The =nil; zkLLVM team recently participated in zkHack Istanbul, our first in-person hackathon. We brought two bounties to the table, and now we're glad to share the results. The winners came up with a zkGaming auto battler titled =nil; chronicle

and an encryption scheme for PVSS, a provable framework for multiparty verification protocols.

Before diving into the details, we want to say thank you to everyone who took part in completing our projects, and came to use and discuss our toolchain, and of course to the amazing zkHack organizing team.

zkHack Bounties

=nil; Foundation offered two bounties totaling $5,000 during the event:

Bounty 1: Template Rust+zkLLVM project

Requirements: Make a template Rust project using zkLLVM toolchain. This project should become a good starting point for other developers and teams building zk-circuits with zkLLVM.

[More details here

](https://www.notion.so/Template-Rust-zkLLVM-project-85e667a6ea4242aeafb23f0f7c25eb34?pvs=21).

Bounty 2: Solution example with zkLLVM

Requirements: Make an example solution for an isolated zero-knowledge problem using zkLLVM.More details here.

Winner 1: PVSS-buidl

A set of functions and circuits for the prover and verifier to build a Public Verifiable Secret Sharing scheme based on lattices.

Prize: $1500

Team: Grigorii Melnikov

Category: Template Rust+zkLLVM project

Project submission

Project documentation

This project opens new possibilities for secure validation by bringing PVSS (Public Verifiable Secret Sharing) scheme to a provable domain.

PVSS is a framework for building various multiparty protocols, such as DKG (decentralized key generation), RNG (random number generation), and more. The main advantage of PVSS over other VSS schemes is public verifiability. The correctness of the scheme can be verified by anyone using only public parameters, without revealing any secret information. In a blockchain setting, PVSS correctness can be verified by Smart Contracts.

How is this solution practically useful? PVSS schemes are used as part of the setup ceremony in decentralized applications to generate parameters for cryptographic schemes that require a trusted setup. A multi-party ceremony allows for a decrease in malicious influence during the parameter generation process. Clearly, involving more parties in the process reduces the need for trust in the parameters. However, even state-of-the-art PVSS schemes are not scalable enough from an on-chain verification perspective. Using zk-SNARKs to "compress" the verification procedure allows verifying a multi-party setup ceremony with thousands of participants.

Winner 2: =nil; chronicle

An auto battler game using zk to keep players' choices hidden from their opponents.

Prize: $3500

Team: ZkAutochess

Category: Solution example with zkLLVM

Project submission

Project documentation

=nil; chronicle is a video game where the initial player's choice of positions is hidden from the opponent until the automated fight begins. The ZkAutochess team built this game on zkLLVM with verification on Ethereum. As the idea of zkGaming

requires, the project compiles the game engine into a circuit with zkLLVM and uses the Placeholder EVM verifier to securely obtain the game result on Ethereum side.

Here's what the team says about why they chose this setup for zk/on-chain implementation:

- Auto-fight can be too expensive to execute on-chain. Battles should run off-chain with their results submitted to contracts, e.g., to issue rewards/payouts to the winners.
- Information on unit composition and placement should not be public and valid. A player should be able to learn about another team's setup only by playing against them.
- A player should be able to change the behavior of auto-controlled units. Advanced players can alter the default AI of a unit, and the game engine can prove that the battle was executed according to the correct algorithm.

Do we finally see a zkGaming example that is actually interesting to play? Our zkLLVM team believes so. In any case, =nil; chronicle is a step towards the future of decentralized gaming that is developing rapidly. We advise not only studying the docs behind this project, but also visiting the website - it's fun.

Keep on hacking, we're open source

We believe that the assets of zk development should be accessible and reused by anyone, which is why our toolchain is open source.

See the resources below to start building with =nil;'s zkLLVM:

- Read the docs at =nil; Developer Hub
- See an example project with C++ and Crypto3 library:https://github.com/nilfoundation/zkllvm-template
- See more circuit & input examples athttps://github.com/NilFoundation/zkLLVM/tree/master/examples

Have any questions left? Don't hesitate to follow us and discuss on any platform you prefer.

- Discord
- Telegram
- Twitter