EthernalChat

Robert
Kifal
Eko
Lakunle
Nicolas

Solidity Bootcamp Q2 2024
Group 6

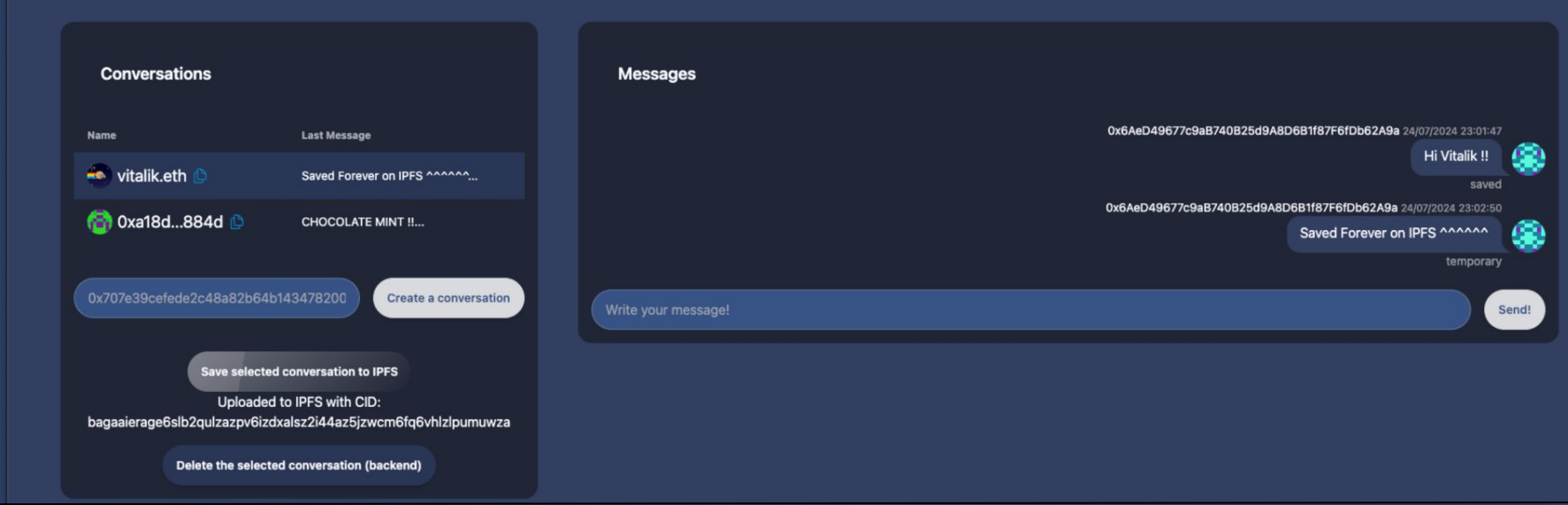
encode CLUB

Eternal messages on IPFS and Ethereum

TABLE OF CONTENTS

The Project

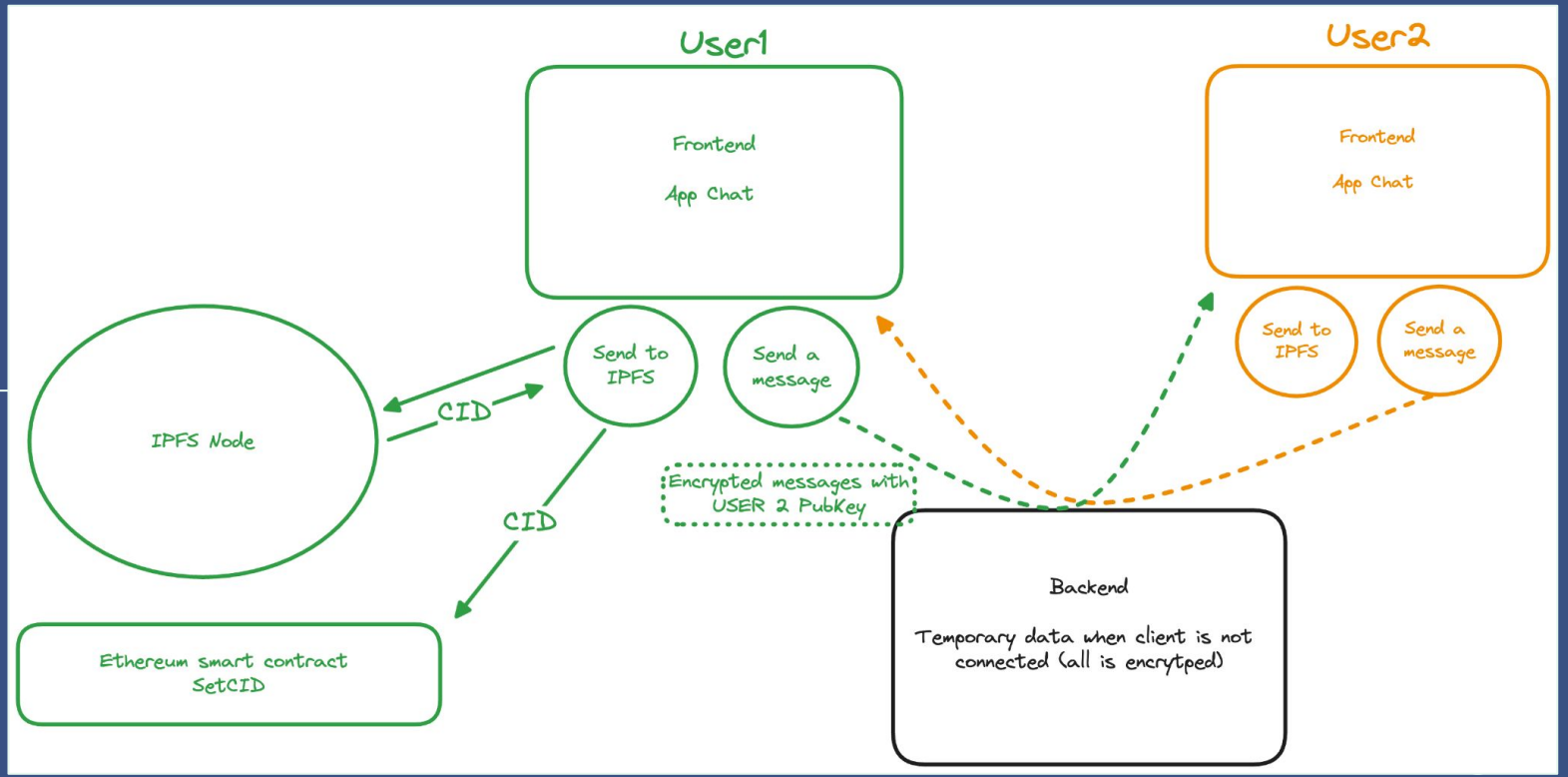
A Simple chat that let you save your encrypted messages on IPFS and talk with people



The CID stored forever

- Data stored on IPFS (Simple Incentive system for node to keep it pinned)
- The CID is stored on an Ethereum Smart Contract (Ensure trustless integrity)
- Backend for temporary encrypted messages (that can be deleted)

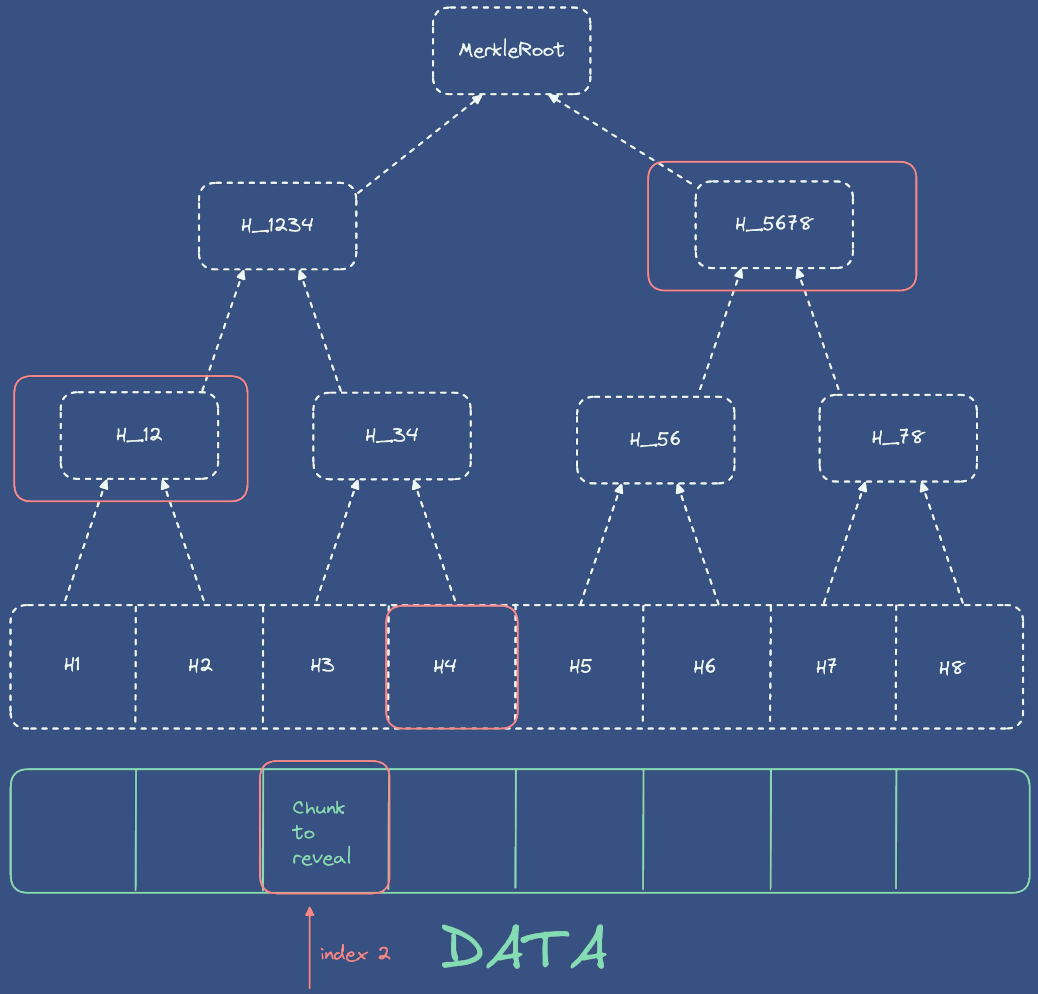
Message Exchange and storage


User1
Frontend
App Chat
User2
Frontend
App Chat
IPFS Node
Send to IPFS
Send a message
CID
CID
Send to IPFS
Send a message
Encrypted messages with USER 2 PubKey
Backend
Temporary data when client is not connected (all is encrytped)
Ethereum smart contract
SetCID

Incentivize Ipfs node

“Simple Storage Proof”

- User divide into chunks and **store the merkle root** on Ethereum.
- A Storage Provider - ipfs node - can every day **ask a challenge** (random index).
- He then can gather reward by providing a proof of storage →



```
function verifyStorageProof(
    uint index,
    bytes32 merkleRoot,
    bytes memory chunkData,
    bytes32[] memory hashes
) internal pure returns (bool) {
    bytes32 chunkHash = keccak256(chunkData);
    for (uint i = 0; i < hashes.length; i++) {
        if (index % 2 == 0) {
            chunkHash = keccak256(abi.encodePacked(chunkHash, hashes[i]));
        } else {
            chunkHash = keccak256(abi.encodePacked(hashes[i], chunkHash));
        }
        index = index / 2;
    }
    return chunkHash == merkleRoot;
}

/// @notice Allows storage to reedem a partial amount of reward in token by answer
/// The full reedmed amount can be withdraw after a month (30days) since lastWitho
/// @param addr Address of the account you are trying to get the reward from.
/// @param hashes Array of hashes allowing the proof : it should be ordered by the
///  it should allow the proof to pass.
function getStorageReward(
    address addr,
    bytes memory chunkData,
    bytes32[] memory hashes
) public OnlyStorageProvider(addr) {
    DataInfo storage dataInfo = mapDataInfo[addr];
    ProviderInfo storage providerInfo = mapProviderInfo[
        dataInfo.storageProvider
    ];
    require(dataInfo.challengeAsked, "No challenge was asked");
    require(
        mapDataInfo[addr].cid != bytes32(0),
        "No DataInfo store by this address"
    );
    require(
        block.timestamp >= dataInfo.lastTimeRewardRedeemed + 1 days,
        "Not Enough time has passed"
```

Show me the code

- **User provide eth** and a storage **Provider can only withdraw Eth monthly.**
- ‘Oops’ we may need **amortization** here… Proof verification is **O(log2(n))** —> n number of chunks

As a reference 1GB = 2^30 bytes

Web App Demo

A few improvements to do

Adding Signatures

Ensuring data integrity from the sender on messages

Improving Ipfs connection

Connecting to Ipfs directly from the frontend, without a node on the backend.

Better Proof of Storage

See Filecoin: Proof of Replication and Proof of spacetime, using ZK proof to off-chain the computation...

THANKS!

Special thanks to
Matheus

And the encode team !

Any Questions ?

encode CLUB