Summary

In this article, we propose an instant optimistic ERC20 transfer between two EVM chains (i.e., a bridge). Instead of using a two-step mechanism (request/challenge) in most optimistic solutions, where the user has to wait a long challenge time (e.g., Optimistic bridge between Mainnet and a POS chain), the proposed method can instantly complete the operation on the target chain (e.g., minting a wrapped token or withdraw the token from lockbox contract). Further, the method is chain-consensus agnostic, meaning that it can be easy to extend to bridge the assets on multiple EVM chains regardless of their consensus.

A Simplified Setup

- Source chain S with token T
- Destination chain D with wrapped token W
- Users who want to exchange token T on source chain S to token W on dest chain D with 1:1 ratio
- Minter who puts some amount of token W as collateral on dest chain D and mints token W for users

Basic Idea

The basic idea is to allow anyone to become a minter that is able to mint token W infinitely as long as

- The sum of minted value in recent CHALLENGE_PERIOD <= the value of collateral the minter locked on dest chain D / COLLATERAL_RATIO; and
- The miner is not challenged in the recent CHALLENGE_PERIOD.

Example (Happy Path)

Considering transfer USDT from ETH (source) to BSC (dest). Suppose COLLATERAL_RATIO = 2, CHALLENGE_PERIOD = 1 day. A minter locked 100k wrapped USDT as collateral on BSC, which means it could mint up to 50k wrapped USDT on any 1-day window on BSC.

- At time 0, user 1 locks/transfers 20k USDT from ETH, the minter observes the event and mints 20k WUSDT to the user on BSC (the remaining quota at the moment is 30k).
- At time 12h, user 2 locks/transfers 30k USDT from ETH, the minter observes the event and mints 30k WUSDT to the user on BSC (the remaining quota at the moment is 0).
- At time 18h, user 3 locks/transfers 10k USDT from ETH, the minter observes the event, however, it cannot mint any WUSDT since it already minted 50k in the recent 1-day window.
- At time 24.1h, the minter mints 10k USDT to user 3 on BSC (the remaining quota at the moment is 10k).

Example (Challenge Path)

Suppose the PENALTY_RATE = 1

- At time 0, user 1 locks/transfers 20k USDT

from ETH. The minter observes the even, however, it mints 30k WUSDT

, i.e., extra 10k WUSDT

is minted to the user on BSC.

- Within CHALLENGE_PERIOD, Validators capture the malicious mint and challenge the mint on BSC with a majority vote.
- After the challenge succeeded, validators will share PENALTY_RATE * 10k = 10k WUSDT from the minter's collateral as a penalty reward, and burn 10k WUSDT from the minter's collateral to support extra minted 10k WUSDT

. As a result, the minter will have only 100k - 20k = 80k WUSDT as the remaining collateral (and can continue to mint 40k WUSDT in any CHALLENGE_PEROID).

Example (Worst-Case Challenge Path)

Suppose the PENALTY_RATE = 1

- At time 0, the minter maliciously mints 50k WUSDT

to itself on BSC.

- Within CHALLENGE_PERIOD, validators capture the malicious mint and challenge the mint on BSC with a majority vote.
- After the challenge succeeded, validators will share PENALTY_RATE * 50k = 50k WUSDT from the minter's collateral as a penalty reward, and burn 50k WUSDT from the minter's collateral to support maliciously minted 50k WUSDT

. As a result, the minter will have only 100k - 100k = 0 WUSDT as the remaining collateral.

Extensions

Using Native Token Instead of Wrapped Token on Dst. Chain

In the case that the destination chain has the same native token as that of the source chain (e.g., USDT is issued on multiple chains), we could use the native token as the collateral. When transferring the token cross-chain, instead of mining the wrapped token, the minter would just withdraw the token from the contract on dst. chain, where the token may be from:

- native tokens locked locally (and mint/withdraw on another chain)
- collateral, i.e., liquidity provisioning via collateral

.

In the case that the native token on dst. chain is exhausted, a hybrid solution can support both native token and wrapped token on dst. chain:

- If there is sufficient liquidity on dst. chain, only native token will be withdrawn for a cross-chain transfer; or
- If not, the minter may withdraw the native token and then mint the rest via the wrapped token.

In the hybrid solution, the wrapped token can be

- Converted to a native token locally as long as there is sufficient liquidity; or
- Transferred to another chain as an exchange for the native token or wrapped token on another chain following our proposed method.

Other extensions

- Initial Wrapped Token Transferred

. Since there may not have the wrapped tokens for minters' collateral, we can still use the two-step mechanism or majority vote to create these initial wrapped tokens.

- Supporting multiple chains

. The miner can listen to multiple chains and mint as long as there are corresponding lock/transfer events found on multiple chains.

- Validator set changes

. We could use a fixed validator set with reputable identities when starting, but can further use validator set change protocols developed by existing optimistic bridges.

- Large amount transfer

. The two-step request/challenge mechanism can still be used, while a minter can continuously mint until the large transfer is satisifed.

- Avoid gas war.

If multiple minters would like to mint a lock/transfer event on the source chain, the user may assign a preferred minter for the

cross-chain transfer, and only the preferred minter is able to mint in PREFERRED_MINT_PEROID (e.g., 30 mins)

Similarity

- It has some similarities with lending protocols such as Compound/Aave with different “borrowing” constraints.