zkStark: WuKong

Noy, RuoYan, Geoff

What is it?

- A strategy auto battle game
- Player distributes 30 point among strength, agility, intelligence to create a strategy
- Use one strategy to battle with another one to get higher rank
- Built with **Cairo, Dojo, Cocos, React, Typescript**

Demo

Why made this

1. AW → Fully On Chain Game → Provable Off Chain Game

2. Research On How cairo works on provable part

The progress of zkstark proof

Stone Prover: https://github.com/starkware-libs/stone-prover/

Integrity: https://github.com/HerodotusDev/integrity

The progress of zkstark proof

1. Write a Calculator In Cairo

```
fn main(input: Array<felt252>) -> Array<felt252> {
    let challengerSt = input.slice(0, 3);
    let defenerSt = input.slice(3, 6);

    let result = ResultTrait::decodeFromArray(@input.slice(6, 7));

    let mut simulator: Simulator = SimulatorTrait::initialize(
        @StrategyTrait::decodeFromArray(@challengerSt), @StrategyTrait::decodeFromArray(@defenerSt)
    );

    simulator.startBattle();

    // assert(result == simulator.getResult(), 'result not right');

    return array![1];
}       You, 7 days ago • feat: basic battle logic (#2) …
```

The progress of zkstark proof

2. Use stone-prover to generate proof

```
mkdir -p proof
cairo1-run ./target/dev/prover.sierra.json --args "[1 2 3 3 2 1 2]" --layout=recursive --air_public_input=proof/public_input.
json --air_private_input=proof/private_input.json --trace_file=proof/trace.bin --memory_file=proof/memory.bin --proof_mode
```



```
cpu_air_prover --out_file=proof/proof.json --private_input_file=proof/private_input.json --public_input_file=proof/
public_input.json --prover_config_file=config/cpu_air_prover_config.json --parameter_file=config/cpu_air_params.json
```

The progress of zkstark proof

3. Verify recursive proof on chain

```
impl CairoVerifierImpl<
    TContractState, +HasComponent<TContractState>
> of super::ICairoVerifier<ComponentState<TContractState>> {      Bartosz Nowak, 6 months ago • fa
    fn verify_proof(
        ref self: ComponentState<TContractState>,
        stark_proof: StarkProof,
        cairo_version: CairoVersion
    ) -> (felt252, felt252) {
        stark_proof.verify(SECURITY_BITS);
        let (program_hash, output_hash) = match cairo_version {
            CairoVersion::Cairo0 => stark_proof.public_input.verify_cairo0(),
            CairoVersion::Cairo1 => stark_proof.public_input.verify_cairo1(),
        };
        self.emit(ProofVerified { program_hash, output_hash });
        (program_hash, output_hash)
    }
}
```

Check Proof Arguments

- program hash = calculator hash
- output contains the function input


```
  "n_steps": 131072,
  "memory_segments": {
    "pedersen": {
      "begin_addr": 2201,
      "stop_ptr": 2201
    },
    "range_check": {
      "begin_addr": 5273,
      "stop_ptr": 5385
    },
    "output": {
      "begin_addr": 2190,
      "stop_ptr": 2201
    },
    "program": {
      "begin_addr": 1,
      "stop_ptr": 53
    },
    "execution": {
      "begin_addr": 1181,
      "stop_ptr": 2190
    },
    "bitwise": {
      "begin_addr": 21657,
      "stop_ptr": 21657
    }
  },
```

Findings

- Generate Proof Time: 118s in AMD EPYC 7763, 4 cores, Github codespace container

```
    Finished release target(s) in 4 seconds
Generate fri steps list successfully
Generate Proof Time: 118.273041223 seconds
root@2b9746d72ff3:/data/packages/prover# bash scripts/generateProof.sh
```

- Privacy mode is not available

gkaempfer commented on Oct 17, 2023

we have no plans to support this as it isn't required for the scaling use case and practically the current proofs are used as sufficiently hiding for cases where privacy is desirable (but doesn't have to be cryptographically provable). If you have sufficient understanding of the protocol and what needs to be done to achieve ZK on Stone, you could attempt to contribute. It is a delicate task though, and as I mentioned above, it could also affect performance somewhat due to the increase in trace size.

What it could be

What it could be

1. More Strategy Depth
2. Token staked on strategy
3. An experiment on generate L2

Thanks for Listening