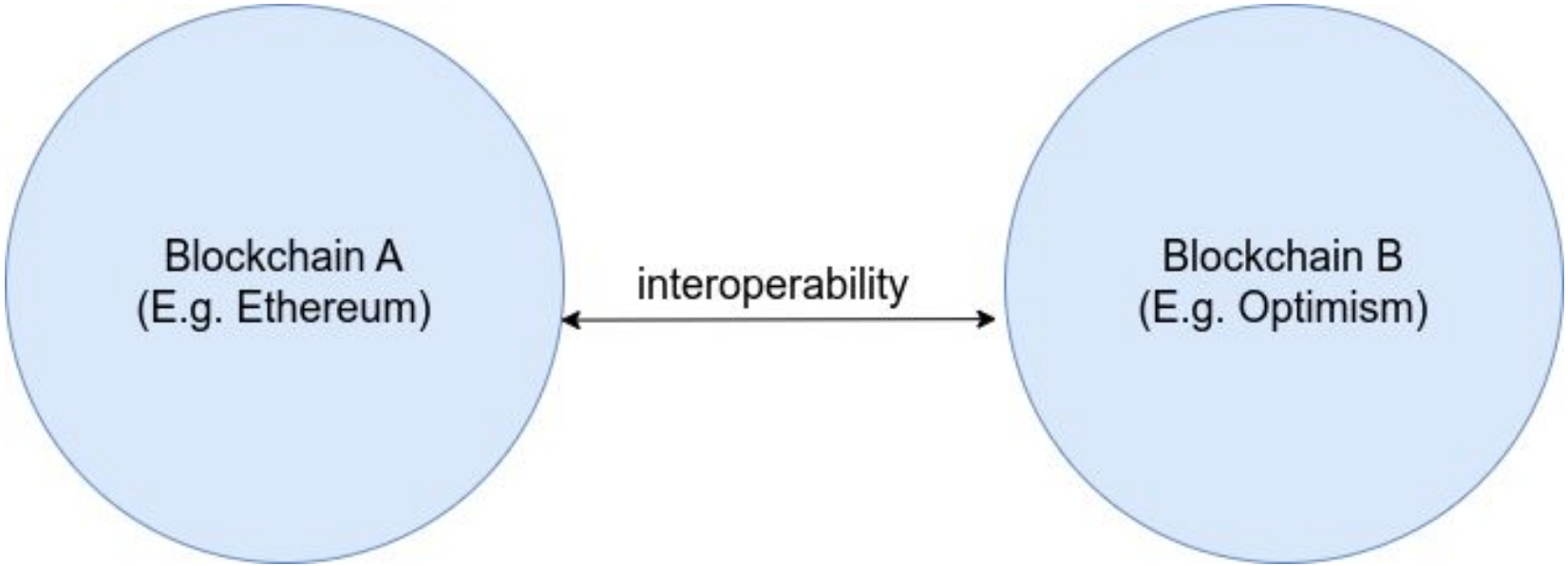
Cross-Chain Stablecoins Made Easy with CMTAT

Ryan
(Taurus/CMTA)

CMTAT

- Security Token Standard framework
- Multi-blockchain support:
  - Public: Ethereum/EVM, Solana (Specification only), Tezos
  - Privacy: Aztec (2025), Zama FHE (March)
- Used by Taurus, UBS (money market fund), Obligate (Debt), Daura (equities), 21.co (past),..
- Stablecoin:
  - Emirati Dirham (AED)

Cross-chain


Blockchain A
(E.g. Ethereum)
interoperability
Blockchain B
(E.g. Optimism)

Cross-chain - Challenge

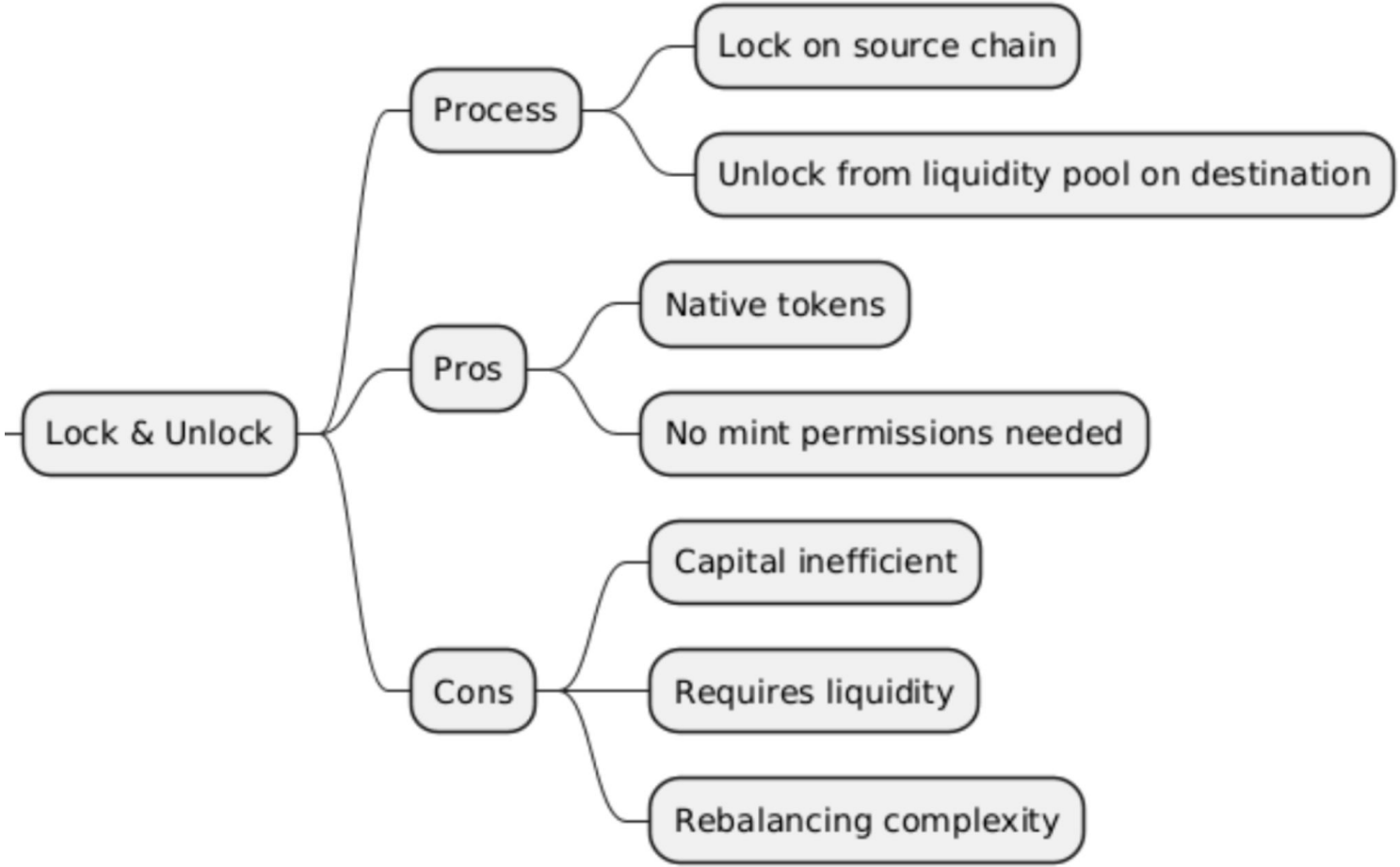
Deploy stablecoin in several multiple blockchains

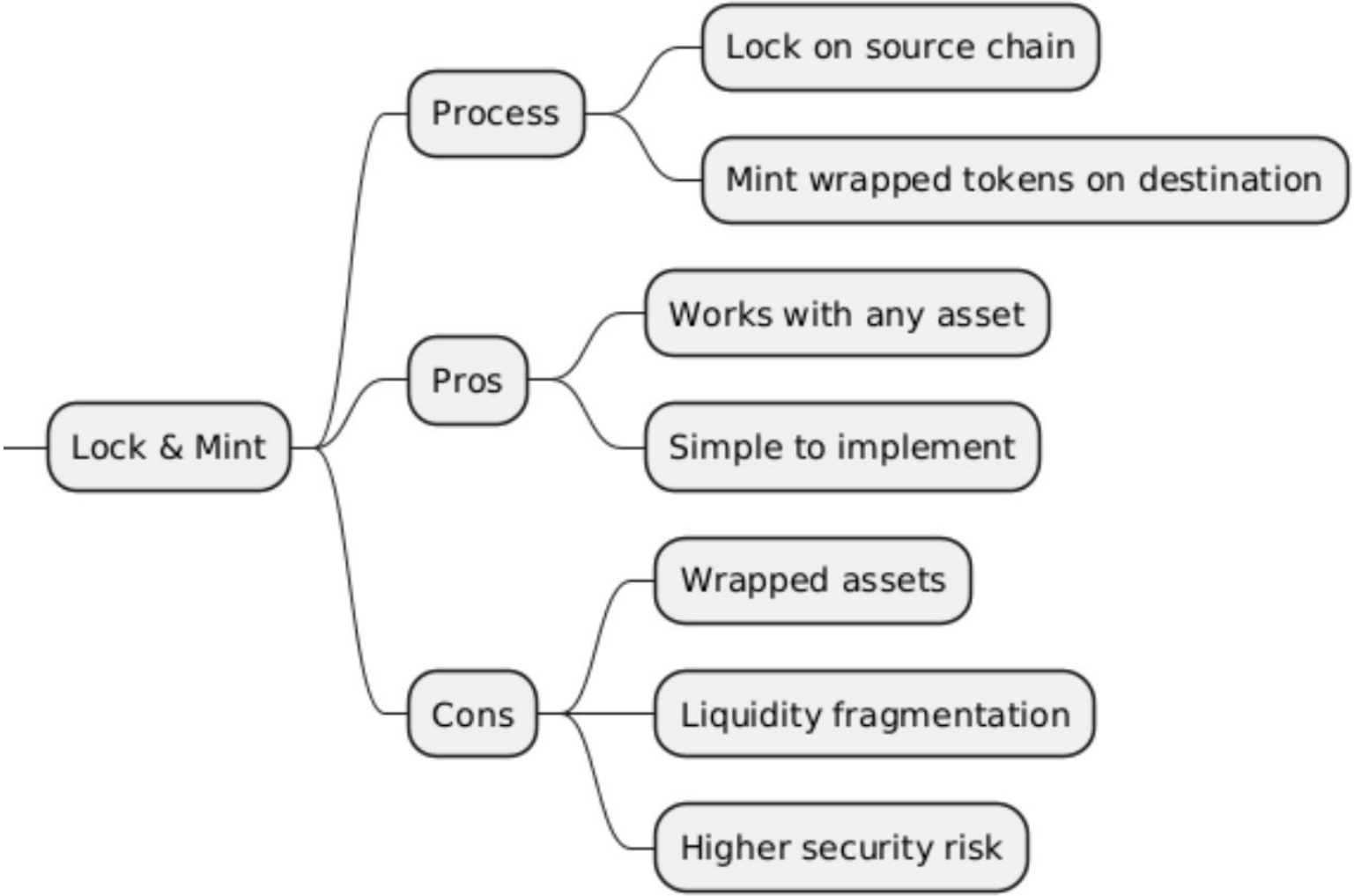
Challenge

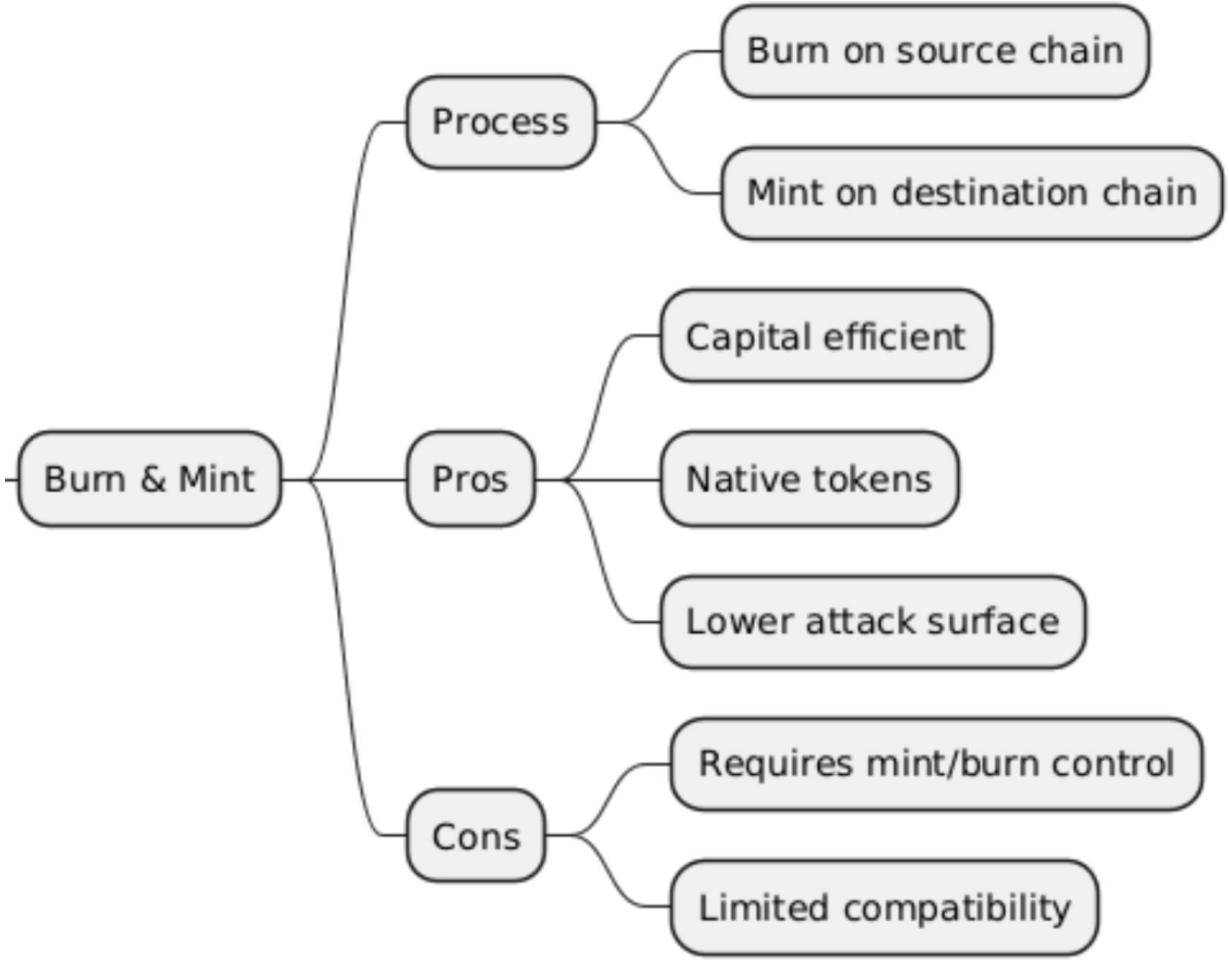
- Maintain same right and properties on each blockchain
  - Same whitelist/blacklist across all blockchains
- Transferability between the different blockchains (focus)
  - Bridge selection and integration
  - Token–bridge compatibility
  - Bridge configuration and support

Three models

- Mint and Burn
- Lock and Unlock
- Lock and Mint


Lock & Unlock
Process
Lock on source chain
Unlock from liquidity pool on destination
Pros
Native tokens
No mint permissions needed
Cons
Capital inefficient
Requires liquidity
Rebalancing complexity


Lock & Mint
Process
Lock on source chain
Mint wrapped tokens on destination
Pros
Works with any asset
Simple to implement
Cons
Wrapped assets
Liquidity fragmentation
Higher security risk


Burn & Mint
Process
Burn on source chain
Mint on destination chain
Pros
Capital efficient
Native tokens
Lower attack surface
Cons
Requires mint/burn control
Limited compatibility

Why burn and mint model

- **Supply integrity:** No duplicated or wrapped supply across chains
- **Legal clarity:** No ambiguity regarding wrapped token rights
- **Security:** Avoids custody risks from locked smart contracts

CMTAT cross-chain functionalities

- ERC-7802: Optimism / Unichain (2025)
- Chainlink CCIP (2025)
- LayerZero Adapters (February)

Next bridge in the roadmap: Axelar

ERC-7802 (Optimism, Unichain)

```
/// @notice Mint tokens through a crosschain transfer.
/// @param _to     Address to mint tokens to.
/// @param _amount Amount of tokens to mint.
function crosschainMint(address _to, uint256 _amount) external;

/// @notice Burn tokens through a crosschain transfer.
/// @param _from   Address to burn tokens from.
/// @param _amount Amount of tokens to burn.
function crosschainBurn(address _from, uint256 _amount) external;
```

```
function sendERC20(
    address _token,
    address _to,
    uint256 _amount,
    uint256 _chainId
)
    external
    returns (bytes32 msgHash_)
{
    if (_to == address(0)) revert ZeroAddress();

    if (!IERC165(_token).supportsInterface(type(IERC7802).interfaceId)) revert InvalidERC7802();

    ISuperchainERC20(_token).crosschainBurn(msg.sender, _amount);

    bytes memory message = abi.encodeCall(this.relayERC20, (_token, msg.sender, _to, _amount));
    msgHash_ = IL2ToL2CrossDomainMessenger(MESSENGER).sendMessage(_chainId, address(this), message);

    emit SendERC20(_token, msg.sender, _to, _amount, _chainId);
}
```

https://github.com/ethereum-optimism/optimism/blob/develop/packages/contracts-bedrock/src/L2/SuperchainTokenBridge.sol

Chainlink CCIP - CMTAT Support

CMTAT & CCIP Integration Overview

- Dedicated functions to set and manage the CCIP admin
- Standard minting and burning mechanisms (ERC-3643)
- Direct compatibility with CCIP

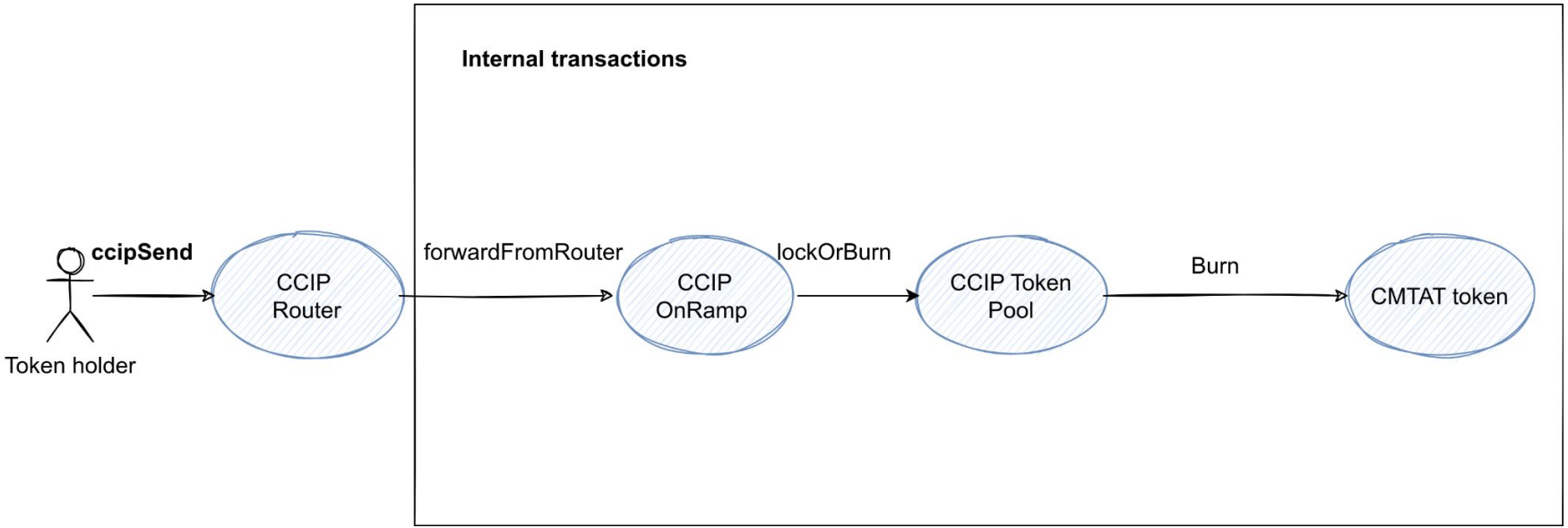
Considerations

- Slight increase in smart contract code size
- You still have to configure CCIP contract

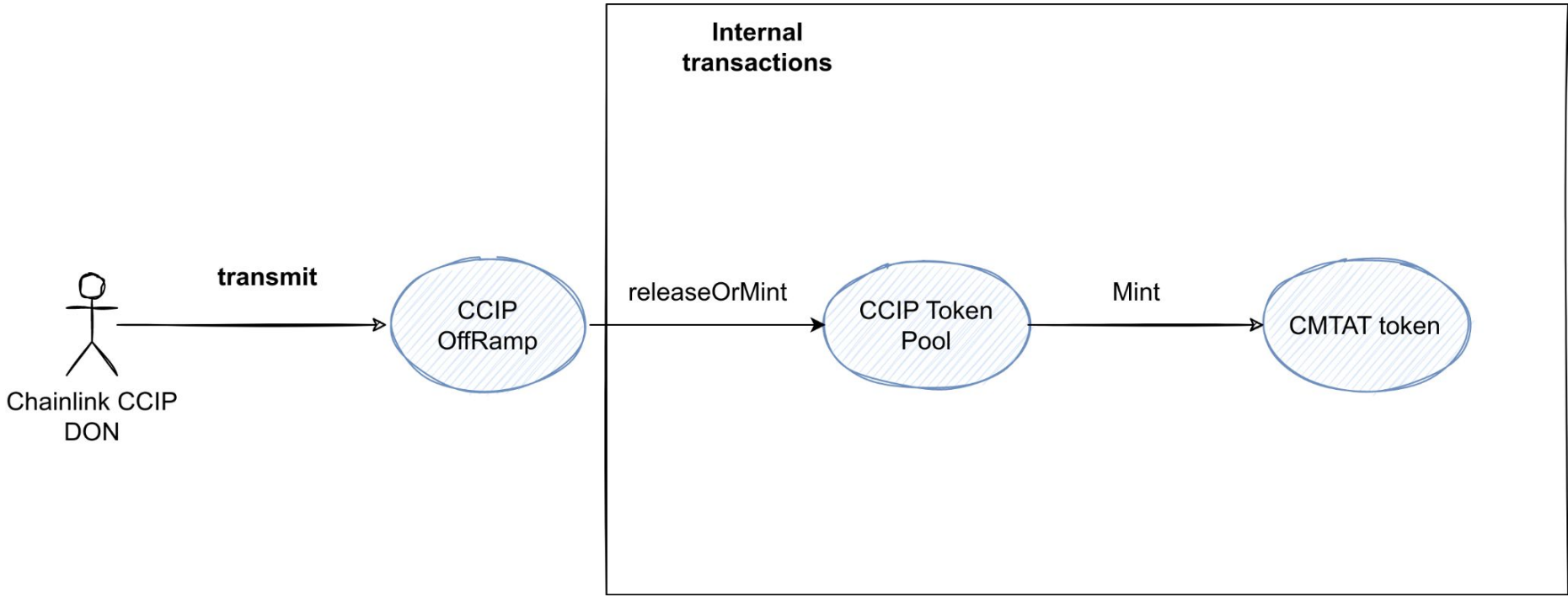
CCIP Transfer Flow

- **CCIP Router** → user entry point
- **CCIP Pool** → authorized contract for minting and burning tokens

CCIP - Source chain



CCIP - Destination chain

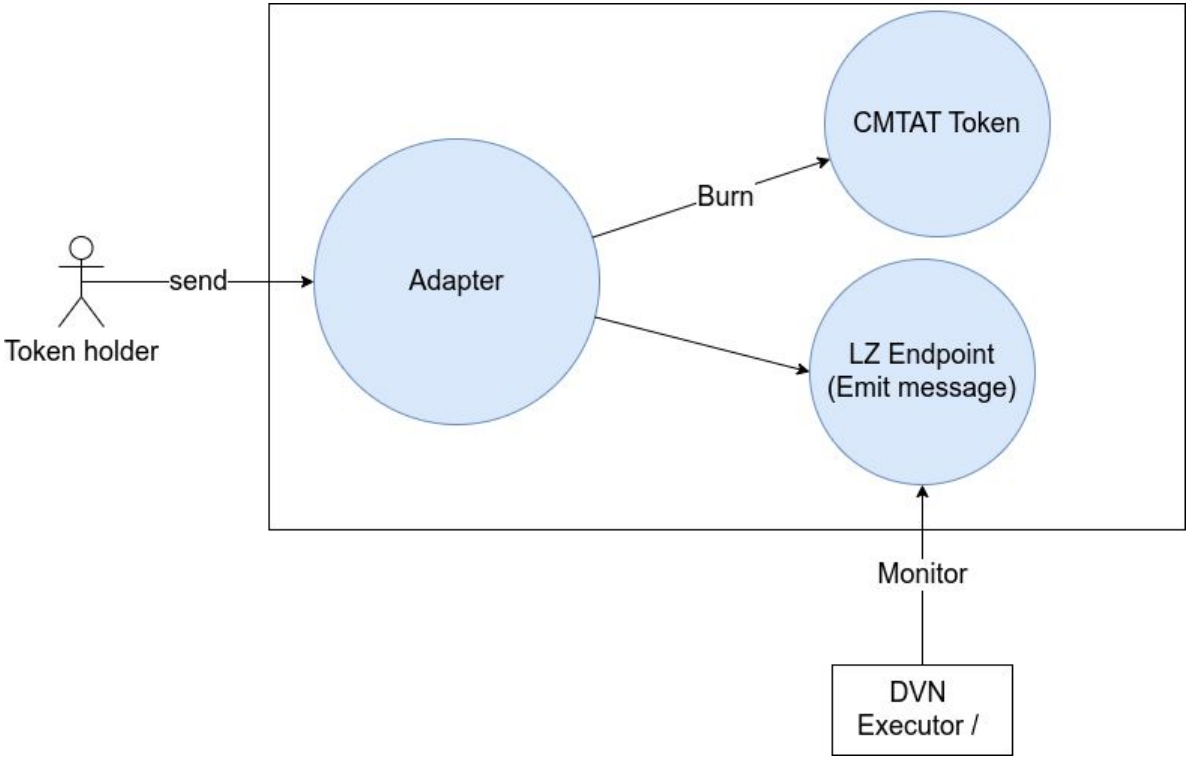


LayerZero - Adapter Contract

External contract

CMTAT offers two adapter contracts:

- ERC-7802 cross-chain minting and burning
- ERC-3643 standard minting and burning



CMTAT GitHub

Linkedin
(personal account)