Arush Singh

ZERO-KNOWLEDGE ENGINEER
SMART CONTRACT DEVELOPER

04arush 04arush
04arush 04arush.github.io
04arush@gmail.com New Delhi, India

PROFILE

Blockchain developer specialising in **Solidity Smart Contract** development and **Zero-Knowledge Cryptography**, with a focus on **DeFi Protocols**, on-chain automation, and rigorous testing. Built and deployed production contracts integrating **Chainlink VRF & Automation**, and authored ZK circuits in **Noir** — spanning Merkle Sum Tree proof-of-solvency and undercollateralised credit verification — with on-chain proof verification via **Barretenberg UltraHonk**. Grounded in smart contract security principles — reentrancy, access control, invariant & fuzz testing — and actively advancing toward a **ZK Engineer** specialisation.

SKILLS

SMART CONTRACTS

Solidity · Foundry · DeFi · Chainlink VRF / Automation / Oracle · OpenZeppelin

SECURITY & TESTING

Unit Testing · Fuzz Testing · Invariant Testing · Ethernaut · SC Security Basics

ZK / CRYPTOGRAPHY

Noir · Barretenberg (bb) · UltraHonk Verifier · Merkle Sum Trees · Poseidon Hashing · SNARKs / STARKs

TOOLING & LANGUAGES

Rust · TypeScript · Node.js · Linux · Git · Etherscan

PROJECTS

Zero-Knowledge Proof of Solvency

↗ github.com/04arush/ZK-Proof-of-Solvency

Noir · Merkle Sum Trees · Poseidon · Barretenberg · Solidity · Foundry · TypeScript

A zero-knowledge proof-of-solvency protocol letting an asset custodian prove liabilities coverage and honest per-user inclusion without disclosing individual account data. Built on a **Merkle Sum Tree** where every node commits to both a Poseidon hash and a running balance sum, cryptographically binding sums into the hash chain so an operator can't swap or forge sibling balances. The Noir circuit proves a leaf's inclusion, matches the root sum to the custodian's declared aggregate liability, and constrains every balance to `[0, 2^64)` via Noir's native range checks — closing the **negative-balance attack**, where an operator masks a reserve deficit by injecting a fake negative-balance account. Off-chain TypeScript builds the tree and derives witnesses, cross-checked against the circuit's Poseidon implementation via a dedicated test suite; bb produces an UltraHonk proof consumed on-chain by `ProofOfSolvency.sol:proveAndRecord()`, which validates the claimed root before calling the generated `UltraHonkVerifier`. Custom contract logic holds **100% line, function, and branch coverage**. Design inspired by Summa (PSE)'s open-source proof-of-reserves reference implementation.

ZK-Powered Undercollateralized Lending Prot.

↗ github.com/04arush/ZK-Powered-Uncollateralized-Lending-Protocol

Noir · Barretenberg · Solidity · Foundry · OpenZeppelin · Next.js

A DeFi lending protocol that enables undercollateralised borrowing (50% collateral ratio vs the industry-standard 150%) by verifying off-chain creditworthiness with a ZK proof — without revealing the user's credit score or income on-chain. The Noir circuit takes `credit_score` and `income` as private inputs and asserts both exceed protocol-defined public thresholds; `nargo prove` produces a proof passed directly to `LendingPool.borrow()`, where the on-chain `UltraVerifier` (generated by Barretenberg) validates it before any funds move. Core contract implements `deposit`, `borrow`, `repay`, and `liquidate` with strict Solidity conventions, NatSpec, `ReentrancyGuard`, and `SafeERC20`. Test suite achieves **100% line, statement, branch, and function coverage** across 23 passing tests: unit tests covering all happy and revert paths, fuzz tests (256 runs each) for random amounts and multi-user independence, and invariant tests running 256 sequences × 500 calls (128,000 actions per invariant) enforcing solvency, accounting integrity, and collateral floors via ghost-variable tracking. Deployed to Sepolia across 5 iterations, paired with a Next.js/wagmi dApp where proof generation runs entirely in-browser via Barretenberg WASM (10–30s) so private data never leaves the user's device.

Decentralized Provably Fair Raffle

↗ github.com/04arush/Raffle-Contest

Solidity · Chainlink VRF v2.5 · Chainlink Automation · Foundry

A fully autonomous on-chain raffle that requires zero admin intervention from entry through to prize payout. Randomness is sourced from **Chainlink VRF v2.5** — a subscription-based, cryptographically verifiable RNG — ensuring no party can predict or manipulate the winner. **Chainlink Automation** monitors `checkUpkeep` (time elapsed, raffle open, non-zero balance, participants present) and autonomously triggers `performUpkeep` when all conditions are met. A `HelperConfig` pattern provides chain-aware deployment: on Anvil it spins up a `VRFCoordinatorV2_5Mock` and a mock LINK token; on Sepolia it wires to live Chainlink contracts. The deploy script auto-handles subscription creation, LINK funding, and consumer registration in a single broadcast. 12 unit tests cover all state transitions, event emissions, access control, and upkeep logic; a fuzz test (256 runs) validates `fulfillRandomWords` can only be called after `performUpkeep`; a `skipFork` modifier gates VRF-mock tests to Anvil only.

HACKATHONS

Frostbyte Hackathon — Finale

↗ frostbyte-hackathon.devpost.com

Devpost · Online · Participation Award

Advanced to the finale of the Frostbyte Hackathon series after placing in the qualifier round. Built the **ZK-Powered Undercollateralized Lending Protocol** — a Noir + Barretenberg + Solidity stack enabling privacy-preserving credit verification on-chain — and received a Participation Award at the finale.

Frostbyte Hackathon — Qualifier

↗ frostbyte.devpost.com

Devpost · Online · Participation & Completion Recognition

Submitted the **Employee Payroll Manager** — a Chainlink Automation–powered on-chain payroll system — earning Participation & Completion Recognition and an invitation to the Frostbyte Finale.

EDUCATION

Indira Gandhi National Open University (IGNOU)

Aug 2023 - Ongoing

Bachelor's — Computer Application

CERTIFICATIONS

Noir Programming & ZK Circuits	Cyfrin Updraft · Apr 2026
Fundamentals of Zero-Knowledge Proofs	Cyfrin Updraft · Mar 2026
Advanced Foundry	Cyfrin Updraft · Mar 2026
Foundry Fundamentals	Cyfrin Updraft · Mar 2026
Solidity Smart Contract Development	Cyfrin Updraft · Jan 2026
Rust Programming Basics	Cyfrin Updraft · Feb 2026
Advanced Web3 Wallet Security	Cyfrin Updraft · Feb 2026
Web3 Wallet Security Basics	Cyfrin Updraft · Feb 2026
Blockchain Basics	Cyfrin Updraft · Dec 2025