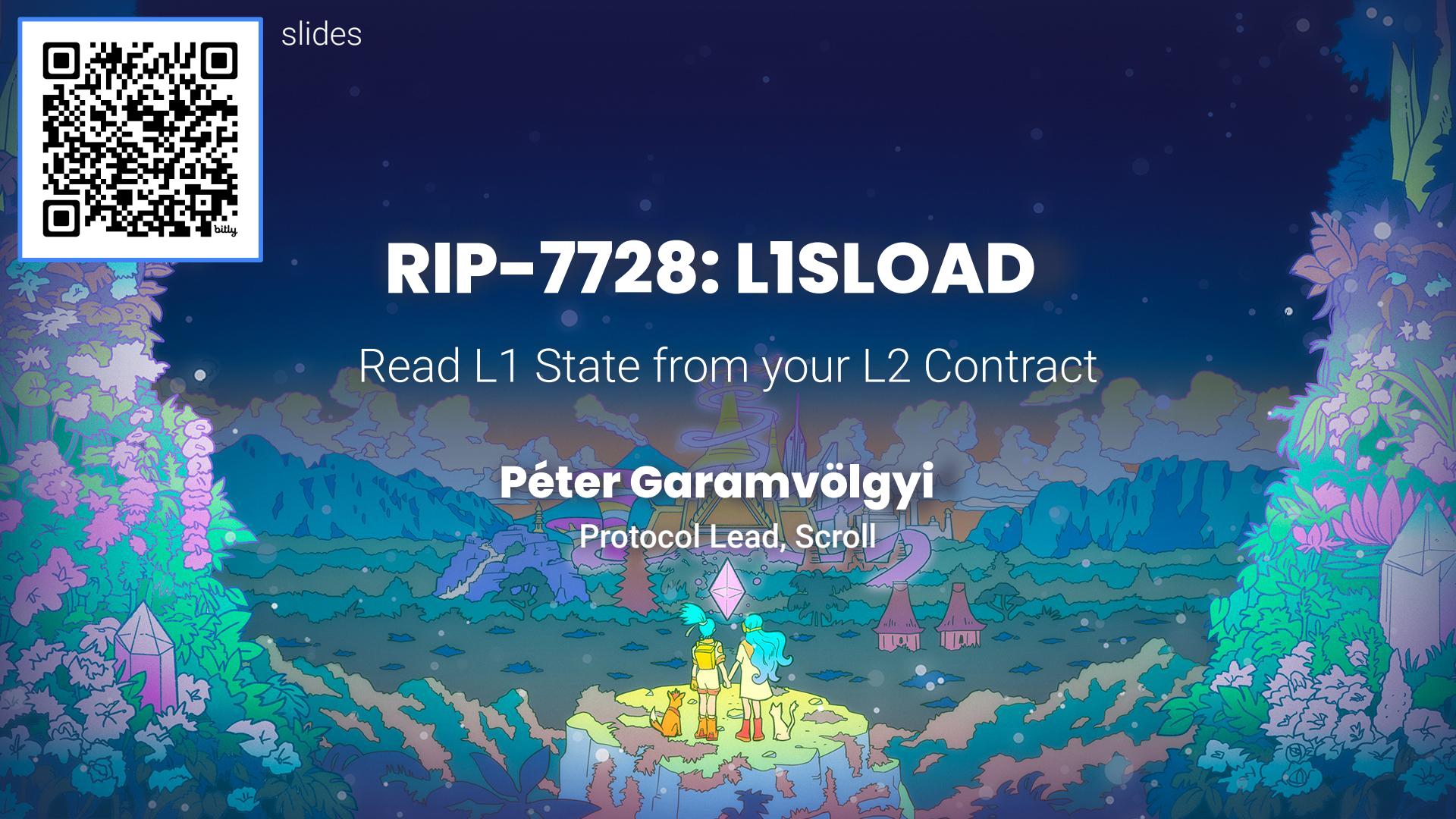
slides

RIP-7728: L1SLOAD

Read L1 State from your L2 Contract



Péter Garamvölgyi

Protocol Lead, Scroll

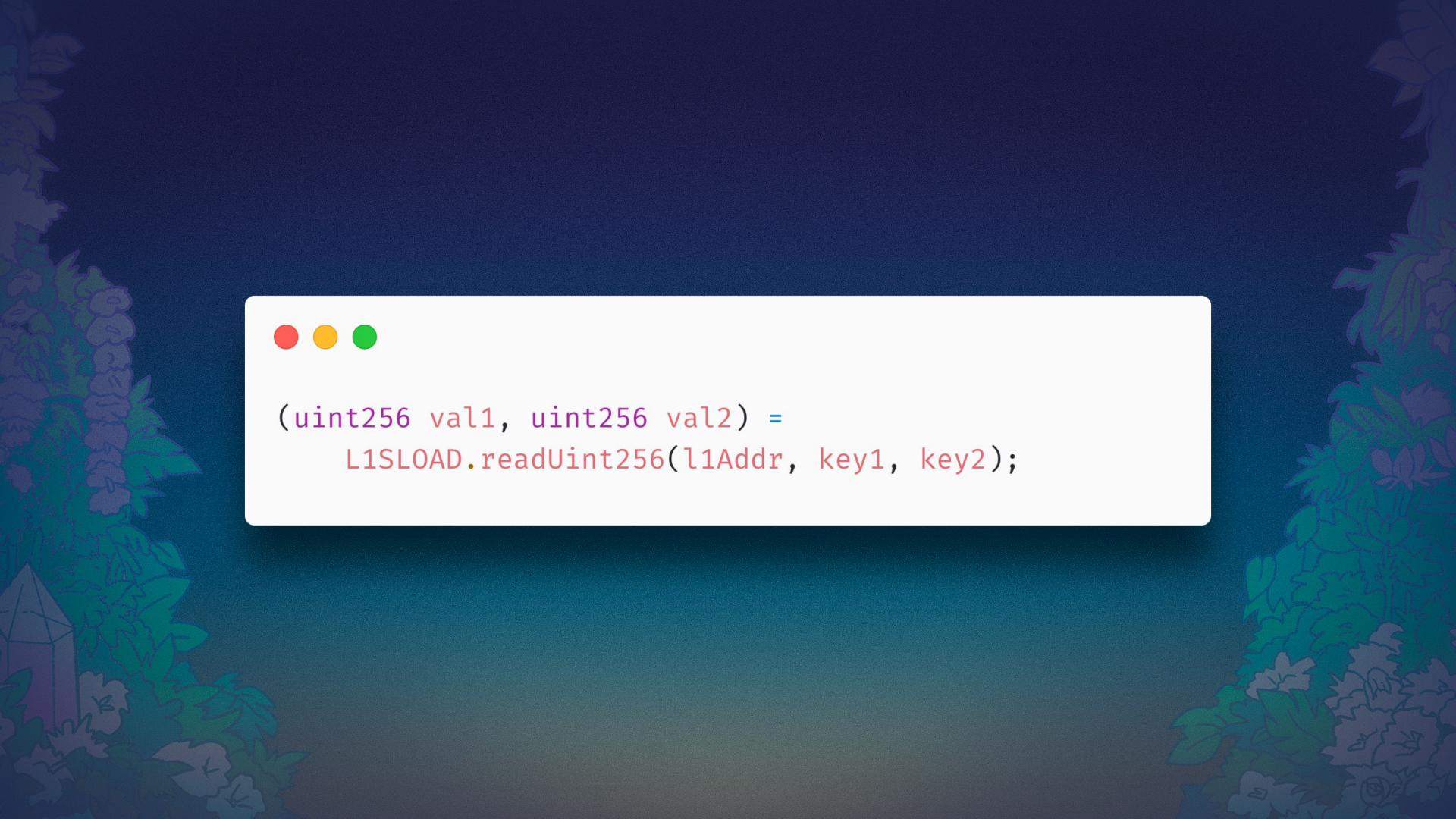
What is L1SLOAD?

L1SLOAD is a precompile for reading L1 state from L2

```
(bool success, bytes memory ret) =
    L1SLOAD.staticcall(
        abi.encodePacked(l1Addr, key1, key2)
    );

if (!succcess) {
    revert("L1SLOAD failed");
}

(uint256 val1, uint256 val2) =
    abi.decode(ret, (uint256, uint256));
```


(uint256 val1, uint256 val2) =
L1SLOAD.readUint256(l1Addr, key1, key2);


STORAGE

Use Case Highlights

Tornado Cash bridge ethglobal.com/showcase/toadnado-vvkcb

Resolve ENS from L2 ethglobal.com/showcase/resolving-ens-on-l2-0a071

Borrow on L2 with L1 collateral github.com/1997roylee/l1-staking-and-l2-borrow

Keystore for AA wallets (Safe, ChatterPay)

```
contract L1Wallet is Wallet {
    mapping(address ⇒ bool) public isAuthorizedSigner;
    // ...
}

contract L2Wallet is Wallet {
  function isAuthorized(address signer) public view override returns (bool) {
        bytes32 mappingSlot = keccak256(abi.encode(signer, 0));
        bool authorized = L1SLOAD.readBool(l1Wallet, mappingSlot);
        return authorized;
    }
    // ...
}
```

Why do L2s need this?

MPT proofs as a sequencer service

Let the sequencer and prover do the heavy lifting

Alternatives to L1SLOAD

There have been similar proposals: L1CALL, remote static call

Too heavy/complex for zk rollups

However, L1CALL works much better with usual ABI stability guarantees

Implementation

Prerequisite feature #1:
Trustless L1 block hash relay

Prerequisite feature #2: Connect to an L1 node

```go
func (c *l1sload) Run(state StateDB, input []byte) ([]byte, error) {
    // load latest l1 block number
    block := state.GetState(rcfg.L1BlocksAddress, rcfg.LatestBlockNumberSlot).Big()

    // parse arguments
    address := common.BytesToAddress(input[0:20])
    // ...

    // execute (batch) RPC call
    res, err := c.l1Client.StoragesAt(context.Background(), address, keys, block)

    // continue EVM execution
    return res, nil
}
```

Verification:

Verify L1 header chain + MPT proofs

Open Questions

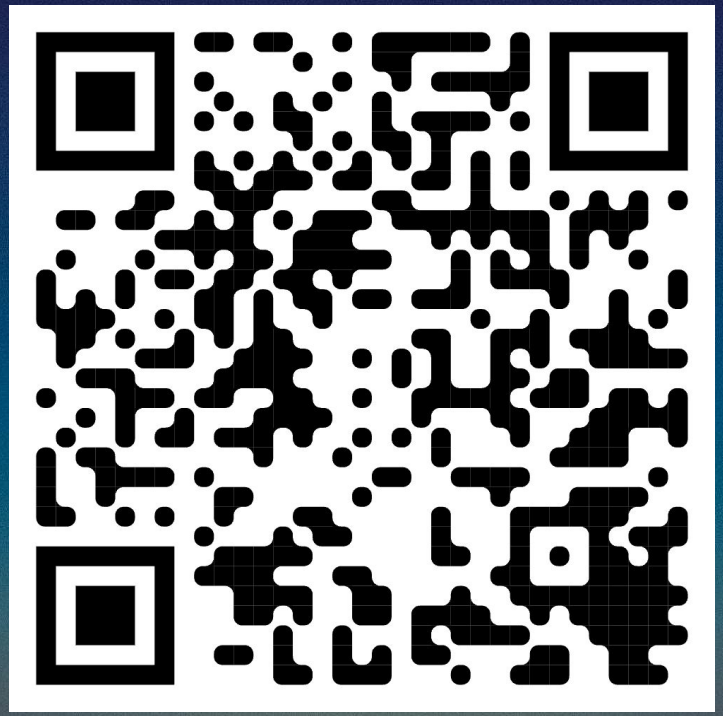
Read state at arbitrary L1 height?

How to deal with RPC errors and latency?

How should we define gas cost?

Where to next?

How to get involved?



Resources

RIP draft
https://github.com/ethereum/RIPs/blob/master/RIPS/rip-7728.md

EthMagicians
https://ethereum-magicians.org/t/rip-7728-l1sload-precompile/20388

Reference implementation draft
https://github.com/scroll-tech/go-ethereum/blob/df5ddd6fef3037c73f3f01375fc2d54a43ac7f94/core/vm/contracts.go#L1166

L1CALL
https://ethresear.ch/t/cross-layer-communication-trivially-provable-and-efficient-read-access-to-the-parent-chain/15396

Remote static call
https://github.com/ethereum-optimism/ecosystem-contributions/issues/76

Resources

Devnet

https://l1sload.scroll.systems

Workshop

https://github.com/scroll-tech/devcon-l1sload-workshop

Intro article

https://dev.to/filosofiacodigoen/l1sload-reading-arrays-structs-and-nested-mappings-4b7h

slides

Thank you!

Péter Garamvölgyi
Protocol Lead, Scroll
peter@scroll.io
@thegaram33