CCIP Conceptual Overview

Before you explore how Chainlink CCIP works in thearchitecture guide , it is best to understand the core concepts.

Prerequisites

Before you learn about Chainlink CCIP, you should understand these fundamental blockchain concepts:

- Accounts : you should understand the differences between Externally-owned accounts (EOA) and Contract accounts.
- Smart Contracts .
- Decentralized Applications (dApps) .
- ERC-20 Token Standard .
- Merkle Trees .

Cross-Chain dApps

Cross-Chain Decentralized Applications (Cross-Chain dApps) aredecentralized applications designed to interact with multiple blockchain networks using cross-chain interoperability protocols. Each application leverages the unique strengths of its underlying blockchain network, providing a unified and seamless user experience. Cross-Chain dApps enable transactions, communication, and data sharing between blockchains, increasing functionality and cooperation between multiple blockchain ecosystems.

Interoperability

Interoperabilityis the ability to exchange information between different systems or networks, even if they are incompatible. Shared concepts on different networks ensure that each party understands and trusts the exchanged information. It also considers the concept offinalityto establish trust in the exchanged information by validating its accuracy and integrity.

Theweb3 ecosystem has become multi-chain, with the rise of layer-1 blockchains and layer-2 scaling solutions, where each network has its own approach to scalability, security, and trust. However, blockchains are isolated networks that operate independently and cannot communicate natively with traditional systems to make external API calls or interact with other blockchains. This limitation is known as theoracle problem . This limitation creates the need for blockchain interoperability protocols, which enable connectivity between different blockchain networks.

CCIP uses interoperability to help web3 developers overcome blockchain restraints, enable specialization of web3 applications, and allow dApps to leverage the liquidity and benefits of multiple blockchain ecosystems.

To learn more, read aboutblockchain interoperability andcross-chain smart contracts .

Finality

Finalityis the assurance that past transactions included onchain are extremely difficult or impossible to revert. If the parameters for finality are properly set, the likelihood of reversibility is extremely low. For CCIP, source chain finality is the main factor that determines the end-to-end elapsed time for CCIP to send a message from one chain to another.

Finality varies across different networks. Some networks offer instant finality and others require multiple confirmations. These time differences are set to ensure the security of CCIP and its users. Finality is crucial for token transfers because funds are locked and not reorganized once they are released onto the destination chain. In this scenario, finality ensures that funds on the destination chain are available only after they have been successfully committed on the source chain.

Lane

A Chainlink CCIPlaneis a distinct pathway between a source and a destination blockchain. Lanes are unidirectional. For instance,Ethereum Sepolia => Polygon MumbaiandPolygon Mumbai => Ethereum Sepoliaare two different lanes.

Decentralized Oracle Network (DON)

Chainlink Decentralized Oracle Networks, orDONs, runChainlink OCR2 . The protocol runs in rounds during which an observed data value might be agreed upon. The output of this process results in areportwhich is attested to by a quorum of participants. The report is thentransmittedonchain by one of the participants. No single participant is responsible for transmitting on every round, and all of them will attempt to do so in a round-robin fashion until a transmission has taken place. In the context of CCIP, alane contains two OCR DON committees that monitor transactions between a source and destination blockchain: theCommitting DONandExecuting DON. Read theArchitecture page to learn more.

Risk Management Network

The Risk Management Network is built using offchain and onchain components:

- Offchain :Several Risk Management nodes continually monitor all supported chains against abnormal activities
- onchain :One Risk Management contract per supported CCIP chain

Offchain Risk Management node

The Risk Management Network is a secondary validation service parallel to the primary CCIP system. It doesn't run the same codebase as theDON to mitigate against security vulnerabilities that might affect the DON's codebase. The Risk Management Network has two main modes of operation:

- Blessing:Each Risk Management node monitors all Merkle roots of messages committed on each destination chain. The Committing DON commits these Merkle roots. (More information on Merkle roots can be found on thearchitecture page). The Risk Management node independently reconstructs the Merkle tree by fetching all messages on the source chain. Then, it checks for a match between the Merkle root committed by the Committing DON and the root of the reconstructed Merkle tree. If both Merkle roots match, the Risk Management node blesses the root to the Risk Management contract on the destination chain. The Risk Management contract tracks the votes. When a quorum is met, the Risk Management contract dubs the Merkle rootblessed.

- Cursing:If a Risk Management node detects an anomaly, the Risk Management node will curse the CCIP system. After a quorum of votes has been met, the Risk Management contract dubs the CCIP systemcursed. CCIP will automatically pause on that chain and wait until the contract owner assesses the situation before potentially lifting the curse. There are two cases where Risk Management nodes pause CCIP:

- Finality violation:A deep reorganization which violates the safety parameters set by the Risk Management configuration occurs on a CCIP chain.

- Execution safety violation:A message is executed on the destination chain without any matching transaction being on the source chain. Double executions fall into this category since the executing DON can only execute a message once.

Onchain Risk Management contract

There is one Risk Management contract for each supported destination chain. The Risk Management contract maintains a group of nodes authorized to participate in the Risk Management blessing/cursing.

Each Risk Management node has five components:

- An address for voting to curse
- An address for voting to bless
- An address for withdrawing a vote to curse
- A curse weight
- A blessing weight

The contract also maintains two thresholds to determine the quorum for blessing and cursing. There are two different voting logics depending on the mode:

- Blessing voting procedure:every time a Risk Management node blesses a Merkle root, the Risk Management contract adds the blessing weight for that node. If the sum of the weights of votes to bless exceeds the blessing threshold, the Risk Management contract considers the contractblessed.
- Cursing voting procedure:a Risk Management node that sends a vote to curse assigns the vote a random 32-byte ID. The node may have multiple active votes to curse at any time. However, if there is at least one active cursing vote, the Risk Management contract considers the node to have voted to curse. The Risk Management contract adds the cursing weight for that node. If the sum of the weights of votes to curse exceeds the curse threshold, the Risk Management contract considers the contractcursed.

If the Risk Management contract is cursed, then the owner of the original contract must resolve any underlying issues the original contract might have. If the owner is satisfied that these issues have been resolved, they can revoke the cursing on behalf of Risk Management nodes.