Verifiable Randomness for Decentralised Raffles

Exploring the secure and transparent future of on-chain lottery systems using Chainlink VRF.

The Problem: Trust in Digital Randomness

Traditional digital raffles often lack verifiable randomness, leading to concerns about fairness and transparency. Participants must trust a centralised entity to select winners impartially.

- Lack of transparency
- Centralisation risks
- Susceptibility to manipulation

Our Solution: Chainlink VRF-Powered Raffle System

We propose a secure, transparent, and provably fair raffle system built on blockchain technology, leveraging Chainlink's Verifiable Random Function (VRF).

1

Decentralised

Operates without central control.

NFT

Transparent

All transactions and winner selections are recorded on-chain.

3

Provably Fair

Randomness is cryptographically guaranteed by Chainlink VRF.

How it Works: A Single-Raffle Process

1

Deposit Period

Users deposit a fixed amount of a specified ERC-20 token (e.g., USDT or a custom token) into the raffle smart contract.

2

Period Closes

Once the deposit window ends, no more entries are accepted, and the system prepares for winner selection.

3

Randomness Request

The smart contract requests a cryptographically secure random number from Chainlink VRF.

4

Winner Selection

Upon receiving the verifiable random number, the smart contract deterministically selects the winner(s) from the pool of participants.

5

Prize Distribution

The winning amount is automatically transferred to the winner's wallet or made available for them to claim.

Key Component: Chainlink Verifiable Random Function (VRF)

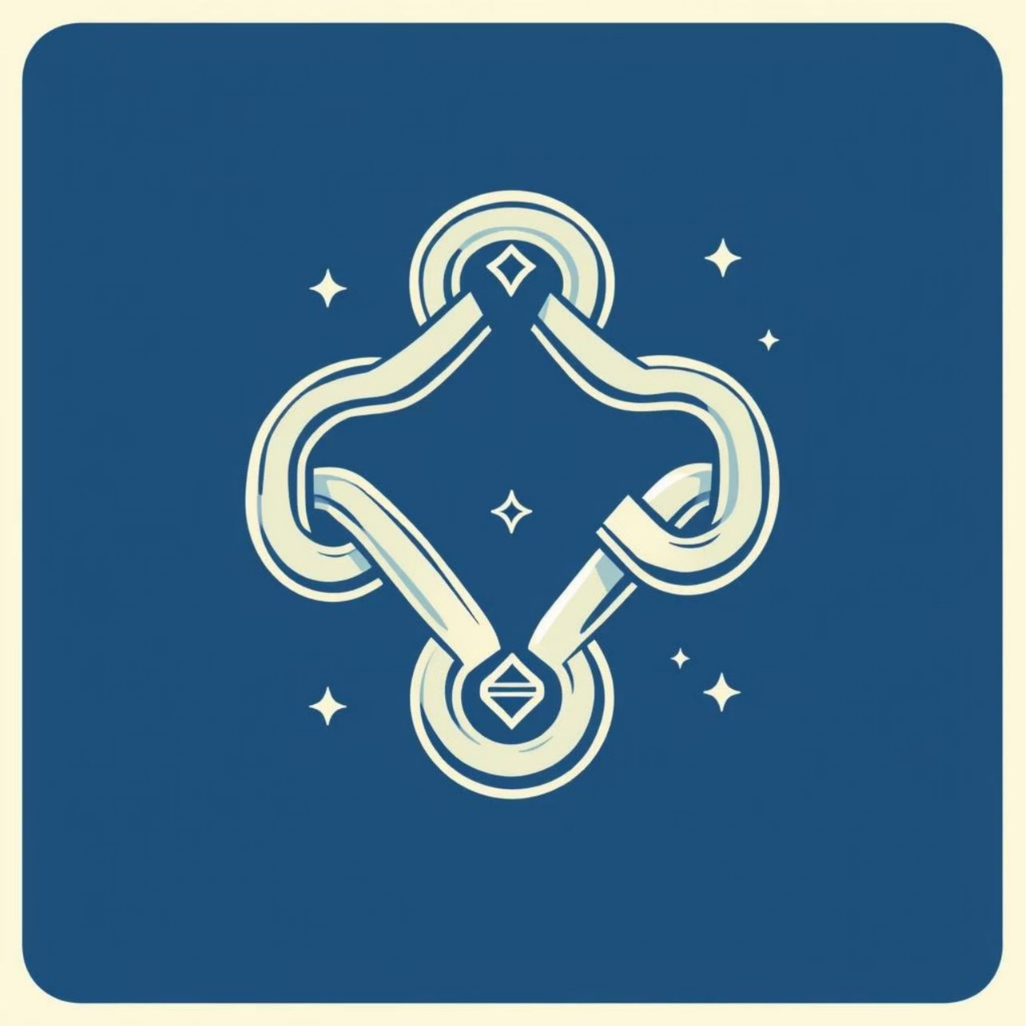
Chainlink VRF provides a highly secure and tamper-proof source of randomness essential for fair raffles.

On-chain verifiable: Cryptographic proofs confirm the randomness is genuine and untampered.

Tamper-resistant: Prevents manipulation by participants or contract owners.

Decentralised: Not reliant on a single point of failure.

EVM-compatible: Seamlessly integrates with Ethereum Virtual Machine-based blockchains.



Functional & Non-Functional Requirements

Functional

- Users can deposit ERC-20 tokens.
- Raffle has distinct deposit and selection phases.
- Chainlink VRF integration for random winner selection.
- Winner(s) can claim or automatically receive winnings.

Non-Functional

- High security against attacks and exploits.
- Transparency of all raffle operations.
- Efficiency in gas usage for transactions.
- User-friendly interface for participation.

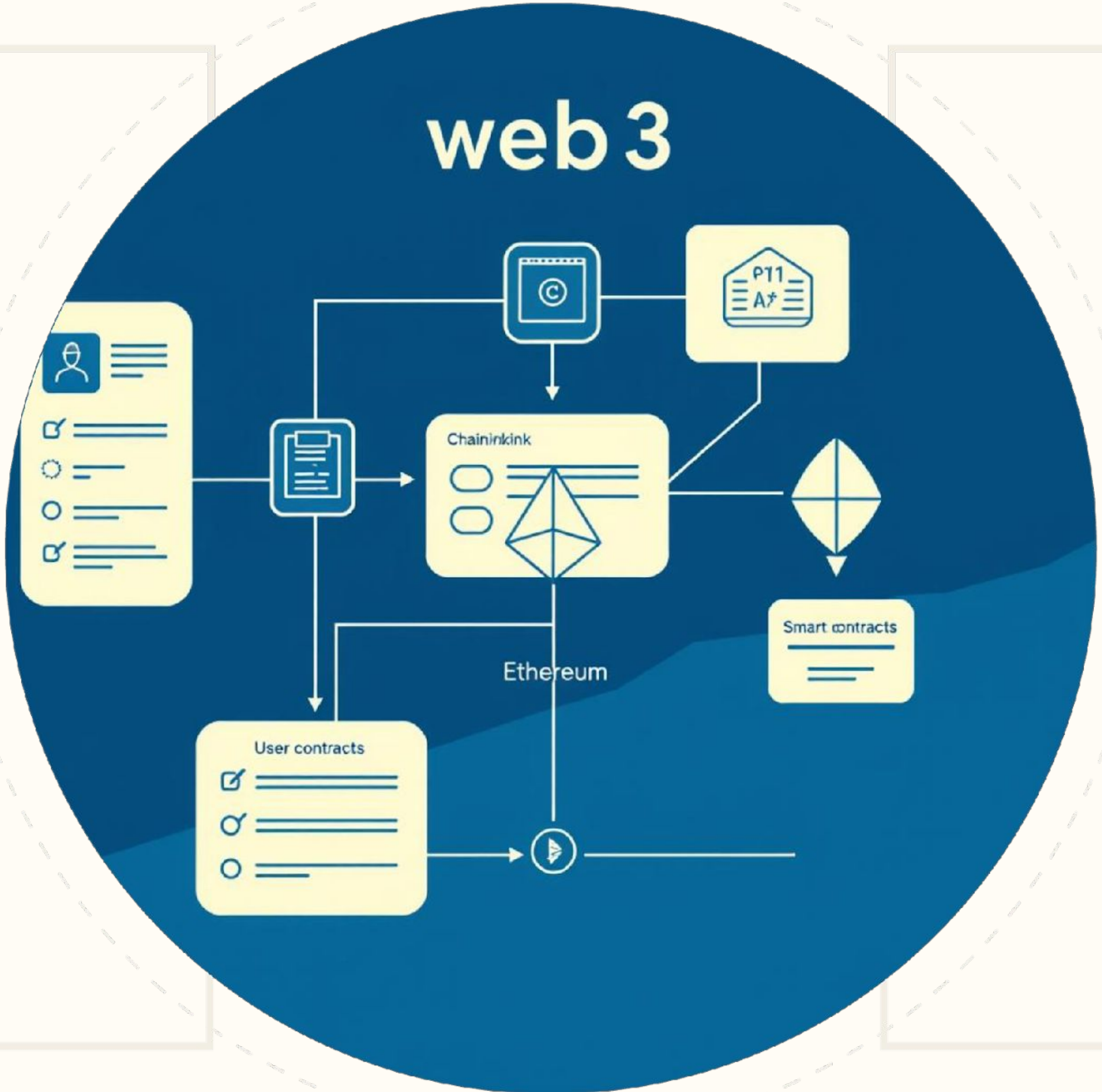
Architecture & Implementation Approach

Telegram Bot Interface

User deposits and claims via Tg APII

Smart Contract

Handles deposits and winner logic



Chainlink Node

Requests VRF and returns randomness

Ethereum

On-chain storage and transactions

Technology Stack & Implementation

Backend Architecture

- Language: Python 3.11+
- Framework: Flask for REST API
- Database: PostgreSQL 14+
- ORM: SQLAlchemy
- Web3: Web3.py library

Link: https://github.com/leaderpartiii/Raffel-System

Blockchain & Infrastructure

- Chain: Ethereum (EVM-compatible)
- Smart Contracts: Solidity
- Randomness: Chainlink VRF
- Deployment: Docker + Kubernetes
- Cache: Redis for events

Link: http://github.com/L3xu5/lottery-smart-contract

Test Cases: Ensuring Robustness

Rigorous testing is crucial to validate the fairness and functionality of the raffle system.

Valid Deposits

Verify users can successfully deposit tokens during the open period.

Invalid Deposits

Ensure deposits are rejected outside the open period or with incorrect token amounts.

Randomness Verification

Confirm Chainlink VRF returns a verifiable random number and the winner selection logic is correct.

Winner Payout

Test successful and accurate distribution of winning funds to the selected winner(s).

Edge Cases

Test scenarios like zero participants, single participant, and multiple winners.

Team Structure: 4 Specialized Roles

🔐 Smart Contract Developer

Solidity development

VRF integration

Blockchain security

⚙️ Backend Developer ⭐

Wallet management

Transaction processing

REST API & listeners

🤖 Bot Developer

Telegram interface

User management (FSM)

Push notifications

🚀 DevOps/Integration

Deployment & monitoring

Scaling & infrastructure

Database optimization

Project Timeline & Team Collaboration

November 18: Team formation and topic approval.

November 20: Unassigned students grouped, topics allocated.

December 6: Project Presentation (10 minutes including Q&A).

December 13: Report Submission (Problem Statement, Objectives, Requirements, Design, Implementation, Conclusion).

Fair participation and work distribution are key to success.

Conclusion: A Fairer Future for Raffles

By integrating Chainlink VRF, our decentralised raffle system offers unparalleled fairness, transparency, and trust.

Key Takeaways:

- Provably fair randomness
- Enhanced transparency
- Decentralised operation
- Secure and robust solution