
```

where we inherit from OpenZeppelin's ERC20 contract. The constructor mints 100,000 tokens to the creator's wallet.

We need now the following: a MetaMask Wallet, a Remix IDE and a ETH (on Goerli testnet or Ethereum mainnet. We have now this procedure. We must open Remix IDE and paste the contract into a `.sol` file. We will compile using the Solidity compiler (v0.8.x). We will connect MetaMask (Injected Web3). We will deploy the contract. MetaMask will prompt us to approve the transaction. Upon deployment, we will receive 100,000 MASTOR tokens.

So, we will present now the Web Frontend (HTML + JS) Below is a minimal frontend for interaction using Ethers.js:

```
<!DOCTYPE html>
<html>
<head>
  <title>MyPropertyToken DApp</title>
  <script
src="https://cdn.jsdelivr.net/npm/ethers@5.7.2/dist/ethers.umd.min.js"></script>
</head>
<body>
  <h2>MyPropertyToken DApp</h2>
  <button onclick="connectWallet()">Connect MetaMask</button>
```

```
<p id="wallet"></p>
<h3>Balance:</h3>
<p id="balance">--</p>

<h3>Transfer Tokens</h3>
<input type="text" id="to" placeholder="Recipient Address" />
<input type="number" id="amount" placeholder="Amount" />
<button onclick="transfer()">Send</button>
<p id="status"></p>

<script>
 const tokenAddress = "YOUR_DEPLOYED_CONTRACT_ADDRESS";
 const abi = [
  "function balanceOf(address) view returns (uint)",
  "function transfer(address to, uint amount) returns (bool)",
  "function decimals() view returns (uint8)"
 ];

 let provider, signer, contract;

 async function connectWallet() {
  if (!window.ethereum) return alert("Install MetaMask!");
  provider = new ethers.providers.Web3Provider(window.ethereum);
  await provider.send("eth_requestAccounts", []);
  signer = provider.getSigner();
  contract = new ethers.Contract(tokenAddress, abi, signer);

  const address = await signer.getAddress();
  document.getElementById("wallet").textContent = address;
  const decimals = await contract.decimals();
  const balance = await contract.balanceOf(address);
```

```
    document.getElementById("balance").textContent =
ethers.utils.formatUnits(balance, decimals);
  }

  async function transfer() {
    const to = document.getElementById("to").value;
    const amount = document.getElementById("amount").value;
    const decimals = await contract.decimals();
    const tx = await contract.transfer(to, ethers.utils.parseUnits(amount, decimals));
    document.getElementById("status").textContent = "Transaction sent...";
    await tx.wait();
    document.getElementById("status").textContent = "Transfer confirmed!";
  }
 </script>
</body>
</html>
```

Before deploying any smart contract to the Ethereum mainnet, conducting a thorough audit is essential to ensure the contract's integrity and to prevent vulnerabilities that could lead to financial or legal risks. Smart contract audits—whether performed internally or by reputable third-party firms—can uncover common issues such as reentrancy attacks, unchecked input validations, or access control flaws. In addition, implementing strict ownership restrictions is crucial. By using an access control mechanism such as OpenZeppelin's Ownable contract, developers can limit critical administrative functions—such as minting new tokens, pausing the contract, or changing parameters—to the contract owner or a set of trusted addresses. This helps protect against unauthorized actions that could compromise the system. Moreover, if the tokenized asset is offered to the public or represents a financial interest, compliance with relevant regulations becomes mandatory. This includes integrating Know Your Customer (KYC) and Anti-Money Laundering (AML) procedures into the platform's user onboarding or transaction processes to verify the identities of participants and to meet the legal requirements of various jurisdictions. Together, these security measures ensure that the tokenization process is not only technically sound but also legally compliant and resilient to attack.

At the moment, we have established a foundational framework for tokenizing real estate on the Ethereum blockchain. However several promising avenues remain for future development and enhancement of such systems. One significant direction is the integration of smart contracts to manage rental income distribution. By linking rental revenue—whether from residential, commercial, or short-term leases—to the tokenized ownership structure, it is possible to automate the periodic disbursement of income to token holders in the form of ETH or stablecoins such as USDC or DAI. This would transform the token from a static representation of ownership into a dynamic income-generating asset, increasing its

utility and attractiveness to investors. Another key advancement lies in the implementation of on-chain governance mechanisms. Empowering token holders to participate in decisions regarding property management, major renovations, or sales through decentralized voting systems would create a more transparent and democratic model of real estate ownership. This could be accomplished by integrating governance protocols, such as quadratic voting or token-weighted voting, into the smart contract architecture. Such frameworks allow for collective decision-making while ensuring that those with greater stakes have proportionally greater influence. Furthermore, cross-chain interoperability offers substantial opportunities for expanding the reach and flexibility of tokenized property systems. By enabling the bridge of property tokens from Ethereum to other compatible blockchains such as Polygon, Avalanche, or Binance Smart Chain (BSC), users could benefit from lower transaction fees, faster confirmation times, and access to different DeFi ecosystems. Cross-chain bridges or wrapping protocols would allow tokens to move fluidly between chains while preserving their value and functionality. These enhancements, taken together, would not only increase the technical robustness of tokenized real estate platforms but also significantly broaden their appeal and accessibility in the global market.

3 Tokenizing Real Estate on the Solana Blockchain: A Case Study for a 100 m² Property. Comparison of the Tokenization on the Ethereum and Solana Blockchain.

In this session, we explore a practical and technical approach to tokenizing a 100-square-meter property into 100,000 tokens on the Solana blockchain. We present all steps—from concept to implementation—using SPL tokens and relevant Solana tooling. Additionally, we compare this methodology with Ethereum's ERC-20 standard, highlighting performance, cost, and ecosystem differences. Blockchain-based tokenization allows dividing tangible assets into digital tokens, increasing accessibility and liquidity. This process can democratize investment, reduce transaction costs, and enable programmable ownership features. Solana, a high-throughput Proof-of-History blockchain, offers a unique alternative to Ethereum due to its performance and cost-efficiency.

We aim again to tokenize a 100 m² property into 100,000 tokens on Solana such that: 1 token = 0.001 m² Tokens are transferable and stored in Solana wallets Tokens can later be used in smart contract logic for rental income, voting rights, or profit sharing

See Table 1 below

Feature	Solana	Ethereum (ERC-20)
Language	Rust, C	Solidity
Token Standard	SPL Token	ERC-20
TPS	~65,000	~30
Transaction Fees	<$0.01	$0.5–$50 (variable)
Finality	~400 ms	12–60 seconds
Wallet Support	Phantom, Sollet	MetaMask
Smart Contracts	More complex, lower level	Easier to write (Solidity)

Table 1

We present now the Tokenization on Solana: Step-by-Step. First of all we must install Solana CLI

```
sh -c "$(curl -sSfL
https://release.solana.com/stable/inst
all)"
```

Then we Configure CLI

```
solana config set --url
https://api.devnet.solana.com
solana-keygen new
```

Now we install SPL-Token CLI

```
cargo install spl-token-cli
```

After that we must Create Token (No Decimals) as follows

```
spl-token create-token --decimals 0
```

This outputs our new token's public address.

We have now to create an Associated Token Account

```
spl-token create-account
<TOKEN_ADDRESS>
```

and finally we Mint 100,000 Tokens

```
spl-token mint <TOKEN_ADDRESS> 100000
```

Tokens now reside in our wallet and represent fractional ownership of the property.

We have to present now thw Web-Based Interaction with SPL Token. First we must connect Phantom Wallet and Show Balance (JavaScript)

```
<script type="module">
import {
  Connection,
  PublicKey,
  clusterApiUrl
} from "https://cdn.jsdelivr.net/npm/@solana/web3.js/+esm";

import {
  getAssociatedTokenAddress,
  getAccount,
} from "https://cdn.jsdelivr.net/npm/@solana/spl-token/+esm";

const tokenMint = new PublicKey("YOUR_TOKEN_MINT_ADDRESS");

async function checkBalance() {
  const provider = window.solana;
  await provider.connect();
  const wallet = provider.publicKey;
  const connection = new Connection(clusterApiUrl("devnet"));

  const tokenAddress = await getAssociatedTokenAddress(tokenMint, wallet);
  const tokenAccount = await getAccount(connection, tokenAddress);

  alert("You own: " + tokenAccount.amount.toString() + " tokens.");
}

document.querySelector("button").addEventListener("click", checkBalance);
</script>
<button>Check Token Balance</button>
```

We need to Replace YOUR_TOKEN_MINT_ADDRESS with our actual mint address.

We present now some legal and regulatory aspects: Tokenizing real estate introduces several legal and regulatory considerations that must be addressed to ensure compliance with relevant laws. Foremost among these are securities regulations. If the tokenized property is marketed and sold as an investment vehicle, the tokens may be classified as securities under the laws of many jurisdictions. This could subject them to registration requirements and oversight by financial regulatory bodies. Another crucial aspect is property law. The digital tokens must be linked to real, legally recognized ownership of the property. This typically requires interfacing with traditional land registries and may involve legal contracts that recognize token holders as beneficial owners or stakeholders in the real asset. Custody and compliance are also essential, especially if the platform facilitates secondary trading or involves large numbers of participants. This necessitates robust KYC (Know Your Customer) and AML (Anti-Money Laundering) procedures to ensure that all token holders are verified and transactions are traceable and legitimate. Due to the complex legal landscape and potential jurisdictional variances, it is imperative to consult qualified legal advisors before launching a real estate tokenization project.

Beyond simple token creation and transfer, smart contracts can be developed to add advanced features to the tokenized property system. Using Solana's Anchor framework, which simplifies the development of smart contracts in Rust, developers can create robust applications that mirror the flexibility of Ethereum-based dApps. One possible extension is rental income distribution, where token holders automatically receive a proportional share of rental income based on their holdings. Another valuable feature is governance mechanisms, allowing token holders to vote on key property decisions—such as leasing terms, property management, or sale proposals. To meet compliance requirements, smart contracts can also include KYC enforcement and token freezing capabilities. This ensures that only verified users can hold or transfer tokens and that malicious actors can be restricted as needed. Anchor streamlines the process of writing and deploying such contracts by providing macros, type-safe instructions, and easy-to-use testing tools, making Rust development more accessible and efficient.

4. Comparison of the tokenization with Solana and Ethereum Which is better?

We compare now the tokenization with Solana and Ethereum Which is better?
In Table 2, we present a comparison between the tokenization on the Solana Blockcahin and and tokenization on the Ethereum Blockcahin.
In Table 3 and Table 4, we present the Strengths of each Blockcahin

Comparison: Solana vs Ethereum for Real Estate Tokenization		
Criterion	**Solana**	**Ethereum**
Speed (TPS)	~65,000 transactions per second	~30 TPS (Ethereum mainnet)
Transaction Fees	<$0.01 per transaction	$0.5–$50 (depends on network congestion)
Finality Time	~400 ms	12–60 seconds
Smart Contract Language	Rust (complex but fast)	Solidity (easier, more mature tools)
Token Standard	SPL Token	ERC-20/ERC-721/ERC-1155
Ecosystem Maturity	Growing, strong DePIN/NFT/infra projects	Highly mature, massive DeFi/NFT ecosystem
Wallet Support	Phantom, Sollet, Backpack	MetaMask, Coinbase Wallet, Trust Wallet
Developer Support	Lower (steeper learning curve in Rust)	Very high (Solidity is widely known)
Legal Tooling	Emerging legal infrastructure	More legal precedents & tokenized property pilots
Stability	Newer chain, more frequent outages historically	

Table 2

Solana Strengths:

- **Speed & Efficiency**: Processes thousands of transactions per second at negligible cost.
- **Low Fees**: Ideal for micropayments or large token distributions.
- **High Scalability**: Better for systems with many fractional owners.

Ethereum Strengths:

- **Smart Contract Flexibility**: Better tooling for complex contracts (e.g., automated rent splits, DAO governance).
- **Regulatory Maturity**: More projects have tested legal models for property tokenization.
- **Ecosystem Integration**: Easy access to DeFi protocols, stablecoins, identity layers (e.g., Kleros, ENS).

Table 3

Which One Is Better?

It depends on our goal:

Our Priority	Best Blockchain
Low-cost minting, transfers, high-speed access	✓**Solana**
Access to DeFi tools, mature contracts, NFTs	✓**Ethereum**
Legal compliance, previous pilot projects	✓**Ethereum**
Modern wallet UX, scaling to millions of users	✓**Solana**
Simpler smart contract development and tooling	✓**Ethereum**

Table 4

After this study we recommend this:

We will use Solana if we want scalable, low-cost real estate tokenization with fast execution, especially for a large number of fractional owners or transactions. We will use Ethereum if our priority is DeFi integration, compliance tooling, or smart governance logic backed by a rich developer ecosystem and stable legal groundwork.

5. Conclusion

In this paper, we presented the tokenization of a 100 m² property into 100,000 tokens, which are then deployed on both the Ethereum and Solana blockchains. Each token represents 0.001 m² of the property (equivalent to 0.00001 of its total value). A step-by-step overview was provided, accompanied by sample code, demonstrating how to perform this tokenization on Ethereum using the ERC-20 token standard—the most widely adopted standard for Ethereum-based tokens. Additionally, we demonstrated that tokenizing the property on the Solana blockchain is entirely feasible, though the process differs significantly from Ethereum's approach. Solana utilizes the SPL Token standard, and its programs are written in Rust, contrasting with Ethereum's Solidity-based development environment.

Tokenizing real estate assets on blockchain platforms presents a transformative opportunity to unlock liquidity, fractional ownership, and enhanced transparency. Both Solana and Ethereum offer viable paths for such tokenization, but they cater to different needs and priorities. Solana excels with its high throughput, minimal transaction fees, and rapid finality, making it highly suitable for projects requiring large-scale fractional ownership and cost efficiency. In contrast, Ethereum provides a more mature ecosystem with robust developer tools, extensive DeFi integrations, and a stronger legal and regulatory foundation, which benefits projects focused on complex smart contract functionality and broader market acceptance. Ultimately, the choice between Solana and Ethereum should align with the project's specific technical requirements, compliance considerations, and user experience goals. As blockchain technology continues to evolve, hybrid approaches leveraging the strengths of both platforms may also emerge as optimal solutions for real estate tokenization.

References:

[1] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," [Online]. Available: https://bitcoin.org/bitcoin.pdf

[2] M. Crosby, P. Pattanayak, S. Verma, and V. Kalyanaraman, "Blockchain technology: Beyond bitcoin," *Applied Innovation Review*, vol. 2, pp. 6–10, Jun. 2016.

[3] K. Christidis and M. Devetsikiotis, "Blockchains and Smart Contracts for the Internet of Things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016.

[4] J. K. Liu, D. S. Wong, E. Y. Zhang, and X. Deng, "Enhancing privacy and security in outsourced biometric identification," IEEE Transactions on Dependable and Secure Computing, vol. 12, no. 5, pp. 428–439, Sept.–Oct. 2015.

[5] A. Giaretta, M. Dion, and M. Goodwin, "Preserving Digital Heritage: Technological Challenges and the Role of Blockchain," in Proc. 2020 IEEE Intl. Conf. on Big Data (Big Data), Atlanta, GA, USA, Dec. 2020, pp. 5790–5792.

[6] P. Zhang, D. C. Schmidt, J. L. White, and G. Lenz, "Blockchain Technology Use Cases in Health Care," in Proc. 2018 IEEE Intl. Conf. on Blockchain (Blockchain), Halifax, Canada, Jul. 2018, pp. 177–183.

[7] L. Chen et al., "Decentralized Identity: Towards a Trustworthy Ecosystem for the Digital Society," IEEE Communications

Standards Magazine, vol. 5, no. 4, pp. 40–46, Dec. 2021.

[8] Solana Foundation, "Solana Documentation." [Online]. Available: https://docs.solana.com. [Accessed: May 2025].

[9] SPL Token Program, "Solana Program Library — SPL Token." [Online]. Available: https://spl.solana.com/token. [Accessed: May 2025].

[10] Anchor Labs, "Anchor Framework Documentation." [Online]. Available: https://book.anchor-lang.com. [Accessed: May 2025].

[11] Ethereum Foundation, "ERC-20 Token Standard." [Online]. Available: https://eips.ethereum.org/EIPS/eip-20. [Accessed: May 2025].

[12] V. Buterin and G. Wood, "Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform," 2014. [Online]. Available: https://ethereum.org/en/whitepaper/. [Accessed: May 2025].

[13] M. G. Catalano and A. Meiklejohn, "Tokenization of Real-World Assets on Blockchain: Challenges and Opportunities," Journal of Blockchain Research, vol. 3, no. 1, pp. 15–30, 2020.

[14] S. Choi and Y. Kim, "Legal Implications of Blockchain-Based Property Tokenization," International Journal of Law and Technology, vol. 12, no. 2, pp. 87–103, 2021.

[15] G. W. Peters and E. Panayi, "Understanding Modern Banking Ledgers through Blockchain Technologies: Future of Transaction Processing and Smart Contracts on the Internet of Money," Bank of England Working Papers, 2016.

[16] G. Wood, "Polkadot: Vision for a Heterogeneous Multi-Chain Framework," 2016. [Online]. Available: https://polkadot.network/Polkadot-Whitepaper.pdf. [Accessed: May 2025].

[17] Y.-Y. Hsieh, J.-P. Vergne, and S. Wang, "The internal and external governance of blockchain-based organizations: Evidence from cryptocurrencies," Blockchain Research Journal, vol. 4, no. 3, pp. 101–115, 2018.