GimaBlockchain Whitepaper: Rug-Proof Vault Standard (RPV-20) & GimaTrust Badges

GimaBlockchain

September 16, 2025

Owned and Operated by GimaBlockchain – Securing DeFi for Every Investor

Abstract

GimaBlockchain, owned and operated as the pioneer of secure decentralized finance, introduces the **Rug-Proof Vault Standard (RPV-20)**, a revolutionary token and vault protocol that eliminates DeFi's trust deficit. By enforcing permanent liquidity locks, automated yield strategies, and market-aligned incentives, RPV-20 ensures investor safety. Paired with **GimaTrust Badges**, oracle-integrated, soulbound NFTs that provide real-time yield and compliance data, RPV-20 creates a transparent, institution-ready DeFi ecosystem. This whitepaper details the vision, technical architecture, tokenomics, and roadmap for GimaBlockchain's mission to secure DeFi for retail and institutional investors alike.

Contents

1 Executive Summary

GimaBlockchain introduces the **Rug-Proof Vault Standard (RPV-20)**, a protocol-level token standard designed to eradicate rug pulls and restore investor confidence in DeFi. Unlike ERC-20, which allows liquidity withdrawal by malicious actors, RPV-20 enforces:

- **Permanent Liquidity Locks**: Liquidity provider (LP) tokens are irrevocably committed to audited vaults.
- **Automated Yield Strategies**: Yearn-style vaults optimize returns across DeFi protocols (e.g., Aave, Curve).
- **Market-Aligned Incentives**: Creators profit only through token appreciation, not extractive fees.
- **GimaTrust Badges**: On-chain, oracle-fed NFTs verify compliance, yields, and trust metrics in real-time.

Key Benefits:

- **Investors**: Guaranteed liquidity safety, transparent APY, and verifiable compliance.
- **Creators**: Instant credibility via GimaBlockchain's seal of approval.
- **Institutions**: Auditable, risk-managed exposure for $5M+ deployments.
- **Ecosystem**: A trust standard as transformative as ERC-20 was for fungibility.

RPV-20 and GimaTrust position GimaBlockchain as the backbone of a secure, scalable DeFi future, owned and operated to empower every investor.

2 Background & Motivation

DeFi has unlocked trillions in liquidity but remains plagued by trust issues: rug pulls, exit scams, hidden admin keys, and misaligned incentives. GimaBlockchain's RPV-20 builds on DeFi's historical milestones to deliver the next wave—trust embedded at the protocol level.

2.1 Historical Context

- **2009 – Bitcoin**: Decentralized money, free from intermediaries.
- **2015 – Ethereum**: Programmable smart contracts, enabling DeFi apps.
- **2020 – DeFi Summer**: Yield farming and automated pools (Uniswap, Compound) scaled TVL to billions.
- **2021 – NFTs & Meme Coins**: Viral adoption showed community power.
- **2022–2024 – Layer 2 & RWAs**: Scaling solutions and real-world asset tokenization.
- **2025+ – RPV-20**: GimaBlockchain's trust standard for secure DeFi.

2.2 Problem Insight

- **Rug Pulls**: Over 80% of new DEX tokens risk liquidity withdrawal, costing investors billions.
- **Opaque Controls**: Hidden admin privileges or minting functions erode trust.
- **Misaligned Incentives**: Creators profit without market exposure, encouraging scams.
- **Fragmented Trust**: Lockers, audits, and KYC are optional and often bypassed.

Motivation: GimaBlockchain aims to make DeFi investable by embedding trust into the token standard, ensuring safety, transparency, and scalability for all investors.

3 Problem Statement

DeFi adoption is hindered by:

- **Liquidity Vulnerability**: ERC-20 tokens allow creators to drain liquidity pools at will.
- **Mistrust**: Retail fears scams; institutions avoid unverified exposure.
- **Fragmented Tools**: Existing solutions (lockers, audits) are not protocol-native.
- **Incentive Misalignment**: Creators gain without risk, undermining credibility.

Solution: RPV-20, owned and operated by GimaBlockchain, embeds permanent liquidity locks, automated yields, and verifiable trust, securing DeFi for investors worldwide.

4 Solution Overview – RPV-20

RPV-20 is a fungible token and vault hybrid, combining ERC-20's composability with Yearn-style yield automation and Pump.fun's launch simplicity, but with an ironclad rug-proof guarantee.

4.1 Key Features

- **Permanent Liquidity Lock**: LP tokens are locked forever in GimaBlockchain's audited vaults.
- **Vault-Based Yields**: Auto-compounding strategies maximize returns.
- **Market-Aligned Incentives**: Creators deposit tokens into vaults, earning via market appreciation.
- **Composability**: Integrates with DEXs, wallets, and analytics platforms.
- **Public Verification**: On-chain compliance and yield data via GimaTrust badges.

4.2 Deployment Flow

1. Deploy token via GimaBlockchain's `TokenFactory`.
2. Add liquidity on a DEX (e.g., Uniswap, Curve).
3. Transfer LP tokens to `RugProofVault`.
4. Governance initializes vault and deploys yield strategy.
5. Creators deposit allocated tokens into vault.
6. Public launch with GimaTrust badge, signaling trust.

5 RPV-20 Specification

RPV-20 extends ERC-20 with vault and badge mechanics, ensuring investor security.

5.1 Interface Methods

- `vaultTotalLiquidity()`: Returns total LP tokens locked.
- `vaultShareOf(address)`: User's share of vault.
- `claimYield(address, uint256)`: Claims user's earned yield.
- `lockLiquidity(token, amount)`: Irreversibly locks LP tokens.
- `vaultStrategy()`: Returns current yield strategy.
- `isRPVCompliant(address)`: Verifies token compliance.

5.2 Events

- `LiquidityLocked(token, amount, timestamp)`
- `YieldClaimed(user, amount)`
- `StrategyChanged(oldStrategy, newStrategy)`
- `VaultInitialized(vault, strategy)`

5.3 Rules

- LP tokens are irreversible once deposited.
- Yield strategies must be audited and pluggable.
- Yield distribution is proportional to vault shares.
- Governance changes require 7-day timelock and multi-sig.

6 Technical Architecture

GimaBlockchain's RPV-20 is a modular, secure protocol stack, owned and operated for maximum trust.

6.1 Components

- **TokenFactory**: Automates token and vault deployment.
- **RugProofVault**: Locks LP tokens, interfaces with strategies.
- **IStrategy**: Pluggable modules for yield (e.g., Yearn, Aave).
- **Governance**: Timelock + multi-sig for strategy updates.
- **GimaTrust Badges**: Oracle-integrated NFTs for compliance and yields.

6.2 Vault Mechanics

- **Deposit**: LP tokens locked permanently; users/creators mint shares.
- **Yield Harvest**: Auto-compounds via strategy; proportional rewards.
- **Claim**: Users withdraw yields (no principal access).
- **No Withdrawals**: LP tokens are immutable—no exit functions.

6.3 Security Principles

- No LP withdrawal functions (rug-proof guarantee).
- `ReentrancyGuard` for transaction safety.
- High-precision accounting (`ACC_PRECISION = 1e36`).
- Timelock and multi-sig for governance upgrades.

7 GimaTrust Badge System

GimaTrust Badges are soulbound ERC-721 NFTs, auto-minted for RPV-20 tokens, signaling compliance and streaming real-time yield data via oracles.

7.1 Badge Categories

- **Liquidity Badge**: Verifies permanent LP lock.
- **Audit Badge**: Confirms strategy audits.
- **Governance Badge**: Signals transparent DAO control.
- **Sustainability Badge**: Validates treasury yield allocation.
- **Adoption Badge**: Tracks community engagement.

7.2 Gima Seal Badge (Flagship)

- **Implementation**: Soulbound ERC-721 NFT, minted by `TokenFactory`.
- **Oracle Integration**: Chainlink/Pyth feeds for APY, lock health.
- **UI Display**: Green shield in wallets/DEXs: “GimaBlockchain RPV-20 Certified.”
- **Verification**: Public `isRPVCompliant(address)` query.

7.3 Oracle Mechanics

- **Push-Pull Hybrid**: Push vault events; pull APY feeds.
- **Chainlink Jobs**: Fetch yields from Yearn/Aave (e.g., 7.2% APY).
- **Deviation Checks**: Pause updates if feeds diverge >2%.
- **Cross-Chain**: LayerZero/Axelar for multi-chain sync.

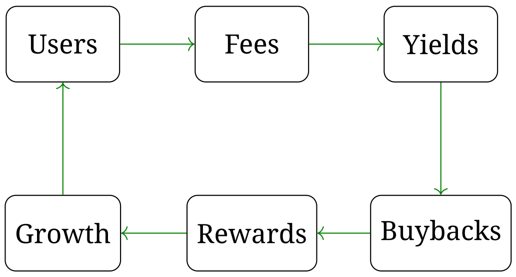
8 Tokenomics Framework

RPV-20 aligns creators, investors, and institutions via transparent, vault-driven economics.

8.1 Example $5M Liquidity Launch

- **70% Permanent Liquidity Vault**: $3.5M locked forever.
- **20% Treasury Vault**: $1M for buybacks, DAO bounties.
- **5% Community Incentives**: $250K for staking, 2-year vesting.
- **5% Team**: $250K, 3–4 year vesting, vault-earning.

8.2 Value Flow



8.3 Guardian Pool

5% of treasury tokens earn yields, funding:

- Community bounties (e.g., dev grants).
- Oracle subsidies (0.01% query fees).
- Governance operations.

9 Ecosystem Roles

- **Investors**: Trade, farm, hold with confidence.
- **Creators**: Launch RPV-20 tokens with trust.
- **Governance**: Issues badges, oversees compliance.
- **Auditors**: Verify strategies, optimize yields.
- **Institutions**: Deploy large positions safely.

10 Security & Attack Resistance

- **Rug Resistance**: No LP withdrawal functions.
- **Immutable Contracts**: Audited, no backdoors.
- **Badge Incentives**: Revoked if non-compliant.
- **Oracle Hardening**: Multi-oracle, deviation checks.
- **Insurance Pools**: Optional for yield risks.

11 Comparison of Token Standards

11.1 Table 1: Token Standards (Part 1)

11.2 Table 2: Token Standards (Part 2)

11.3 Visual Comparison: Liquidity Security

11.4 Key Insights

- **Liquidity Security**: Only RPV-20, owned by GimaBlockchain, guarantees permanent liquidity locks, eliminating rug pulls.

Table 1: Token Standard Comparison (Part 1)

Feature / Standard	**ERC-20**	**ERC-721**	**ERC-1155**	**BEP-20**	**Solana SPL**
Liquidity Security	Optional	N/A	Optional	Optional	Optional
Yield Strategies	External	N/A	External	External	External
Creator Alignment	Often misaligned	One-off NFT	Often misaligned	Often misaligned	Often misaligned
Reputation Layer	None	None	None	None	None
Governance	External	Optional	Optional	External	External
Institutional Safety	Low	Low	Medium	Medium	Medium
Real-Time Yield	No	No	No	No	No
Risk of Rug Pull	High	N/A	Medium	High	High
Composability	High	Medium	High	High	Medium
Ideal Use Cases	DeFi	Gaming	Hybrid	DeFi	DeFi

Table 2: Token Standard Comparison (Part 2)

Feature / Standard	**BRC-20**	**Polygon ERC-20**	**Avalanche C-Chain**	**RPV-20**
Liquidity Security	Optional	Optional	Optional	Permanent LP lock
Yield Strategies	External	External	External	Native vaults
Creator Alignment	Often misaligned	Often misaligned	Often misaligned	Market-aligned
Reputation Layer	None	None	None	GimaTrust badges
Governance	Optional	External	External	Transparent, on-chain
Institutional Safety	Low	Medium	Medium	High
Real-Time Yield	No	No	No	Yes
Risk of Rug Pull	High	High	High	Zero
Composability	Low	High	High	High
Ideal Use Cases	Experiments	DeFi	DeFi	Safe DeFi & Institutional Projects

- **Yield Integration**: RPV-20's native vaults with Chainlink/Pyth feeds automate and verify yields, unlike external solutions.
- **Creator Alignment**: Creators earn via market appreciation, enforced by GimaBlockchain's vault mechanics.
- **Reputation Layer**: GimaTrust badges provide on-chain, oracle-verified trust signals, unique to RPV-20.
- **Institutional Safety**: Auditable vaults and real-time data make RPV-20 ideal for $5M+ deployments.

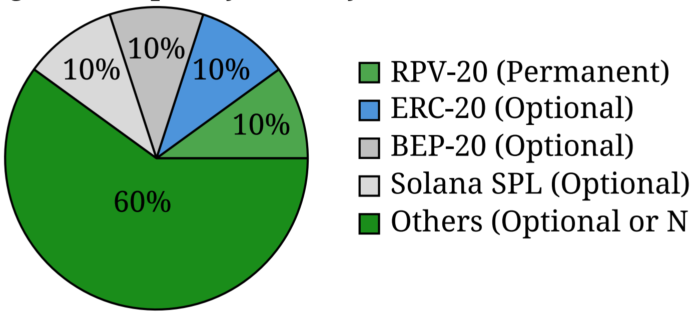
12 Roadmap

- **Q4 2025**: Testnet launch, audits, first RPV-20 tokens.
- **Q1 2026**: GimaTrust badge issuance, community adoption.
- **Q2 2026**: DAO formation, decentralized governance.
- **Q3 2026**: Institutional partnerships, multi-chain vaults.
- **Q4 2026+**: Oracle dashboards, AI analytics, RWA integration.

13 Future Extensions

- **Advanced Governance**: Community-managed strategies, DAO voting.
- **Stablecoin Vaults**: RPV-USD for predictable yields.
- **Cross-Chain**: Unified liquidity via LayerZero/Axelar.

Figure 1: Liquidity Security Across Token Standards



- **AI Analytics**: Predictive yield models.
- **RWA Tokenization**: Secure real-world asset yields.

14 Conclusion

GimaBlockchain's RPV-20 and GimaTrust Badges redefine DeFi:

- **Permanent Locks**: Eliminate rug pulls.
- **Automated Yields**: Sustain growth via audited strategies.
- **Fair Incentives**: Align creators with market success.
- **Oracle Badges**: Real-time, verifiable trust signals.

Summary Statement: *Bitcoin gave us money. Ethereum gave us apps. Uniswap gave us liquidity. GimaBlockchain's RPV-20 gives us trust.*

A RugProofVault.sol

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

import "@openzeppelin/contracts/token/ERC20/IERC20.sol";
import "@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol";
import "@openzeppelin/contracts/security/ReentrancyGuard.sol";

interface IStrategy {
    function deposit(uint256) external;
    function realizeYield() external returns (uint256);
    function withdrawAllToVault() external;
    function estimatedTotalAssets() external view returns (uint256);
}

contract RugProofVault is ReentrancyGuard {
    using SafeERC20 for IERC20;
    IERC20 public immutable lpToken;
    IERC20 public immutable rewardToken;
    IStrategy public strategy;
    address public immutable governance;
    uint256 public totalShares;
    uint256 public accRewardPerShare = 0;
    uint256 public constant ACC_PRECISION = 1e36;
    mapping(address => uint256) public userShares;
    mapping(address => uint256) public userRewardDebt;
    bool public initialized;

    event Deposited(address indexed user, uint256 amount, uint256 shares);
    event Harvested(uint256 amount);
    event Claimed(address indexed user, uint256 reward);
    event StrategyUpdated(address oldStrategy, address newStrategy);

    modifier onlyGov() { require(msg.sender == governance, ”not governance”); _; }

    constructor(address _lpToken, address _rewardToken, address _strategy, address
        lpToken = IERC20(_lpToken);
        rewardToken = IERC20(_rewardToken);
        strategy = IStrategy(_strategy);
        governance = _governance;
    }

    function initializeAndDeployLPToStrategy() external onlyGov {
        require(!initialized, ”already initialized”);
        initialized = true;
        uint256 lpBal = lpToken.balanceOf(address(this));
        require(lpBal > 0, ”no LP”);
        lpToken.safeApprove(address(strategy), lpBal);
        strategy.deposit(lpBal);
    }

    function deposit(uint256 amount) public nonReentrant {
        require(amount > 0, ”zero deposit”);
        rewardToken.safeTransferFrom(msg.sender, address(this), amount);
        uint256 sharesToMint = amount;
        totalShares += sharesToMint;
        userShares[msg.sender] += sharesToMint;
        userRewardDebt[msg.sender] = (userShares[msg.sender] * accRewardPerShare)
        emit Deposited(msg.sender, amount, sharesToMint);
    }

    function depositCreatorTokens(uint256 amount) external { deposit(amount); }

    function harvest() external nonReentrant returns (uint256) {
        uint256 yield = strategy.realizeYield();
        require(yield > 0, ”no yield”);
        if (totalShares > 0) {
            accRewardPerShare += (yield * ACC_PRECISION) / totalShares;
        }
        emit Harvested(yield);
        return yield;
    }

    function pendingClaim(address user) public view returns (uint256) {
        uint256 owed = (userShares[user] * accRewardPerShare) / ACC_PRECISION;
        return owed > userRewardDebt[user] ? owed - userRewardDebt[user] : 0;
    }

    function claim() external nonReentrant {
        uint256 claimable = pendingClaim(msg.sender);
        require(claimable > 0, ”no rewards”);
        userRewardDebt[msg.sender] = (userShares[msg.sender] * accRewardPerShare)
        rewardToken.safeTransfer(msg.sender, claimable);
        emit Claimed(msg.sender, claimable);
    }

    function setStrategy(address newStrategy) external onlyGov {
        require(newStrategy != address(0), ”zero address”);
        address old = address(strategy);
        strategy = IStrategy(newStrategy);
        emit StrategyUpdated(old, newStrategy);
    }

    function estimatedStrategyAssets() external view returns (uint256) {
        return strategy.estimatedTotalAssets();
    }
}
```

B GimaSealBadge.sol

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

import ”@openzeppelin/contracts/token/ERC721/ERC721.sol”;
import ”@openzeppelin/contracts/access/Ownable.sol”;
import ”@chainlink/contracts/src/v0.8/interfaces/AggregatorV3Interface.sol”;
import ”./RugProofVault.sol”;

contract GimaSealBadge is ERC721, Ownable {
    address public immutable tokenFactory;
    address public yieldOracle;
    uint256 private badgeCounter;
    struct BadgeData {
        address token;
        address vault;
        bool isCompliant;
        uint256 lastYieldUpdate;
        uint256 currentAPY;
    }
    mapping(uint256 => BadgeData) public badges;
    mapping(address => uint256) public tokenToBadgeId;

    event BadgeMinted(address indexed token, uint256 badgeId, address vault);
    event YieldUpdated(address indexed token, uint256 apy, uint256 timestamp);
    event ComplianceRevoked(address indexed token, uint256 badgeId);

    modifier onlyFactory() {
        require(msg.sender == tokenFactory, ”not factory”);
        _;
    }

    constructor(address _tokenFactory, address _yieldOracle) ERC721(”GimaSealBadge
        tokenFactory = _tokenFactory;
        yieldOracle = _yieldOracle;
    }

    function mintBadge(address token, address vault) external onlyFactory returns
        require(tokenToBadgeId[token] == 0, ”badge exists”);
        badgeCounter++;
        uint256 badgeId = badgeCounter;
        badges[badgeId] = BadgeData(token, vault, true, block.timestamp, 0);
        tokenToBadgeId[token] = badgeId;
        _mint(tokenFactory, badgeId);
        _setApprovalForAll(tokenFactory, tokenFactory, false);
        emit BadgeMinted(token, badgeId, vault);
        return badgeId;
    }

    function requestYieldUpdate(address token) external {
        uint256 badgeId = tokenToBadgeId[token];
        require(badgeId != 0, ”no badge”);
        require(badges[badgeId].isCompliant, ”non-compliant”);
        (, int256 apy,,,) = AggregatorV3Interface(yieldOracle).latestRoundData();
        require(apy >= 0, ”invalid APY”);
        badges[badgeId].currentAPY = uint256(apy);
        badges[badgeId].lastYieldUpdate = block.timestamp;
        RugProofVault vault = RugProofVault(badges[badgeId].vault);
        require(vault.initialized() && vault.estimatedStrategyAssets() > 0, ”invali
        emit YieldUpdated(token, uint256(apy), block.timestamp);
    }

    function checkCompliance(address token) external returns (bool) {
        uint256 badgeId = tokenToBadgeId[token];
        if (badgeId == 0) return false;
        RugProofVault vault = RugProofVault(badges[badgeId].vault);
        bool compliant = vault.initialized() && vault.estimatedStrategyAssets() > 
        if (!compliant && badges[badgeId].isCompliant) {
            badges[badgeId].isCompliant = false;
            emit ComplianceRevoked(token, badgeId);
        }
        return compliant;
    }

    function getBadgeData(address token) external view returns (BadgeData memory) 
        uint256 badgeId = tokenToBadgeId[token];
        require(badgeId != 0, ”no badge”);
        return badges[badgeId];
    }

    function _beforeTokenTransfer(address from, address to, uint256 tokenId, uint25
        require(from == address(0), ”soulbound: non-transferable”);
        super._beforeTokenTransfer(from, to, tokenId, batchSize);
    }

    function updateYieldOracle(address newOracle) external onlyOwner {
        require(newOracle != address(0), ”zero address”);
        yieldOracle = newOracle;
    }
}
```

C Tokenomics Design Guide

- **70% Liquidity Vault**: Permanent lock for investor safety.
- **20% Treasury**: Yield-funded buybacks, DAO operations.
- **5% Community**: Staking/farming rewards, 2-year vesting.
- **5% Team**: 3–4 year vesting, vault-earning.
- **Guardian Pool**: 5% of treasury for bounties, oracle fees.

D Audit & Testing Checklist

- Verify no LP withdrawal paths in `RugProofVault`.
- Test strategy reward distribution accuracy.
- Confirm `ACC_PRECISION` prevents rounding errors.
- Audit timelock (7-day) and multi-sig governance.
- **End-to-End UI/Dashboard Verification**:
  - **Badge Display**: Test `getBadgeData(token)` □ Verify green shield in MetaMask/Uniswap: “GimaBlockchain RPV-20 Certified.”
  - **Yield Heatmap**: Simulate Chainlink update □ Render APY graph (e.g., “7.2% Curve”).
  - **Lock Tracker**: Query `estimatedStrategyAssets()` □ Display “$4.2M Locked.”
  - **Alerts**: Test oracle deviation (>2% □ pause) □ Push “Yield Anomaly.”
  - **Cross-Chain**: Sync badge via LayerZero □ Unified yield display.
  - **Full Flow**: Launch token □ Mint badge □ Lock LP □ Update yield □ Claim rewards.