# LSD 、EigenLayer、跨链桥

登链社区 - Tiny熊

# 要点

- 什么时候应该开发自己的链
- 链或基础设置面临的问题
- 如何跨链
- 拓展：
  - 如何获取一个安全的随机数 Chainlink VRF
  - 如何让合约获取外部数据

“什么时候应该启动一条链？”

# 启动自己的链

- 自身业务量大，需更高 TPS
- 需要有更好的控制权
- 自定义需求，如隐私、合规等
- 例如：
  - Base、HaskKey、Unichain、WorldCoin、Robinhood(Arbitrum) - Layer2
  - BNB Chain 、 支付巨头 Stripe （Tempo）、稳定币发行商 Circle （Arc） - EVM 兼容 Layer1
  - dYdX V4 - Cosmos Layer1
  - Monad （平行 EVM）、Sui

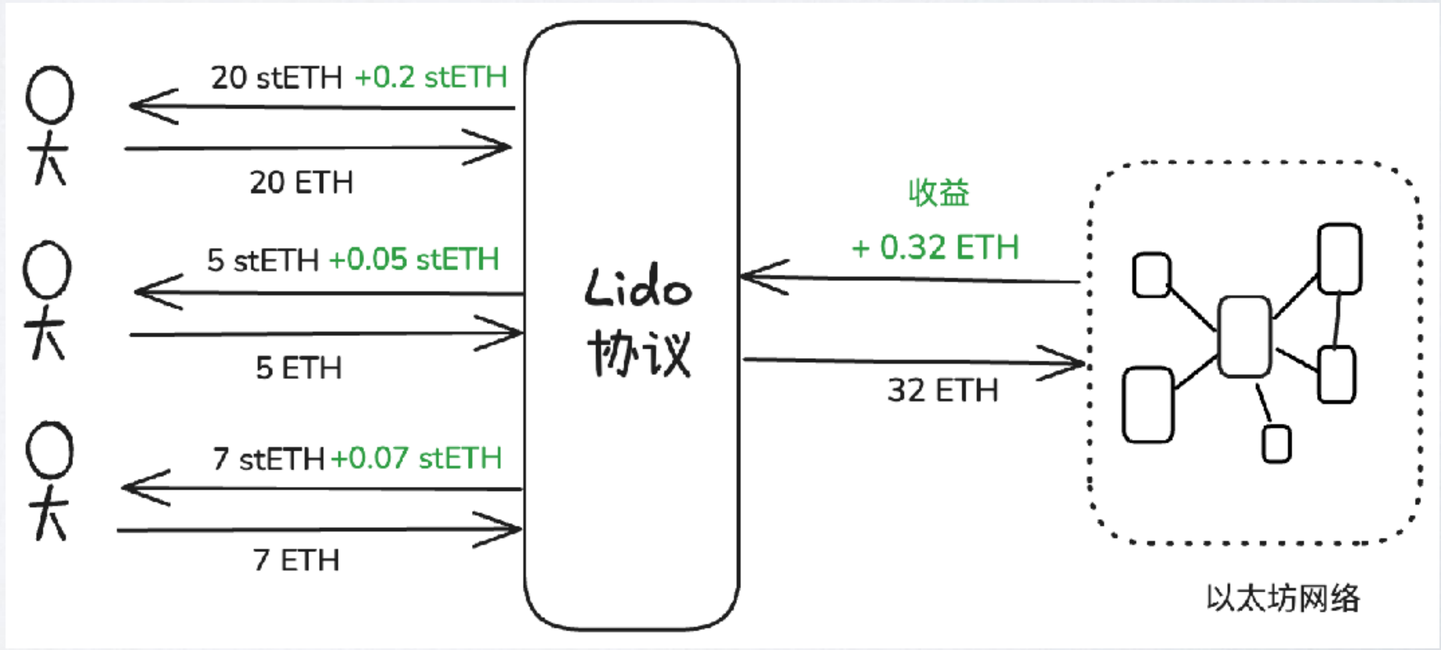
# 启动链面临问题

- 如何确保链的安全性（预言机网络、桥服务等也有类似安全性问题）：
  - POW：足够的算力
  - **POS**：足够的质押
- 启动新的 PoS 网络很困难，抵押者必须放弃其他的收益机会（DEFI 、以太坊原生收益 ）、同时面临项目币更大的波动性。
- 如何解决呢?
  - 可否保留其他质押收益时，同时维护网络安全 （restaking）

# LSD

- 个人质押（Solo Staking）：以太坊质押 32 ETH、成本较高
- LSD： Liquid Staking Derivatives （流动性质押衍生品）- 业务模式统称
  - LST：Liquid Staking Token（流动性质押代币）： 如 stETH 、rETH
  - LSP：Liquid Staking Protocol（流动性质押协议）， 如 Lido 、Rocket Pool
- 用户在 LSP 存入 ETH，协议会帮你质押到验证者节点，并发行对应的 LST

stETH每天根据收益 rebase
增加 stETH 数量

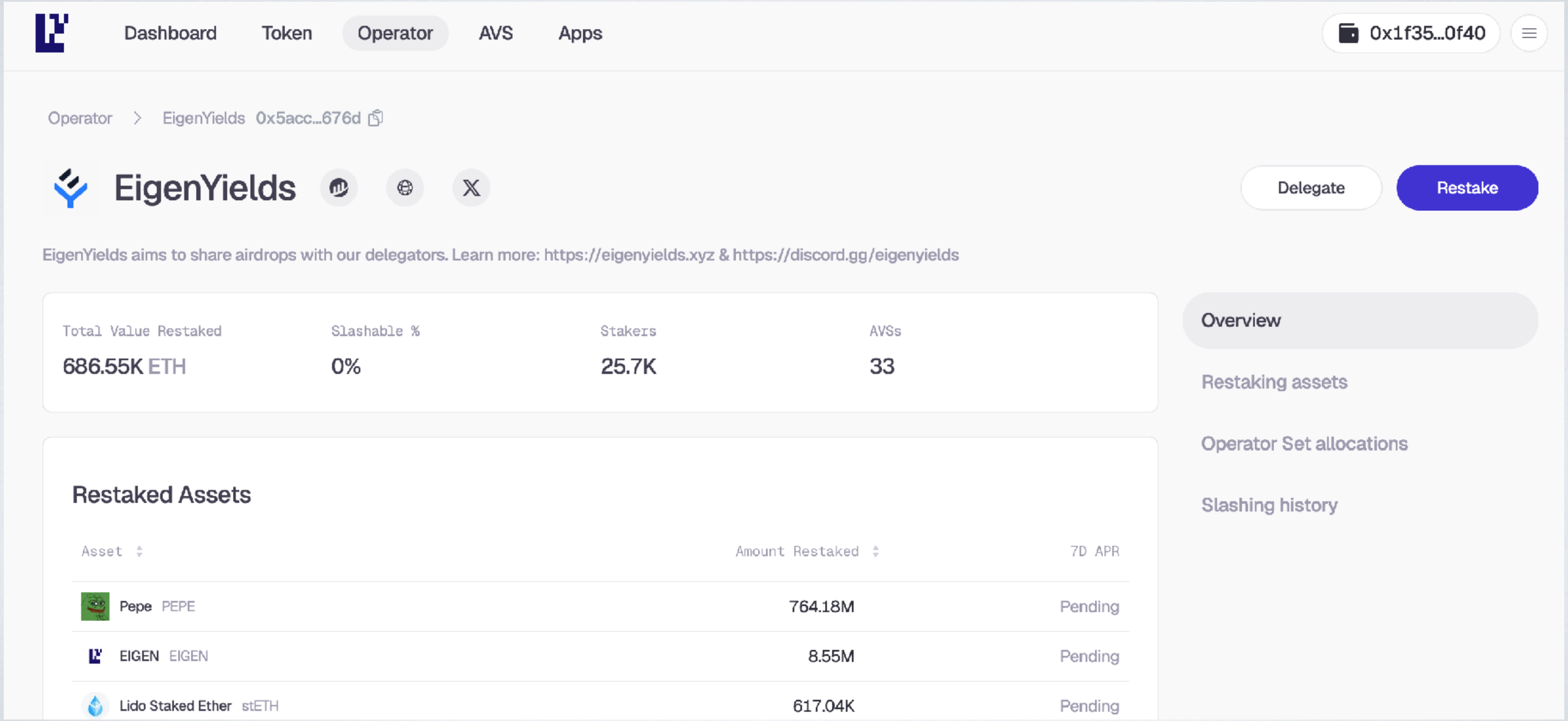


# 再质押（re-staking）

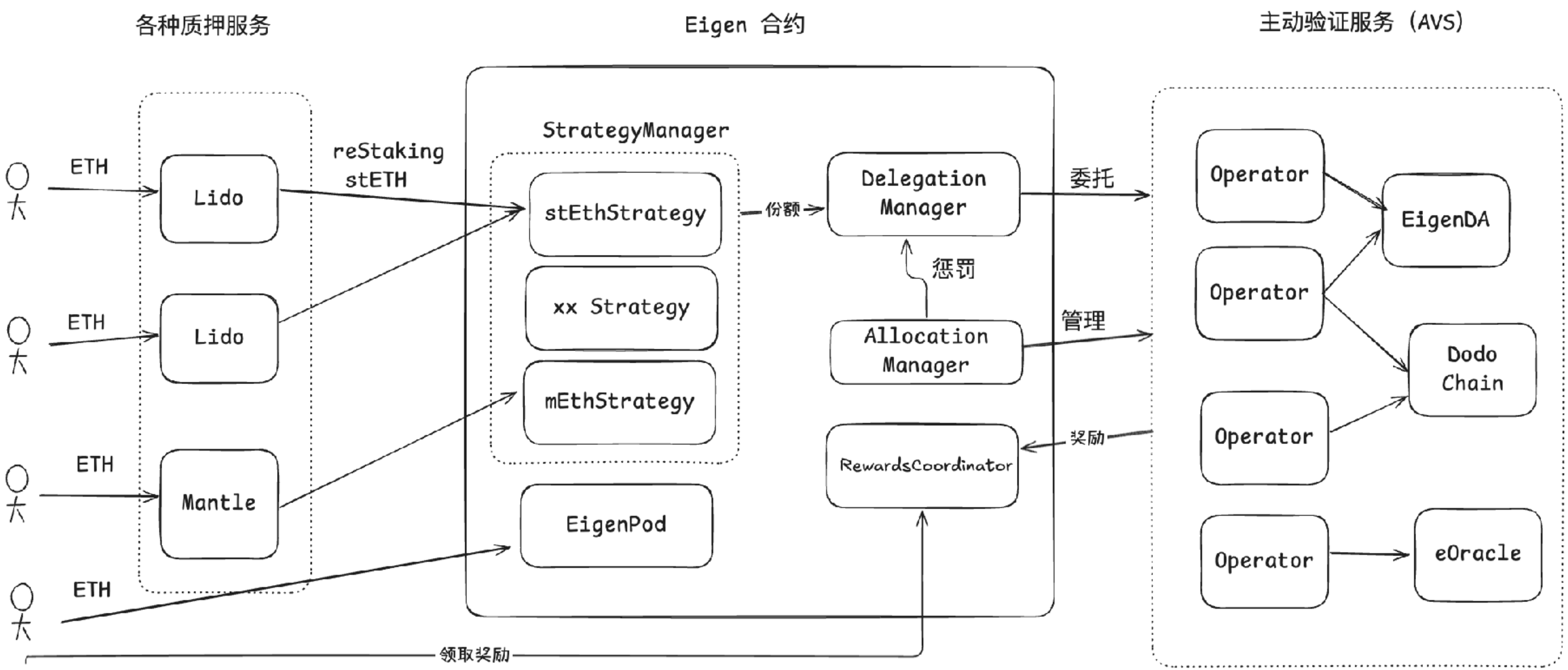
- Restaking : 用户（staker）将自己的权益 Token， 质押到 restaking 协议（EigenLayer / Symbiotic）获取份额。
- 然后将份额委托给 Operator， Operator 来支持基础网络的安全。
- 这样，用户在不损失原有收益的情况下，又可获取新的网络收益，同时也有更多资金保护网络的安全
- 但，如果 Operator 作恶， 用户（staker）有罚没的风险。

https://app.eigenlayer.xyz/

# Eigenlayer

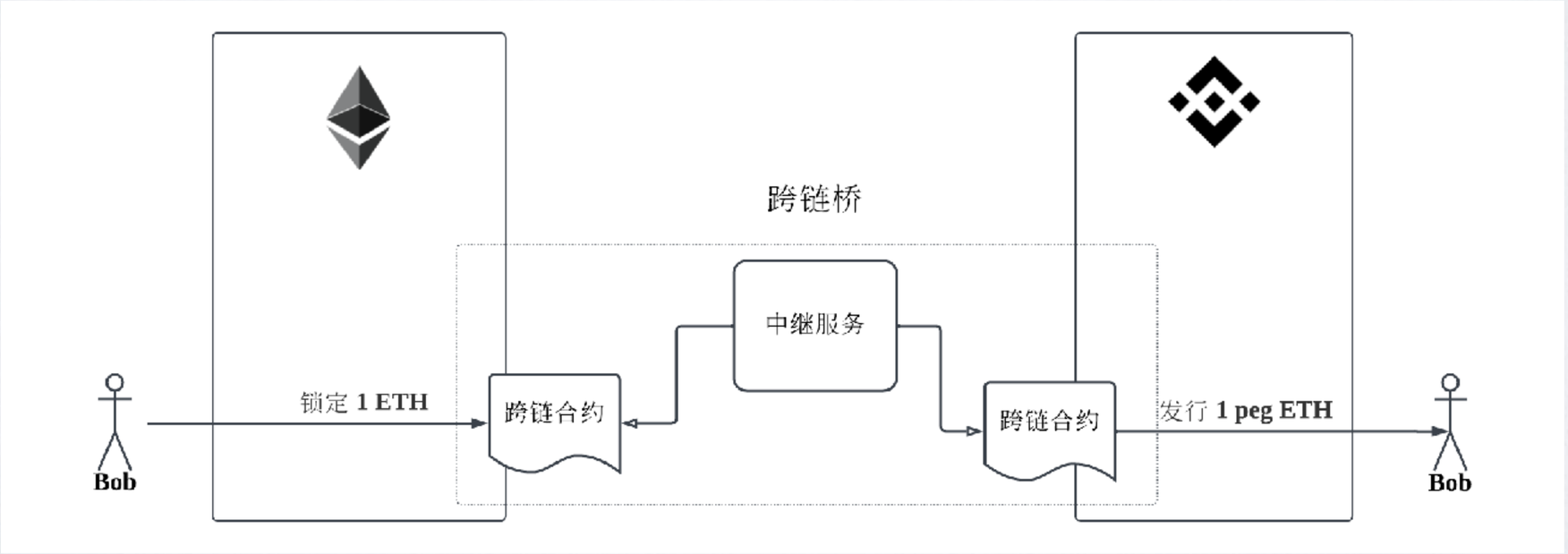

Dashboard
Token
Operator
AVS
Apps
0x1f35...0f40
Operator > EigenYields 0x5acc...676d
EigenYields
Delegate
Restake
EigenYields aims to share airdrops with our delegators. Learn more: https://eigenyields.xyz & https://discord.gg/eigenyields
Total Value Restaked
686.55K ETH
Slashable %
0%
Stakers
25.7K
AVSs
33
Overview
Restaking assets
Operator Set allocations
Slashing history
Restaked Assets
Asset
Amount Restaked
7D APR
Pepe PEPE
764.18M
Pending
EIGEN EIGEN
8.55M
Pending
Lido Staked Ether stETH
617.04K
Pending

# EigenCloud (EigenLayer)



# 跨链桥

- 不同的链，就像一个独立的岛屿，要实现互通，需要在岛屿间建桥
- 桥：在源链触发一个事件消息(由链下的中继服务处理)，事件消息指示在另一条链如何执行。
  - Token 跨链: 指示如何在另一条链释放 Token



# 跨链桥关键问题

- 如何确保消息在目标链会被执行?
- 如何确保消息在目标链仅执行一次?
- 如何确保消息来自于源链的指示?
- 早期跨链桥多采用较中心化方案，问题：
  - 跨链需许可、项目方私钥泄露遭受攻击

# 跨链主流方案

- LayerZero
  - 基于 LayerZero的 DVN（Decentralized Verifier Networks）和 Executors
- Chainlink CCIP
  - 利用 Chainlink 节点网络（3 个独立的网络）
- Layer2 原生跨链桥、Hyperlane、Wormhole 、CirCLE CCTP
- Across(v4)： ZK 桥， Succinct 的 zkVM (SP1) 实现了以太坊状态的 ZK 证明

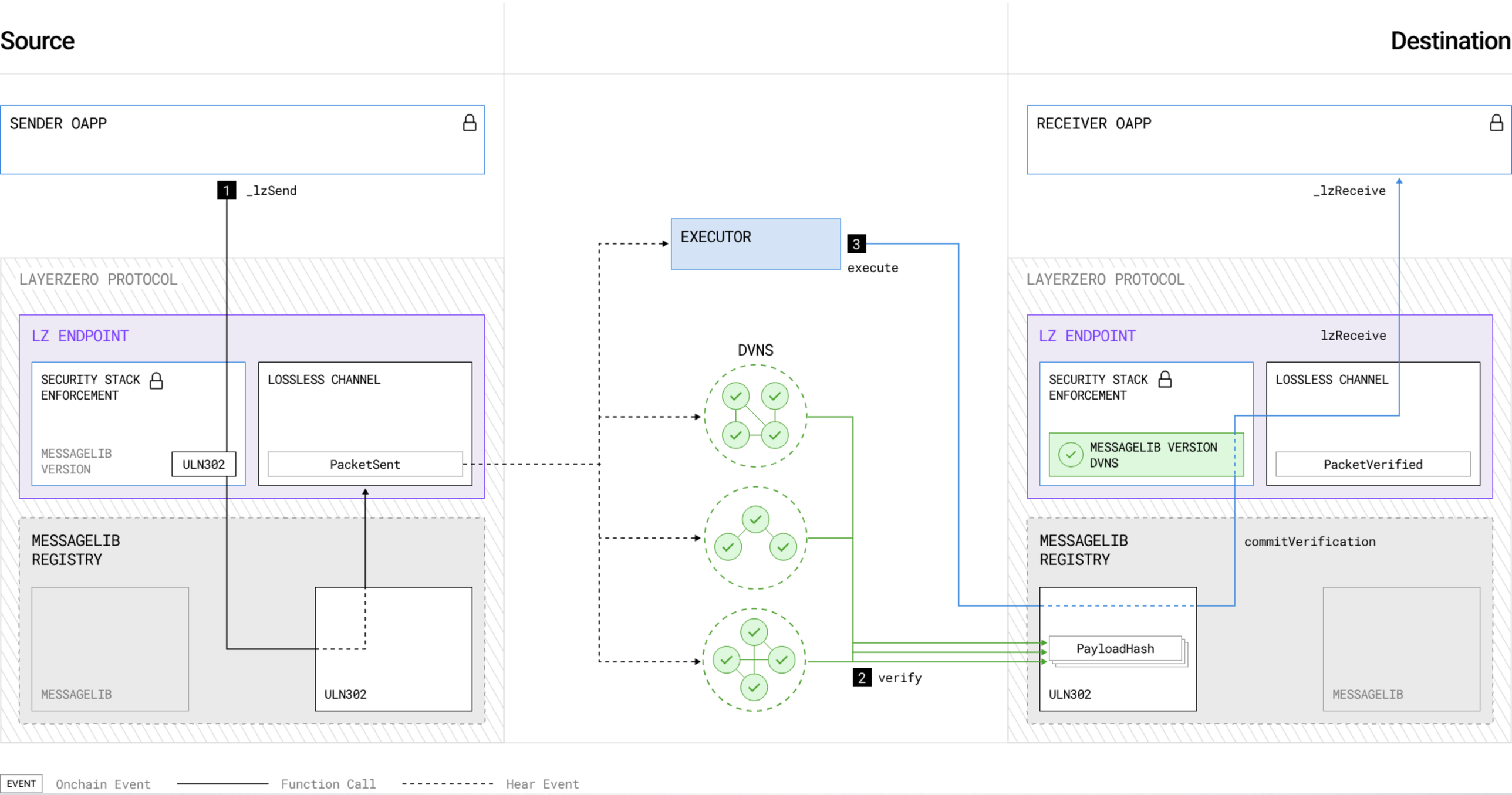
# 跨链应用

- 资产跨链
- 跨链兑换：
  - 一些 DEX（去中心化交易，如：Uniswap） 会集成跨链服务
  - Across 、Stargate 、deBridge、Orbiter Finance
- 跨链抵押借贷：
  - AAVE（based CCIP） 、Radiant（based Layer Zero ）

https://www.bridgewtf.com/

# LayerZero 跨链过程

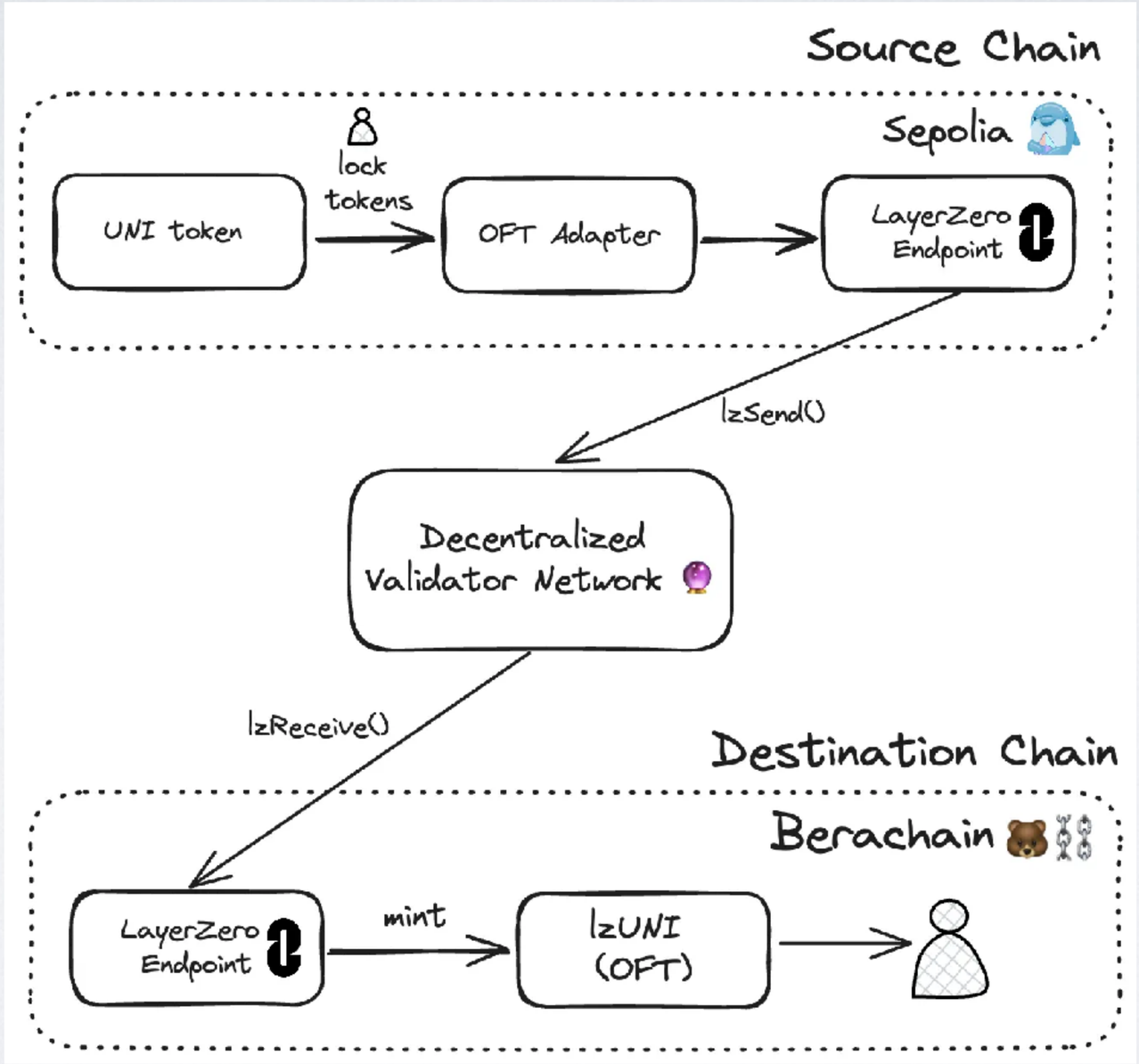
- 在支持的链上部署 LayerZero Endpoint（不可升级）， 处理消息的路由
  - Endpoint 提供了 管理安全配置和发送/接收消息的标准化接口
- 用户在原链合约中调用 LayerZero  Endpoint 的 lzSend()
- DVN （任何人都可加入 DVN） 验证跨链消息， 在目标链上标记验证状态
- 执行器（任何人）在目标链上调用 LayerZero  Endpoint 的 lzReceive() 完成跨链


Source
Destination
SENDER OAPP
RECEIVER OAPP
1 _lzSend
_lzReceive
EXECUTOR
3 execute
LAYERZERO PROTOCOL
LAYERZERO PROTOCOL
LZ ENDPOINT
LZ ENDPOINT
lzReceive
DVNS
SECURITY STACK ENFORCEMENT
LOSSLESS CHANNEL
SECURITY STACK ENFORCEMENT
LOSSLESS CHANNEL
MESSAGELIB VERSION
ULN302
PacketSent
MESSAGELIB VERSION DVNS
PacketVerified
MESSAGELIB REGISTRY
MESSAGELIB REGISTRY
commitVerification
PayloadHash
MESSAGELIB
ULN302
ULN302
MESSAGELIB
2 verify
EVENT Onchain Event Function Call Hear Event

# LayerZero

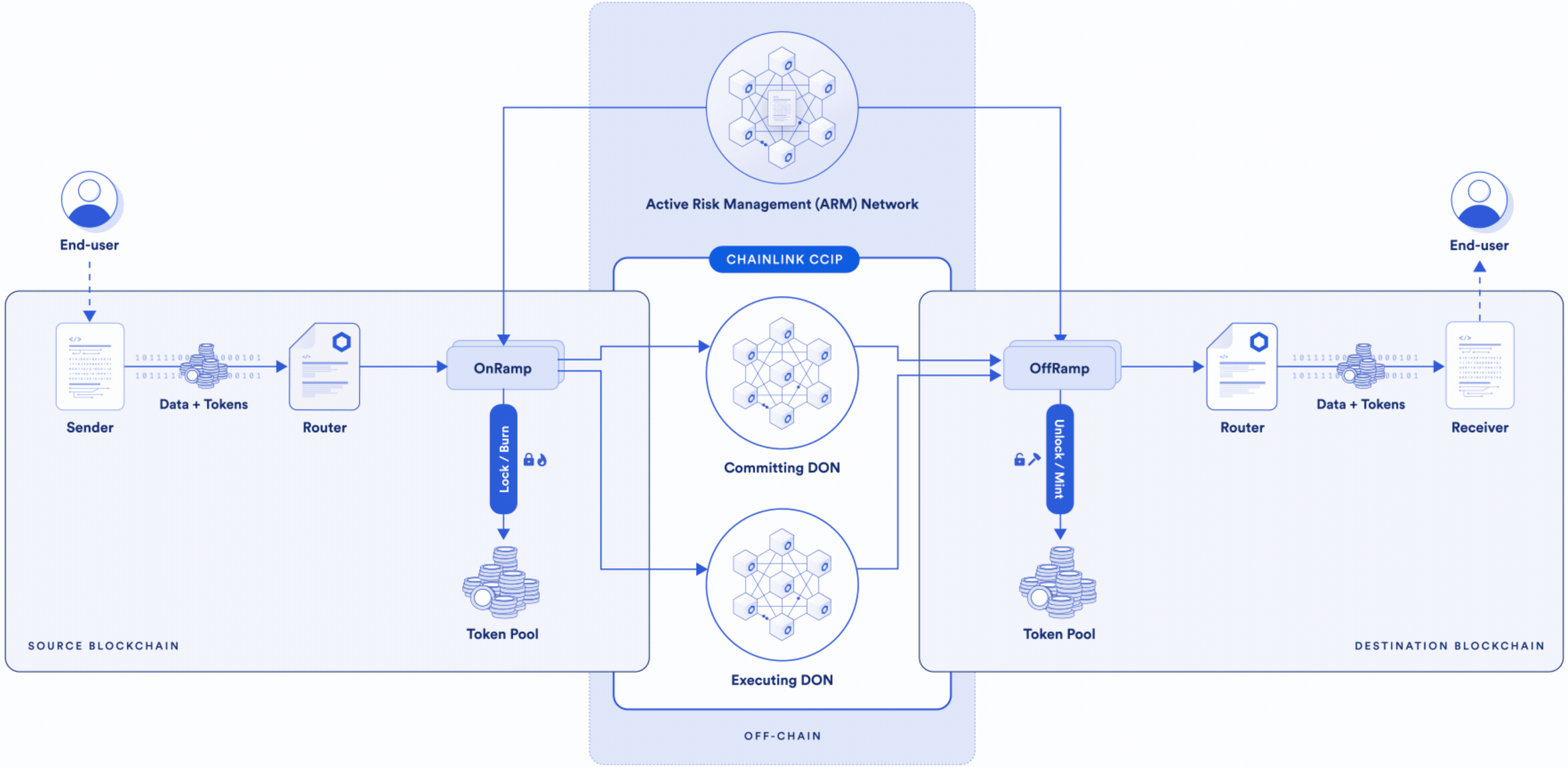
- LayerZero 为跨链应用定义了一些标准
- OApp： 成为跨链应用
- OFT 、ONFT：多个区块链上无缝跨链的代币
  - send() : burn or lock , 然后通过发送一条跨链路由消息
  - setPeer() : 与目标链建立关联（Owner 操作）， 双向设置
- 现有的 Token 跨链，将 Token 转入的 adapter 合约锁定， 由 adapter 实现跨链路由， 通知目标链铸造对应的 Token（需要在目标链上部署 OFT 合约）
- 命令行工具： create-lz-oapp 方便创建跨链应用

# LayerZero


Source Chain
Sepolia
lock
tokens
UNI token
OFT Adapter
LayerZero
Endpoint
lzSend()
Decentralized
Validator Network
lzReceive()
Destination Chain
Berachain
LayerZero
Endpoint
mint
lzUNI
(OFT)

# Chainlink CCIP

- 利用 Chainlink 节点网络，将一条的数据发布到另一条链
- 用户在原链合约中调用 Router.ccipSend()，如果是 Token 转账，锁定或燃烧相应的 Token，Router 将跨链信息转给 OnRamp 处理：生成messageId 并触发 CCIPMessageSent 事件
- Committing DON(提交网络) 监听 CCIPMessageSent 事件，在目标链的 OffRamp.commit() 提交聚合的承诺（确保“源链发生了某件事”），触发 CommitReportAccepted 事件
- Risk Management DON：确认跨链消息正确传递，则 blessed 、如发现伪造消息等，则 cursed
- Executing DON (执行网络) 在目标链确认了源链事件后，开始真正的执行，例如解锁Token，


Active Risk Management (ARM) Network
CHAINLINK CCIP
End-user
End-user
Sender
Data + Tokens
Router
OnRamp
Lock / Burn
Token Pool
Committing DON
Executing DON
OffRamp
Unlock / Mint
Token Pool
Router
Data + Tokens
Receiver
SOURCE BLOCKCHAIN
DESTINATION BLOCKCHAIN
OFF-CHAIN

# CCIP Token 跨链步骤

- 部署 Token， 新 Token 可使用 CCT: Cross-Chain Token 标准， 增加了 mint、burn 等方法
- Token Pools： 在源链和目标链上部署对等的 Token Pools 合约
  - 在源链上 lockOrBurn 在 目标链 releaseOrMint
- 在 CCIP 验证管理员，管理员设置 TokenAdminRegistry 的 setpool 关联 token 的 Token pool， 设置 tokenpool 及跨链传输参数
- 将 Token 授权给 router， 并调用 router.ccipSend 跨链

跨链 Demo: https://github.com/lbc-team/CCIP_Foundry_demo/blob/main/deploy_guide.md

交易 tx

# Cosmos 与 Polkadot

- 曾经的“天王”项目，专门用来做跨链的链， 然后并没有得到理想的采用率。
- Cosmos SDK： 用来构建应用链，可使用 IBC : Inter-Blockchain Communication protocal 接入跨链能力，直接调用另一条链的合约
  - 典型的项目： dYdX v4 、 Celestia 、 Terra（Luna） 、Osmosis ...
- Polkadot Substrate: 用来构建应用链，XCM 跨链

# 链上获取安全随机数

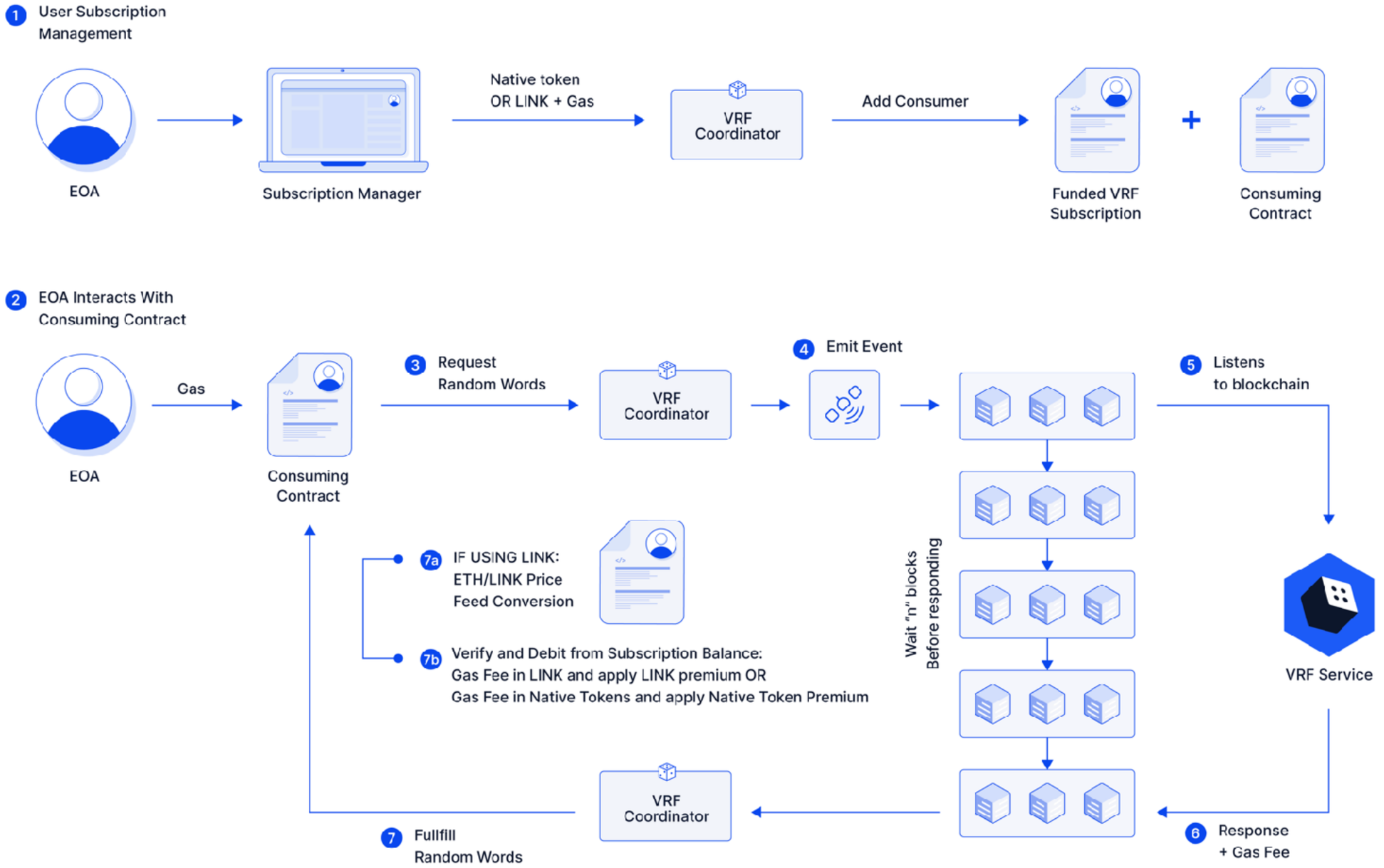
# 通常方法

- 方法 1： 使用 blockhash、timestamp 或合约内数据计算 hash 作为随机数
  - 容易会被矿工/验证者操纵。
- 方法 2： 使用承诺 - 揭示方案（ commit - reveal ）
  - 先提交随机数承诺（例如随机数 Hash)
  - 再提交随机数（r），并用承诺验证随机数。

# Chainlink VRF

- VRF (Verifiable Random Function)： 可验证的随机函数 随机性不可预测、结果可验证、没人能篡改
- 实现原理：
  - 用户合约向 VRFCoordinator 合约提交随机数请求（keyHash + seed），并触发事件
  - Chainlink 节点监听事件，由 keyHash 对应的节点， 用自己的“私钥 + seed” 生成随机数 r
  - 节点将随机数及随机数证明（r, proof）提交给 VRFCoordinator 合约, VRFCoordinator 进行验证
  - VRFCoordinator 将随机数返回给用户合约 fulfillRandomness

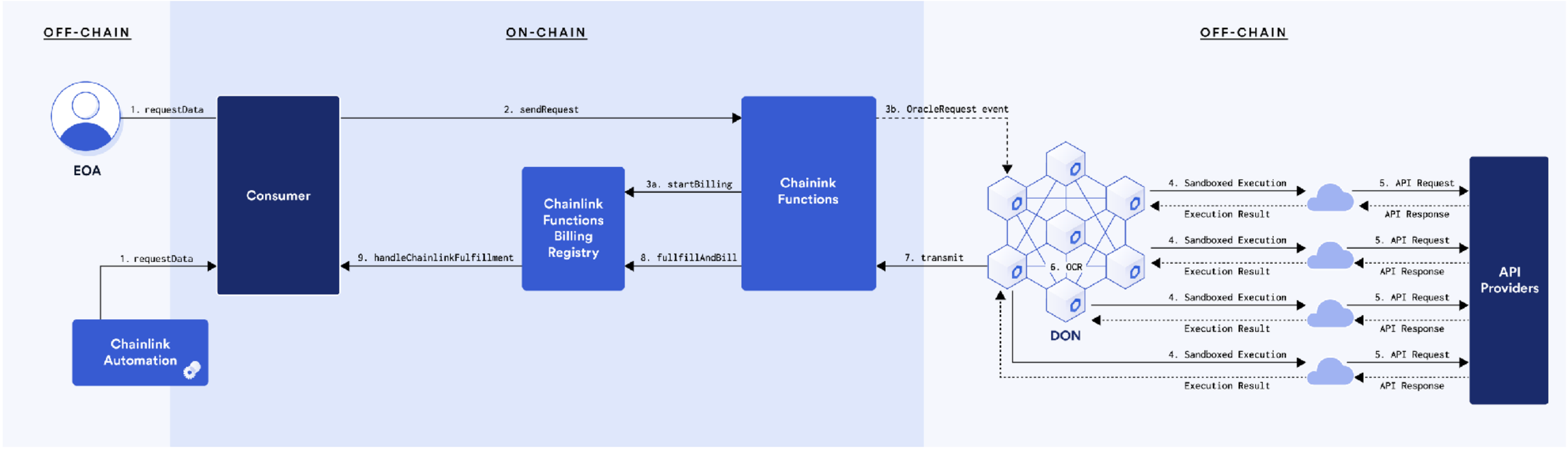
https://docs.chain.link/vrf


1 User Subscription Management
EOA
Subscription Manager
Native token OR LINK + Gas
VRF Coordinator
Add Consumer
Funded VRF Subscription
Consuming Contract
2 EOA Interacts With Consuming Contract
EOA
Gas
Consuming Contract
3 Request Random Words
VRF Coordinator
4 Emit Event
5 Listens to blockchain
VRF Service
Wait "n" blocks Before responding
6 Response + Gas Fee
VRF Coordinator
7 Fullfill Random Words
7a IF USING LINK: ETH/LINK Price Feed Conversion
7b Verify and Debit from Subscription Balance: Gas Fee in LINK and apply LINK premium OR Gas Fee in Native Tokens and apply Native Token Premium

# Chainlink Functions

- 如何让合约安全获取链下数据，如获取某 API 结果（ EVM 不提供对外访问功能）。
- Chainlink Functions 就是一个通用的 off-chain compute + data fetch 框架
- 实现原理：
  - 合约（consumer）调用 Functions Router 合约，请求某个“function”，请求中可包含 JavaScript 脚本、参数 、API 等逻辑。
  - 请求会广播给 Chainlink 去中心化 oracle 网络 (DON)
  - 节点执行你的请求
  - DON 把结果和加密证明一起提交给 Functions Router 合约
  - Router 把结果回调给你的 Consumer 合约

https://docs.chain.link/chainlink-functions


OFF-CHAIN
ON-CHAIN
OFF-CHAIN
EOA
1. requestData
Consumer
2. sendRequest
Chainlink Functions Billing Registry
3a. startBilling
Chainink Functions
3b. OracleRequest event
1. requestData
Chainlink Automation
9. handleChainlinkFulfillment
8. fullfillAndBill
7. transmit
6. OCR
DON
4. Sandboxed Execution
Execution Result
5. API Request
API Response
API Providers