




// CHAINLINK DEVELOPER EXPERTS

Building Cross-Chain with CCIP



Chris Adams

Chainlink Developer Expert

@0xdallascats 

dallascats 

Agenda

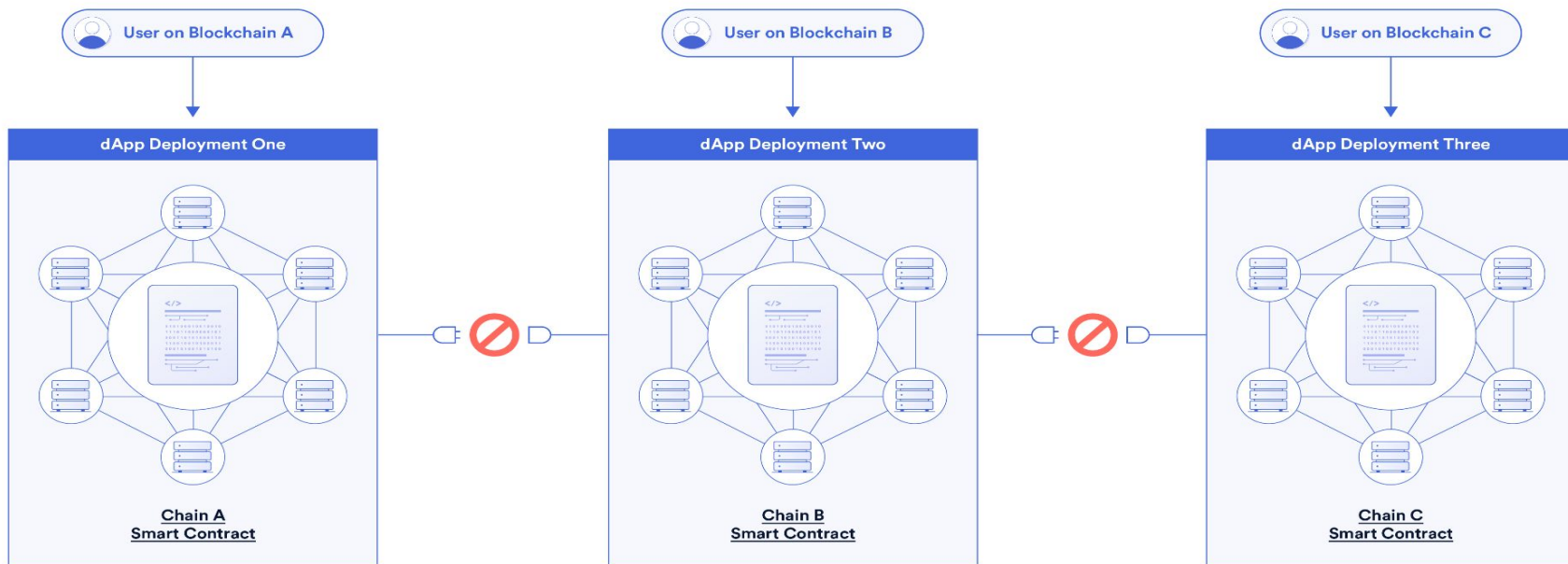
1. **Blockchain Interoperability Problem**
2. Chainlink CCIP Intro
 - CCIP Core Capabilities
 - CCIP Examples
3. Token Transfer Exercise
4. CCIP Architecture in Depth



// Building Cross-Chain with CCIP

Blockchain Interoperability Problem

Multi-Chain Decentralized Applications





// Building Cross-Chain with CCIP

CCIP Introduction

- Core Capabilities
- CCIP Examples

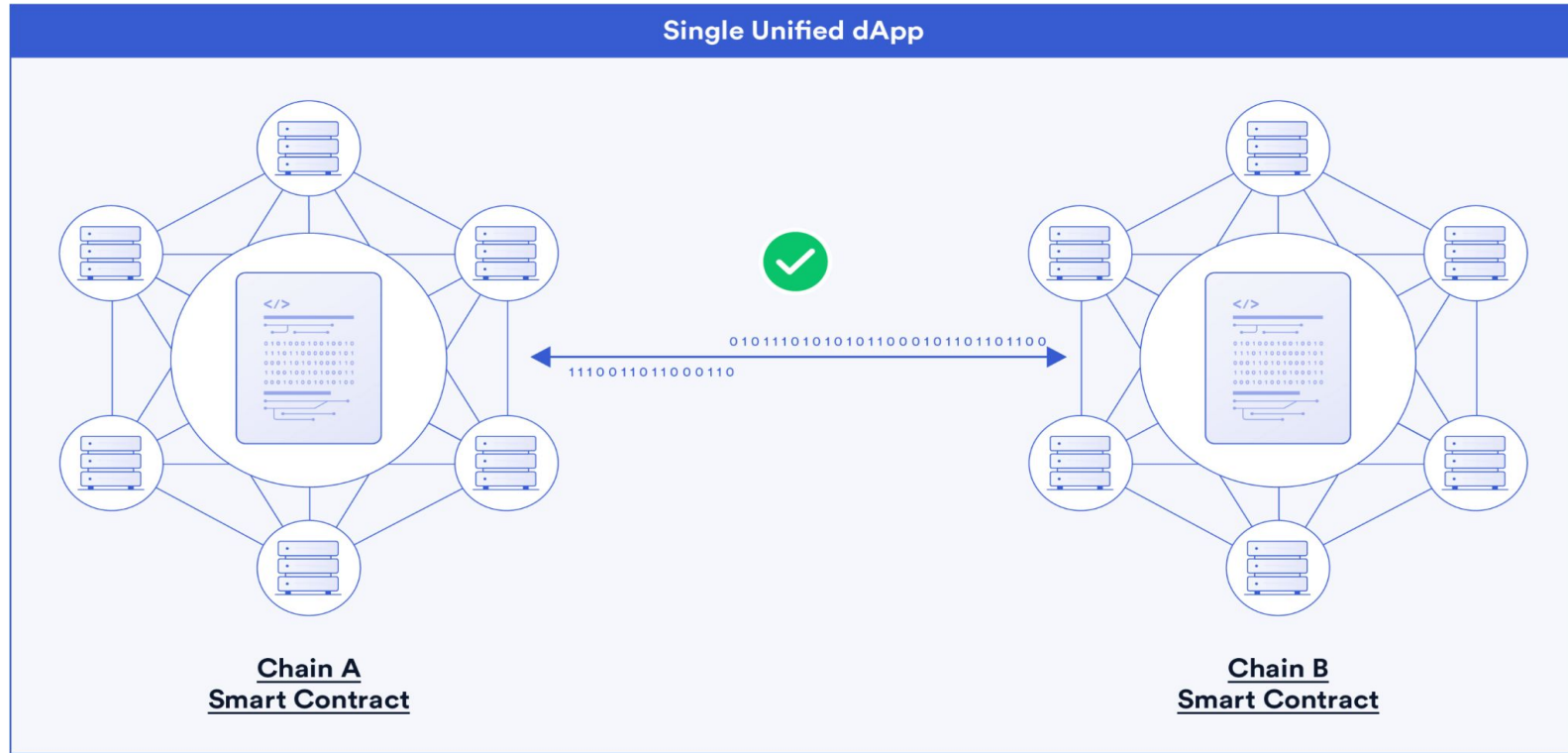


The Chainlink Cross-Chain Interoperability Protocol (CCIP) provides a single simple interface through which dApps and web3 developers can securely meet all their cross-chain needs, including token transfers and arbitrary messaging.

NETWORKS



Cross-Chain Smart Contracts



Core capabilities of CCIP



Arbitrary Messaging

Send arbitrary messages (i.e. bytes) to a receiving smart contract on a different blockchain.



Token Transfers

Transfer tokens to a receiving smart contract or directly to an end-user on a different blockchain.



Programmable Token Transfers

Transfer tokens along with instructions what to do with them to a receiving smart contract on a different blockchain.



CCIP Sender - Source Blockchain

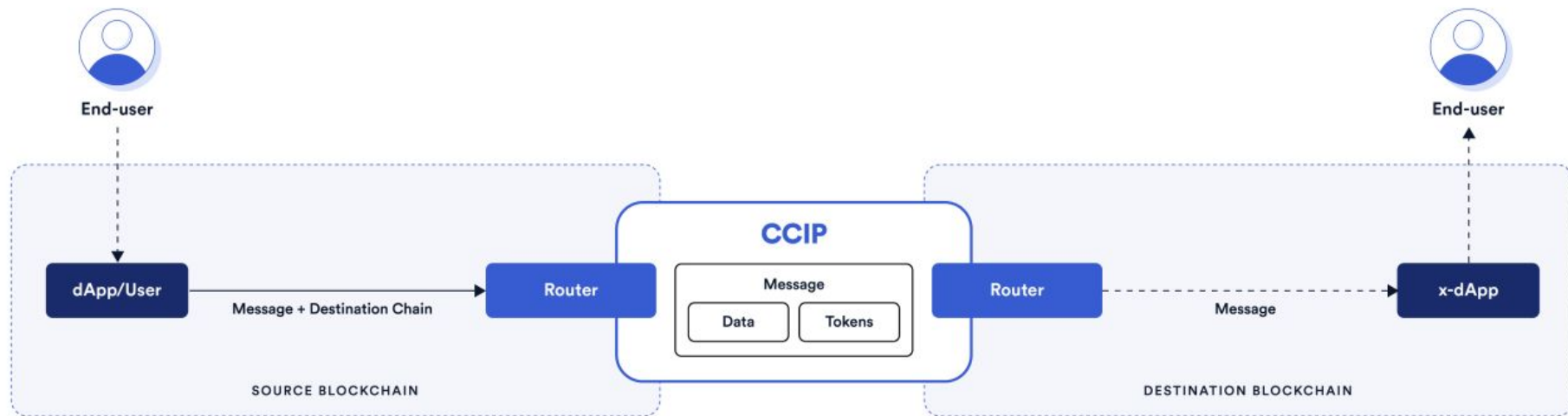
- EOA (Externally Owned Account)
- Smart Contract

CCIP Receiver - Destination Blockchain

- EOA (Externally Owned Account)
- Smart Contract
 - i. Any SC that implements `CCIPReceiver.sol`
 - ii. If no `CCIPReceiver.sol` implementation, only tokens will arrive



Basic Architecture



Developer Interfaces

Source blockchain

```
interface IRouterClient {
    /// @notice Request a CCIP message to be sent to the destination chain
    /// @param destinationChainSelector The destination chain selector
    /// @param message The cross-chain CCIP message including data and/or tokens
    /// @return messageId The message ID
    function ccipSend(
        uint64 destinationChainSelector,
        Client.EVM2AnyMessage calldata message)
        external payable returns (bytes32 messageId);
}
```

```
library Client {
    struct EVM2AnyMessage {
        bytes receiver; // abi.encode(receiver address) for dest EVM chains
        bytes data; // Data payload
        EVMTokensAmount[] tokenAmounts; // Token transfers
        address feeToken; // feeToken address. address(0) means you send msg.value
        bytes extraArgs; // Populate this with _argsToBytes(EVMExtraArgsV1)
    }
    struct EVMTokensAmount {
        address token; // token address on the local chain
        uint256 amount;
    }
    struct EVMExtraArgsV1 {
        uint256 gasLimit; // ATTENTION!!! MAX GAS LIMIT 4M FOR ALPHA TESTING
        bool strict; // See strict sequencing details below.
    }
}
```

Destination blockchain

```
/// @notice Application contracts that intend to receive messages from
/// the router should implement this interface.
interface IAny2EVMMessageReceiver {
    /// @notice Router calls this to deliver a message.
    /// @param message CCIP Message
    /// @dev Note ensure you check the msg.sender is the Router
    function ccipReceive(Client.Any2EVMMessage calldata message) external;
}
```

```
library Client {
    struct Any2EVMMessage {
        bytes32 messageId; // messageId corresponding to ccipSend on source
        uint64 sourceChainSelector; // Source chain selector.
        bytes sender; // abi.decode(sender) if coming from an EVM chain
        bytes data; // payload sent in original message
        EVMTokensAmount[] destTokenAmounts; // Tokens and amounts at destination.
    }
    struct EVMTokensAmount {
        address token; // token address on the local chain
        uint256 amount;
    }
}
```





// Building Cross-Chain with CCIP

CCIP Examples

Some examples of what cross-chain experiences developers can **build with CCIP**



Token Transfers

Transfer tokens across blockchains without having to build your own bridge solution.



Collateral

Deposit collateral on one blockchain and borrow assets on another.



True DeFi Composability

Enable industry wide composability, where all DeFi protocols can be integrated with each other, regardless of chain



NFTs

Give users the ability to mint an NFT on a source blockchain and receive it on a destination blockchain.



Account Abstraction

Build smart contract wallets with native CCIP capabilities to improve the user experience of making cross-chain function calls. Think blockchain abstraction over account abstraction



Blockchain-Agnostic Gaming

Create experiences that enable players to store high-value items on more secure blockchains while playing on more scalable blockchains.



Data Storage and Computation

Enable users to store arbitrary data on a destination chain and execute computations on it using a transaction on a source chain.





Swift

- Chainlink will be used as an enterprise abstraction layer to securely connect the Swift network to the Ethereum Sepolia network, while Chainlink's Cross-Chain Interoperability Protocol (CCIP) will enable complete interoperability between the source and destination blockchain



Synthetix

- Integrated CCIP in the Synthetix V3
- Allows secure cross-chain transfers of sUSD through Synthetix Teleporter



AAVE

- BGD Labs is integrating CCIP to future-proof it's cross-chain governance
- Allow AAVE to launch onto more chains and conduct key operations in a highly secure solution



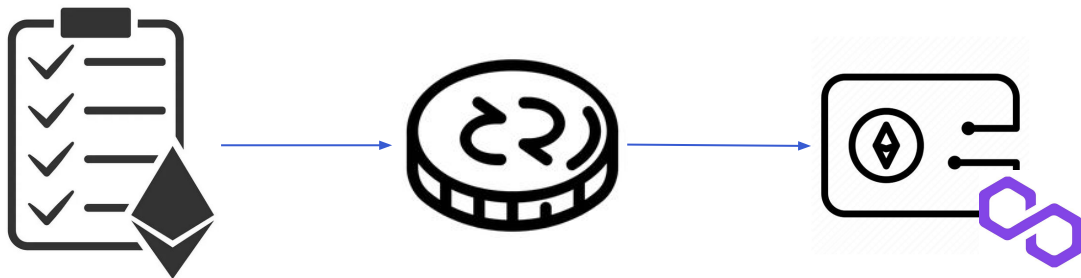


// Building Cross-Chain with CCIP

Token Transfer Exercise

Token Transfer Details

- Send tokens from a smart contract on Ethereum Sepolia to a EOA on Polygon Mumbai
- Using Remix to deploy our smart contract to Sepolia



Official Documentation

<https://docs.chain.link/>



Tutorial Documentation

<https://docs.chain.link/ccip/tutorials/cross-chain-tokens>





// Building Cross-Chain with CCIP

CCIP Architecture in Depth

CCIP Architecture: Token Handling Mechanisms

The CCIP token bridge can support multiple token handling mechanisms at source & destination.

Token handling mechanisms are a key aspect of how token transfers work.

They each have different characteristics with trade-offs for issuers, holders and DeFi applications.

Lock & Mint
(Burn & Unlock)

Typical usage

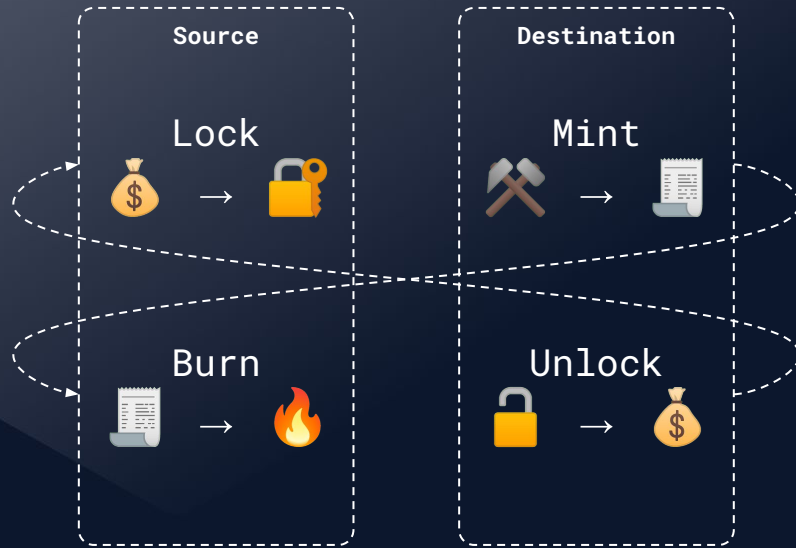
Bridge native tokens to chains where they are non-native by minting a **synthetic/wrapped** version

Burn & Mint

Tokens issued on multiple chains
(e.g. stablecoins, IB tokens, synthetic assets, wrapped tokens)



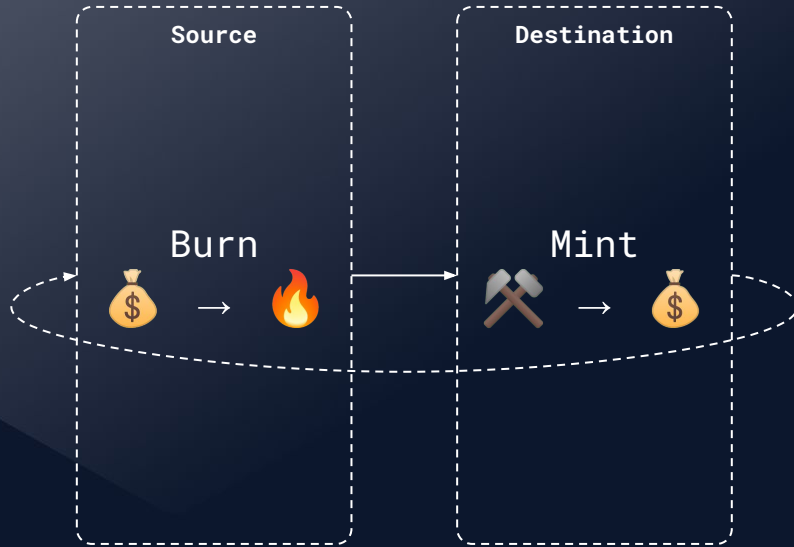
Token Transfers: Lock & Mint: CCIP-LnM



- Tokens are locked on the source chain (in Token Pools) and wrapped/synthetic/derivative tokens that represent the locked tokens, are minted on the destination chain.
- **Use cases:**
 - Tokens minted on a single chain (e.g. LINK)
 - Tokens with encoded constraints (supply/burn/mint)
 - Secure minting function with Proof-of-Reserve



Token Transfers: Burn & Mint: CCIP-BnM



- Tokens are burned on the source chain and minted natively on the destination chain.
- **Use cases:**
 - Tokens that are natively minted on multiple blockchains and have a variable total supply.
 - Examples: stablecoins, synthetic/derivative tokens, wrapped tokens (from Lock & Mint)



Arbitrary Messaging

- Send arbitrary data (encoded as bytes) to a receiving smart contract on a different blockchain.
- You can use arbitrary messaging to trigger an informed action on the receiving smart contract:
 - Rebalancing an index
 - Minting a specific NFT
 - Update the state of a game
 - Calling an arbitrary function with the sent data as custom parameters.
- You can encode multiple instructions in a single message, enabling them to orchestrate complex, multi-step, multi-chain tasks



Programmable Token Transfers



- Simultaneously transfer tokens and arbitrary data (encoded as bytes) within a single transaction.
- This mechanism allows users to transfer tokens and send instructions on what to do with those tokens.
 - For example, a user could transfer tokens to a lending protocol with instructions to leverage those tokens as collateral for a loan, borrowing another asset to be sent back to the user.



// CHAINLINK DEVELOPER EXPERTS

Thank you!

Presentation



Disclaimer: This post is for informational purposes only and contains statements about the future, including anticipated product features, development, and timelines for the rollout of these features. These statements are only predictions and reflect current beliefs and expectations with respect to future events; they are based on assumptions and are subject to risk, uncertainties, and changes at any time. Chainlink CCIP is in the "Early Access" stage of development, which means that Chainlink CCIP currently has functionality which is under development and may be changed in later versions. There can be no assurance that actual results will not differ materially from those expressed in these statements, although we believe them to be based on reasonable assumptions. All statements are valid only as of the date first posted. These statements may not reflect future developments due to user feedback or later events and we may not update this post in response. Chainlink CCIP is a messaging protocol which does not hold or transfer any assets. Please review the Chainlink Terms of Service which provides important information and disclosures.



Billing and Payments

- Fees = Gas Cost + Premium
- Fees can be paid with LINK or native gas (or their wrapped version).
- Smart Execution
 - Charge fees once and at source, incl. gas cost for processing transaction at destination
 - CCIP provides a reliable execution of cross-chain transactions regardless of destination chain gas spikes by automatically bumping gas price if required



