Chainlink CCIP for Cardano

From Solidity to Aiken: Patterns and Practices - March 2025

Kasey White

Chainlink CCIP

What is Chainlink CCIP?

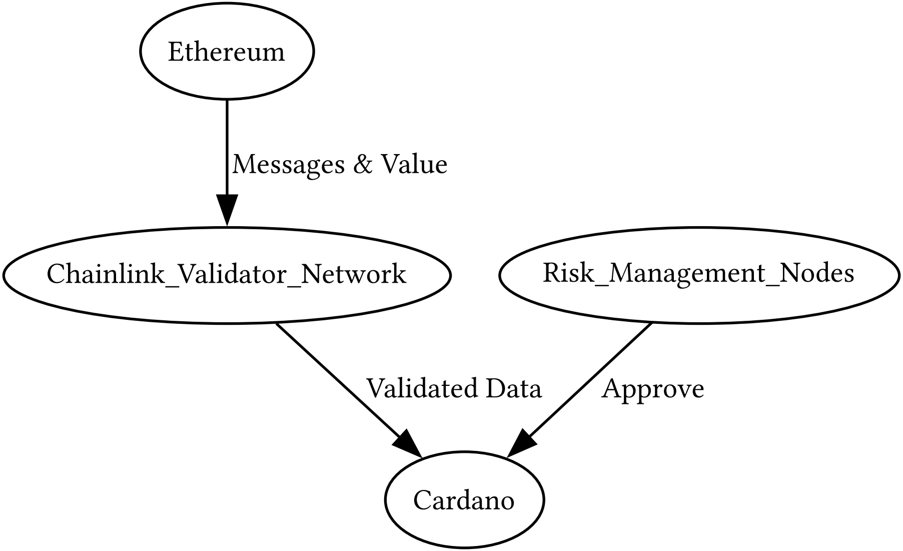
- Cross-chain Interoperability Protocol (CCIP) facilitates the transmission of messages containing data and value across different blockchains

- Said messages can be used to call contracts on other chains

- The Messages follow the EVM2ANY format with the binary abi encoding from EVM

- Messages are validated by chainlink validator network and are double checked by a multisig Risk management set of nodes

Architecture Layout



CCIP Message Structure

Solidity Message Structure

```
struct Client {
  bytes32 sender;
  uint64 nonce;
  uint256
sourceChainSelector;
  address receiver;
  bytes data;
  EVMTokenAmount[]
tokenAmounts;
}
struct EVMTokenAmount {
  address token;
  uint256 amount;
}
```

Aiken Equivalent (Preview)

```
type Client {
  sender: ByteArray,
  nonce: Int,
  source_chain_selector:
Int,
  receiver: ByteArray,
  data: ByteArray,
  token_amounts:
List<TokenAmount>,
}
type TokenAmount {
  token: ByteArray,
  amount: Int,
}
```

Converting Solidity to Aiken

Common patterns in Solidity

- Proxy contracts
- State mapping i.e. (address => struct)
- abi encoding of structs and keccak_256 hashing
- explicit integer sizes
- Merkle Trees

Translating Solidity Patterns to Aiken

Proxy Contracts

- Direct script lookup via `scriptContext` enables action proxying
- Memory in UTXOs with NFTs - easier to manage than Solidity's address storage

State Mapping

- Per-UTXO approach for mapping elements
- Transactions only include relevant UTXOs
- Space costs managed via Merkle trees or element lists in single UTXOs

ABI Encoding & Hashing

- Custom encoding logic required in Aiken
- Native keccak_256 builtin matches Solidity functionality

Translating Solidity Patterns to Aiken (Cont.)

Integer Handling

- Aiken integers have unlimited size - no overflow concerns
- Use fixed-size bytearrays for bitwise operations

Merkle Trees

- We have existing merkle tree implementations in Aiken
- Was tackled in previous Aiken projects (Fortuna)

Implementation Challenges

- Global state UTXO contention (solutions in development)
- Recursion makes looping more expensive than Solidity
- Focus on business logic over gas optimization
- Rely on unit/property testing for correctness

Summary

Summary

- Since there’s little to no effort or expertise on Cardano from Chainlink, we can take action ourselves to do the bulk of the effort

- The work used to do conversions can apply to other cross-chain protocols and standards like LayerZero, Wormhole, or the Token Bridge Standards Alliance

- We now have concrete ways to transform Solidity to Aiken contracts to emulate the same behavior (the newer builtins added recently help a lot)

Appendix

CCIP Terminology

Term	Definition
EVM2ANY format	A standardized message format that enables communication between EVM-based chains and non-EVM chains (like Cardano). It provides a common structure for cross-chain data exchange.
Binary ABI encoding	The Application Binary Interface encoding method used by Ethereum to serialize and deserialize data.
Validator network	A decentralized set of Chainlink nodes that verify cross-chain messages, ensuring data integrity and preventing malicious transactions.

Contact & Resources

- **Email**: byword-hitters.9h@icloud.com

- **Twitter**: @Microproofs

- **GitHub**: github.com/Microproofs

Project Links

- Aiken Prototype
- Pragma Discord
- Chainlink CCIP Documentation
- Aiken Language Documentation