Lesson 9

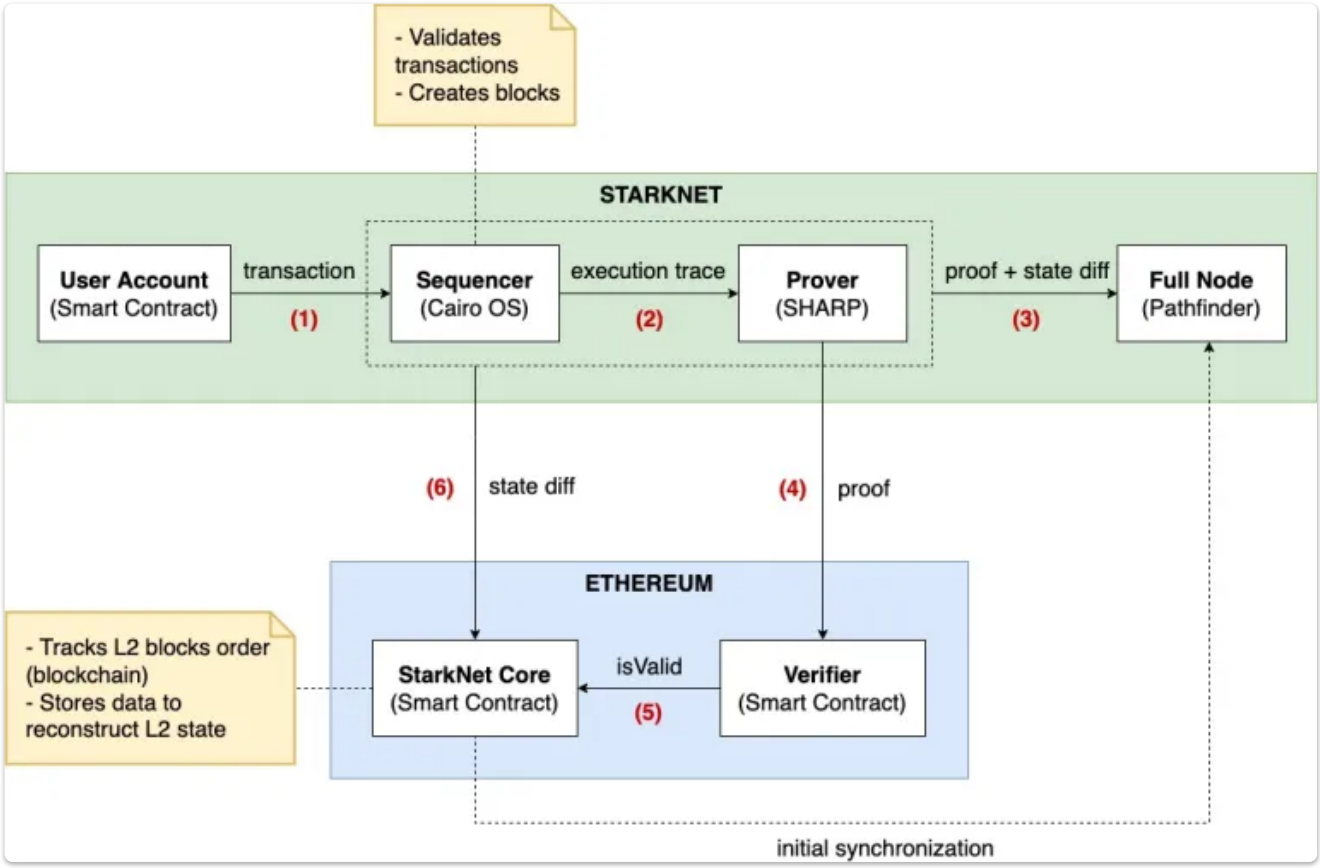
This week

Mon : Contract development continued
Tues : Writing a user interface
Weds : DeFi on Starknet
Thurs : Security / MEV / Messaging bridges

Current ZKP DSLs

1. Circom (Rust)
2. Mina - Snarky.js (Typescript)
3. Zokrates (~Python)
4. Cairo
5. Solidity (Warp)
6. Noir (~Rust)

Transaction process



Transaction Status

- NOT_RECEIVED

Transaction is not yet known to the sequencer

- RECEIVED

Transaction was received by the sequencer. Transaction will now either execute successfully or be rejected.

- PENDING

Transaction executed successfully and entered the pending block.

- REJECTED

Transaction executed unsuccessfully and thus was skipped (applies both to a pending and an actual created block). Possible reasons for transaction rejection:

- An assertion failed during the execution of the transaction (in StarkNet, unlike in Ethereum, transaction executions do not always succeed).

- The block may be rejected on L1, thus changing the transaction status to `REJECTED`

- ACCEPTED_ON_L2

Transaction passed validation and entered an actual created block on L2.

- ACCEPTED_ON_L1
  Transaction was accepted on-chain.

Variables - Tempvar or local

Variables may be aliased or evaluated:

- Aliased
  - Value Reference: `let a = 5`.
  - Expression Reference: `let a = x`.
    - An alias may be evaluated with `assert a = b` (checks x equals b).
- Evaluated
  - Temporary variable: `tempvar a = 5 * b`.
  - Local: `local a = 5 * b`.
  - Constant: `const a = 5`.

1. tempvar

- Based on the allocation pointer so it can be revoked, due to jumps or function calls.
- Is required when using the 'new' operator to reserve memory

```
        func foo() {
    tempvar ptr: MyStruct* = new MyStruct(a=1, b=2);
    assert ptr.a = 1;
    assert ptr.b = 2;
    return ();
}
```

- Can be used with variables that will change during a function such as with a loop
- Can reduce the memory needed

2. local
   - Requires alloc_locals statement
   - Based on frame pointer, so will not be revoked.
   - "Note that unless the local variable is initialized in the same line, the `local` directive itself does not translate to a Cairo instruction (this is another difference from `tempvar`) – it simply translates to a reference definition. This is one of the reasons you must increase the value of `ap` manually."

Contract storage clashes and namespaces

Useful article
Storage is organised by hashing the key value pairs.
This can lead to clashes if a contract uses a library with variables named the same.
Hence the need for namespaces to differentiate between the variables, see notes from Lesson 7.

Dev Tools support

There is a hardhat plugin available
and an experimental Foundry integration
