Verifereum

Formal verification of implementation correctness for applications that run on the Ethereum virtual machine

Ramana Kumar
HOL4 Users' Workshop 2024

What is Ethereum?

The most popular public blockchain.
An application platform that is:

Credibly neutral

- Capture resistant — nobody can change the rules for private benefit
- Censorship resistant — anyone with an internet connection can use it
- Transparent — anyone can examine and verify these properties

Safe and live

- Decentralised
- Incentivised (proof of stake consensus mechanism)

Actively developed + in use

DEXes (decentralised exchanges) (Uniswap, Matcha, CoW, 1Inch), lending markets (Aave, Myso), Stable coins (USDC, USDT, DAI), Liquid Staking (Rocket Pool, Lido), names (ENS, Nouns), attestations (EAS), gamefi, socialfi, desci, …

How Ethereum works

Ethereum is a shared decentralised computer.
There is global consensus on the state of the machine.
State updates obey deterministic rules, driven by user txns.

The state is organised into accounts, containing:

- a 20 byte address
- an ether balance
- storage: a map from `bytes32` to `bytes32`
- (optionally) code: a sequence of bytes

Externally-owned account (EOA)

has private key associated with address, but no code

Smart contract account

has code, but no private key

How Ethereum works: message calls

An account can be called by another account.

Calls, used for both transfers and compute, contain:

- sender address
- target address
- (optional) ether value to transfer
- (optional) data

The target's code is executed with the call data as input

- execution context for stack-based virtual machine with RAM
- may spawn *internal* calls (new contexts), which return data
- transactional: all or none of the storage and balance updates persist

A transaction (txn) from an EOA is a call wrapped in extra info

transaction info, including fee limit, is signed with the private key; fees (gas) are charged for execution and deducted from the sender account

Static calls (no state changes), aka view functions, are free

used to read data from the blockchain; no sender required

How Ethereum works: blocks

The entire history of state updates is a sequence of txns.
These are organised into blocks.
The consensus mechanism selects a block (which also refers to a parent block) as the canonical head of the chain.

Access/participation via a decentralised peer-to-peer network of nodes

each node has a view on the current machine state, agreeing at least on the latest finalised block

Transactions are broadcast to the network

a validating (proof of stake) node is randomly selected every slot (12 seconds) to propose a block that specifies the included transactions and their order

How Ethereum works: ABI

There are conventions for interpreting call data.
The most popular *application binary interface* (ABI) is provided by the Solidity high-level smart contract programming language.

Byte encoding of typed data by convention

`uint256`, `address` (`bytes20`), `bool[]`, ...
32 bytes per value of basic types, length-prefixed encoding of compound types

Function selector: initial 4 bytes of call data

convention: initial bytes of hash of function signature
example: `0xa543ccea` for `setWithdrawalAddress(address,address,bool)`

Example: ERC-20 token

Standard (https://eips.ethereum.org/EIPS/eip-20) minimal interface for (fungible) tokens

```
function name() view returns (string)
function symbol() view returns (string)
function decimals() view returns (uint8)
function totalSupply() view returns (uint256)

function balanceOf(address _owner) view returns (uint256 balance)
function transfer(address _to, uint256 _value) returns (bool success)

function transferFrom(address _from, address _to, uint256 _value) returns (bool success)
function approve(address _spender, uint256 _value) returns (bool success)
function allowance(address _owner, address _spender) view returns (uint256 remaining)
```

Example: 0xsplits Swapper

https://docs.splits.org/core/swapper

EVMs beyond Ethereum

The future of Ethereum applications is on layer 2 rollups.
These are blockchains that settle their state updates as data on Ethereum, typically using ZK SNARK proofs.

The need for layer 2s is scaling

freedom to explore strategies—e.g. less decentralised—for higher throughput

Many layer 2s use the EVM for computation

or at least are EVM-compatible, but some use different models entirely

It remains to be seen what is most adopted

however, it looks like ETH as asset and Ethereum as settlement layer has strong network effects

What is Verifereum?

Infrastructure and techniques for formally verifying
Ethereum applications in HOL4.

This is a great domain for verification

- well-specified platform
- small programs (more code is more expensive to deploy)
- correctness has a very high value: applications protect large sums
- immutability by default: high cost to change, so get it right before deploying

Motivations

- auditing market charges good money: $1000s/day for a report
- many applications have large ($100k–$10m) bug bounties
- correct-by-construction application development
- stronger optimisation (runtime gas usage, code size) with confidence

How Verifereum works: the plan

Initial guiding star: be able to verify (or fail by finding bugs in) any application/protocol implemented as smart contracts on Ethereum (layer 1)

Formal model of the EVM

- machine state and update logic
- keep updated: at least the live EVM version

Program logic for reasoning about applications

- abstract away the rest of the machine
- application-specific top-level theorems

Decompilation of bytecode into logic

- reason about bytecode at a high level

Verifereum: current status

Past: draft EVM model is mostly complete

- missing: precompiled (built-in) contracts
- may need refactoring: relatively unused (for both proof and execution)

Now: attempting to run external tests

- currently trying to prove a very simple VM test
- working on evaluation-in-logic: tests must prove automatically
- aim: pass the whole test suite (as do the actual clients run by nodes)

Next: verify some simple contracts

- example: WETH, one of the simplest ERC20 tokens
- example: Rocket Rebate, a simple contract I made

Get involved

Verifereum on GitHub and Discord

https://github.com/verifereum/verifereum

`#verifereum` channel in CakeML Discord

Possibly starting a company

Verifereum as an alternative (or complement) to auditing

Other Ethereum verification work

- Ethereum Foundation programme with $20m budget
- Goal: formal verification of ZKEVM (for use by ZK layer 2 rollups)
- Lots of interest—in community coming together—in using Lean and RISC Zero