Lesson 10

Week 3

Lesson 9 - What's next in L2 part 2 : L3s/Hyperchains

Lesson 10 - Privacy in Layer 2

Lesson 11 - What are ZK EVMs part 1 - overview

Lesson 12- What are ZK EVMs part 2 - universal circuits/circuit compiler

Introduction

See Coin Telegraph article

'One of the largest remaining challenges in the Ethereum ecosystem is privacy ... In practice, using the entire suite of Ethereum applications involves making a significant portion of your life public for anyone to see and analyze.' - Vitalik

The focus in this area has been on privacy preserving token transfers. Some of the challenges in this area are

- Obfuscate the ownership of an asset.
- Provide a simple means to know that an asset has been transferred to a person.
- Simplify key management

Why do we want privacy ?

- Confidential transactions
- Working with private data, for example with machine learning.

Use cases

- Sealed bid auctions
- Programmable confidentiality for DeFi
- Dark pools
- Encrypted name services

> **Privacy, confidentiality, anonymity**
>
> Aztec give some useful definitions of Privacy, Anonymity and Confidentiality
>
> Privacy: all aspects of a transaction remain hidden from the public or third parties.
>
> Confidentiality: the inputs and outputs of a transaction are hidden from the public but the transaction parties remain public.
>
> Anonymity: the inputs and outputs of a transaction are public but the transaction graph is obscured from one transaction to the next, preventing the identification of the transaction parties.

Privacy Techniques

Obfuscation

One way functions such as hash functions provide a way to obfuscate a pre image thanks to the hiding property of the commitment scheme.

Confidential computing from Intel

See Docs
Focussing on hardware assisted solutions such as trusted execution environments, these are used more for centralised systems (though see Obscuro)

FHE

Fully homomorphic encryption allows us to perform operations on encrypted data as if it were plain text.
This would permit in theory use to encrypt the state of a blockchain and perform operations (such as the state transitions from a transaction in the EVM) on the encrypted data to produce a new encrypted state.
The owner of the state would have the encryption key and could therefore decrypt the state.
This technique is used by Zama.ai and Aztec.

Commitment schemes and nullifiers

A general approach to allow ownership and transfer of assets to be maintained without revealing ownership is through commitment schemes and nullifiers.

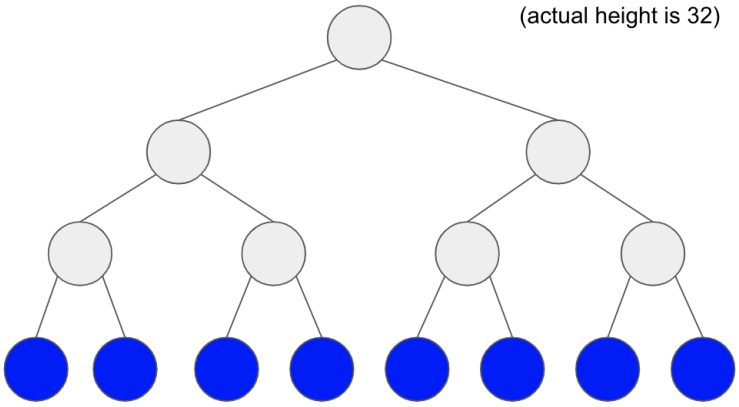
The general process is

- Commitments to items are added to a data structure such as a merkle tree. The commitment hides details of the asset and the owner of the

asset. This is the merkle tree usded in ZCash

The Global Merkle Tree of Commitment Notes

- Holds all “commitment notes”
  - similar to Bitcoin’s UTXO model
- A Pedersen hash of the **value** of the note, and its **owner**
- **Only additive**
  - Spending a note does not remove it from the tree (nullifier set’s role)
- Like the UTXO set, Miners have to update their local Merkle Tree based on incoming transactions



leaves are note note commitments (cm)

- When the asset is transferred, the commitment is 'cancelled' by the use of a nullifier and a new commitment created with the new owner. The nullifier is deterministically derived from the original asset details, and is added to a nullifier set. The presence of the nullifier indicates that the asset from which it has been derived has been spent.
  This is roughly the process used by ZCash and Tornado Cash.

Projects emphasising privacy

Zama.ai

See Docs
See White paper
Zama have created a fhEVM incorporating FHE into the operation of the EVM.
Specifically

- An fhEVM, which integrates Zama's open source FHE library TFHE-rs nto a typical EVM, exposing homomorphic operations as precompiled contracts.
- A decryption mechanism based on distributed threshold protocols that prevents misuse by malicious smart contracts, yet is flexible and non- intrusive to honest smart-contract developers.
- A Solidity library that makes it easy for smart-contract developers to use encrypted data in their contracts, without any changes to compilation tools.

Users can create a private blockchain uses this EVM as the execution layer.
It would be interesting to see this combined with other chains via some off the interoperability mechanisms we have seen.

Aztec

See Site
Execution is performed off chain, and state is encrypted and private to its owner.

Namada

See Docs
See Vision
Namada is a protocol which supports multi-chain asset-agnostic privacy
Namada uses the CometBFT BFT consensus algorithm. Namada enables

multi-asset private transfers for any native or non-native asset using a multi-asset shielded pool derived from the Sapling circuit.

Obscuro

See Docs

This is a L2, where all transactions and the internal state of application contracts are encrypted and hidden. Execution is performed by a bespoke encrypted EVM.

Penumbra

See Docs

Penumbra is a shielded, cross-chain network allowing private trading in crypto assets.

Penumbra records all value in a single multi-asset shielded pool based on the Zcash Sapling design, but allows private transactions in any kind of IBC (see IBC protocol) asset.

Inbound IBC transfers shield value as it moves into the zone, while outbound IBC transfers unshield value.

Anoma

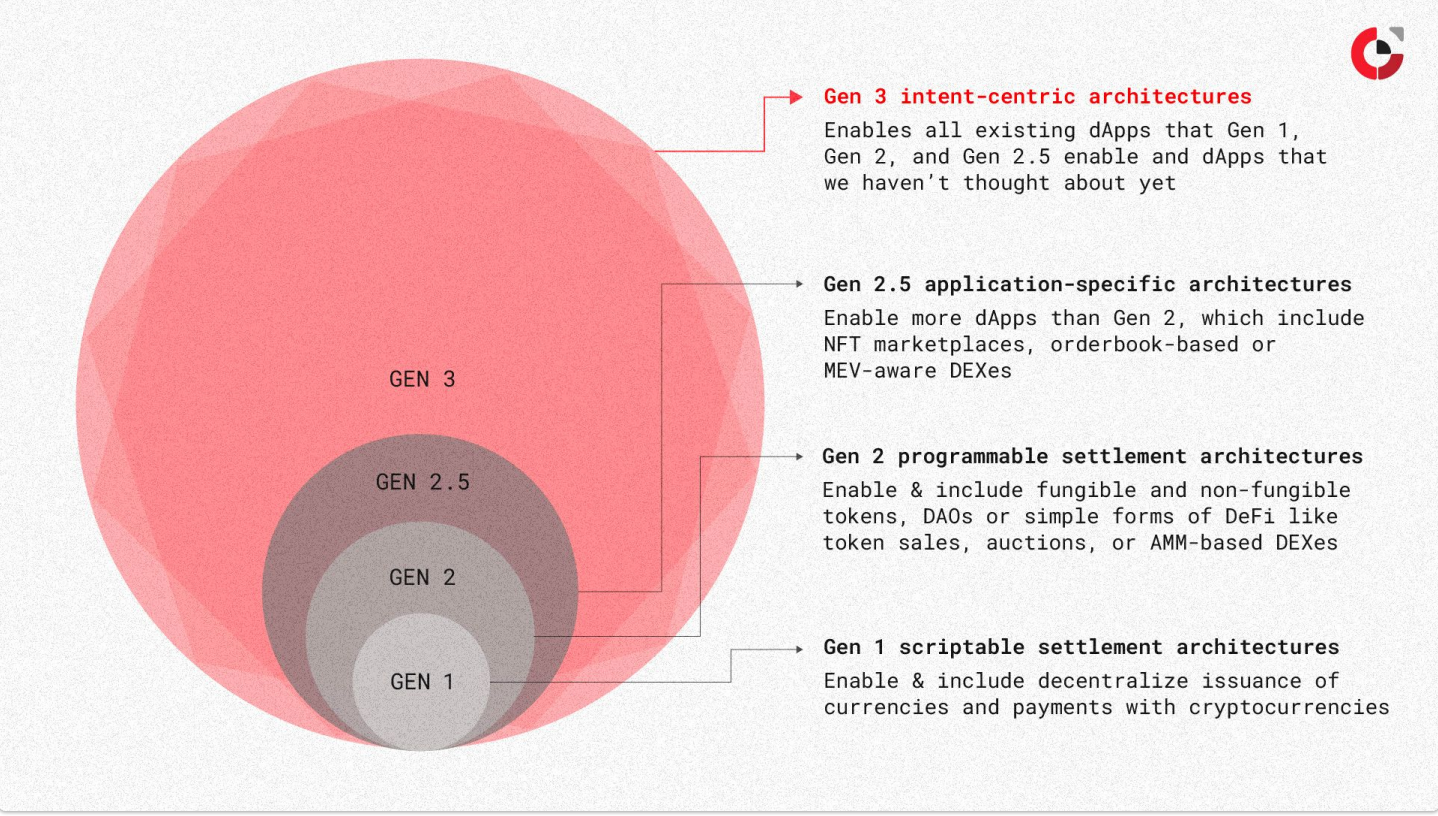
See paper

See research

Listen to ZK Podcast

See Blog

Anoma is designed around the concept of an Intent centric architecture



Anoma design features

1. Intents: Users supply transaction intentions, which are matched with counterparts.
2. Privacy Measures:
   Transaction and user details are kept confidential
3. Cross-Chain Interactions:
   Anoma supports cross chain interactions
4. MTCS:
   Multilateral Trade Credit Set-Off, this allows for the offsetting of multiple obligations simultaneously, optimising liquidity usage.
5. Atomic Settlements:
   Interaction across chains is atomic.