DOID Bridge: A secure cross-chain communication, identity and asset transfer protocol

DOID Bridge Protocol

The goal of "DOID Bridge" is to provide a technical service for secure cross-chain communication, cross-chain asset transfer, and multi-chain decentralized identity.

In order to achieve this goal, DOID Bridge introduces a secure cross-chain communication protocol DCCP (DOID cross-chain communication and asset transfer protocol) to realize cross-chain communication and ensure that users can protect their assets and complete cross-chain operations quickly.This protocol will combine the advantages of HTLC and MPC and avoid their shortcomings.

In addition, in DOID Bridge, we also want to be able to implement Lock-Mint, Burn-Release, Burn-Mint, and Swap operations. This will provide cross-chain operations in two phases, the first phase will provide Lock-Mint, Burn-Release and Burn-Mint operations, and the second phase will introduce a new cross-chain liquidity pool protocol DCLP (DOID cross-chain liquidity protocol). When the types of assets on the chain are sufficient, liquidity pool will be introduced and Swap operation will be provided. When an asset is swapped from one chain to another, The assets on the source chain will be added to the liquidity pool of the source chain, and the corresponding assets will be obtained from the liquidity pool of the target chain.

In the DCLP protocol, in order to maintain the liquidity of the liquidity pool, liquidity usage fee will be applied as a benefit for the liquidity provider, in addition, a new liquidity balance algorithm will be introduced to protect the assets of the liquidity provider from being stolen or by malicious additional issuance of a certain chain and users are encouraged to balance the liquidity pools between different chains.This will be described in detail in the technical details of DCLP.

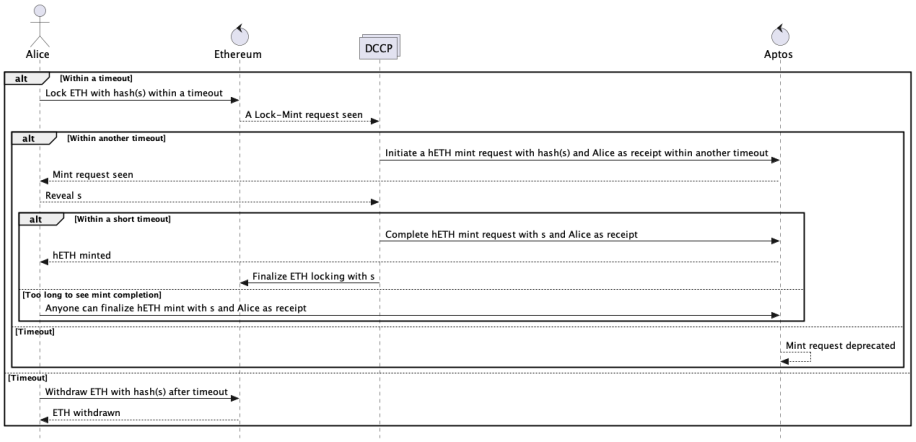
Decentralized identity is another inevitable requirement of multi-chains. On the basis of linking multiple blockchains, DOID Bridge Decentralized Identity introduces a new decentralized identity protocol DOID (Decentralized OpenID) to

meet this requirement. On the premise of protecting user privacy, Decentralized identity information can be realized by zero-knowledge proof.

Function Definitions

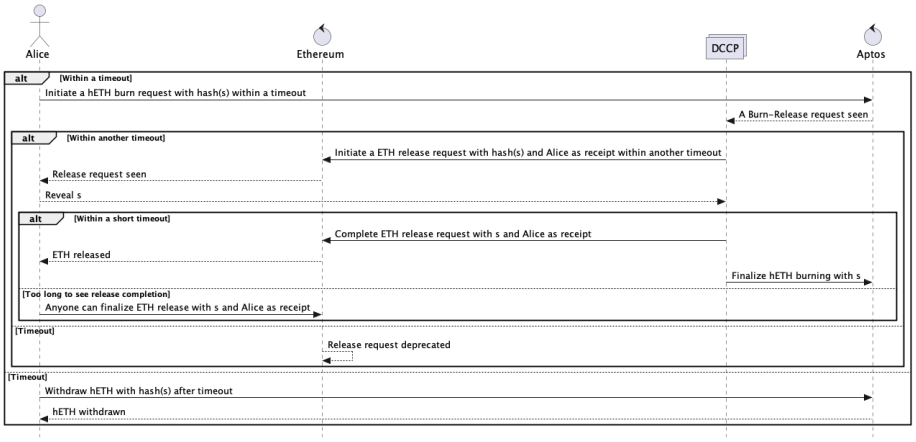
Lock-Mint

Lock-mint refers to the process of mint a new asset on the target chain after a user locks an asset on the source chain. For example, after locking an ETH on Ethereum, a user casts an DCCP-pegged ETH on the Aptos chain, tentatively called hETH, The process is as follows:



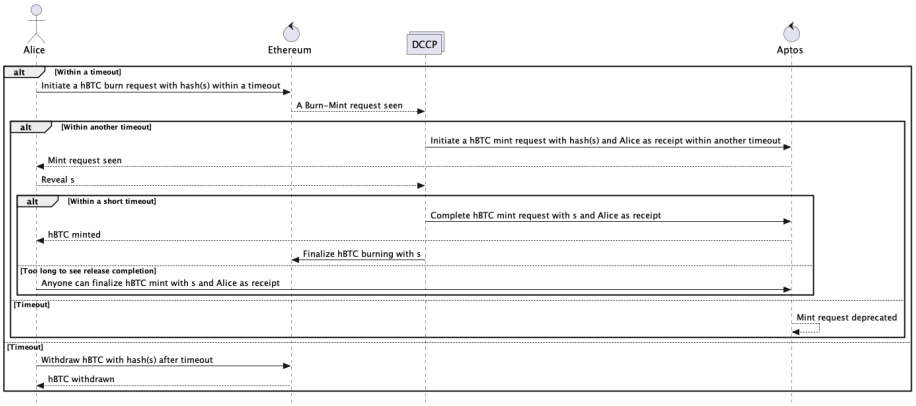
Burn-Release

Burn-Release refers to the process of releasing the source assets on the source chain after the user burns DCCP-pegged assets on the target chain. For example, a user burns hETH on Aptos, and accordingly obtains the released ETH on Ethereum. The process is as follows:



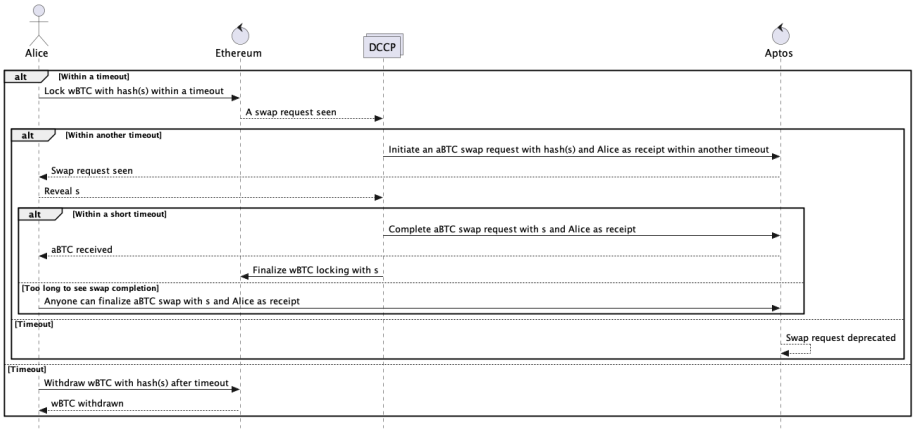
Burn-Mint

Burn-Mint means that after the user burns DCCP-pegged asset on one target chain, the corresponding DCCP-pegged asset is minted on another target chain. For example, a user burns hBTC on Ethereum and mints corresponding hBTC on Aptos, the process is as follows:



Swap

Swap refers to the operation in which users lock an asset on the source chain and exchange it for another homogeneous asset on the target chain. For example, if a user locks wBTC on Ethereum and assumes that there is another BTC anchor asset on Aptos, which is aBTC, the user can redeem the corresponding aBTC on Aptos. In this case, there will be an aBTC liquidity pool on Aptos network as the exchange source. The process is as follows:



Fees, rewards and penalties

User of the DOID Bridge protocol

Users who use the DOID Bridge protocol for asset conversion and identity authentication need to pay a certain amount of protocol usage fees for the DCCP maintainers in addition to the network usage fees of the source chain and target chain. Those who perform Swap operations also need to pay a certain amount of liquidity pool usage fee.

DCCP maintainer

The maintainer of DCCP needs to pledge a certain amount of assets to participate in the maintenance of the DCCP protocol network. As a maintainer, the protocol usage fee of the cross-chain operation amount can be obtained as a profit. This rate can be fixed rate or floating rate decided by community voting.

However, if the operations are not executed in time, other maintainers will compete for benefits. If there is a wrong operation and successfully challenged by other maintainers, it will lose a certain amount of mortgage assets or even all of it.

DCLP Liquidity Provider

DCLP liquidity providers can get a certain percentage of swap operations as profit, and this percentage will fluctuate between 0.01% and 0.4%, depending on the liquidity pool and determined by community voting.

DCLP Liquidity Balancer

DCLP's liquidity pool design will allow for benefits to be gained when balancing liquidity pools across multiple chains.

Thanks to the algorithm design of the DCLP protocol, the cost borne by DCLP users is not obvious, which depends on the balance of the inter-chain liquidity pool. When the imbalance is obvious, the cost will rise and can be used as the benefit of the balancer. More information will be shared in DCLP protocol.

Technical Details

DCCP

Consensus in DCCP is archived with a BFT algorithm, and decides which nodes are selected to participate in the generation of the decentralized key pair and replace it regularly. The generation and signature algorithms of the decentralized key pair are as follows:

Generation of decentralized key pairs

The public key of a decentralized key pair is disclosed and the private key is held in the hands of the participants in a decentralized way. The public key can be verified only after more than a certain number of participants complete the signature. If less than this number of participants, the verifiable signature cannot be obtained. This is often called threshold signature algorithm or threshold signature algorithm, which is an application of zero-knowledge proof.

DCCP uses a similar algorithm to implement decentralized threshold signatures[1] as a valid verification signature for cross-chain operations. In order to form such a decentralized key pair, DCCP needs to reach a consensus on the current network.

The generation process is as follows:

1. Each participant P_i selects a random number $x_i \in_R Z_q$, calculates $y_i = g^{x_i} \in \mathcal{G}$, and calculates $[C_i, D_i] = \mathsf{Com}(y_i)$ as the proof for y_i, and broadcasts C_i
2. Each participant P_i broadcasts
    1. D_i so that any other parties can calculate $y_i = \mathsf{Ver}(C_i, D_i)$
    2. $\alpha_i = E(x_i)$
    3. A zero-knowledge argument Π_i stating the following information:
        1. $\exists \eta \in [-q^3, q^4]$ such that
        2. $g^\eta = y_i$
        3. $D(a_i) = \eta$
3. All parties calculate $\alpha = \oplus_{i=1}^{t+1} \alpha_i$ and $y = \Pi_{i=1}^{t+1} y_i$

At this point, there is a decentralized public key y and the private key is held in the hands of the participants.

[1] Gennaro, R., Goldfeder, S., & Narayanan, A. (2016, June). Threshold-optimal DSA/ECDSA signatures and an application to bitcoin wallet security. In International Conference on Applied Cryptography and Network Security (pp. 156-174). Springer, Cham.

Decentralized threshold signature generation

Like the ordinary threshold signature, the decentralized threshold signature is also a signature of a hash value m, the difference is that the private key in the decentralized key pair used for the private key has at least $\frac{t}{n}$. The process of such a signature algorithm is as follows:

1. Each participant P_i selects a random number $\rho_i \in_R Z_q$, calculates $u_i = E(\rho_i)$ and $v_i = \rho_i \times_E \alpha = E(\rho_i x)$, computes $[C_{1,i}, D_{1,i}] = \mathsf{Com}([u_i, v_i])$ and broadcasts $C_{1,i}$
2. Each participant P_i broadcasts
    1. $D_{1,i}$ so that any other parties can calculate $[u_i, v_i] = \mathsf{Ver}(C_{1,i}, D_{1,i})$
    2. A zero-knowledge argument $\Pi_{(1,i)}$ stating the following information:
        1. $\exists \eta \in [-q^3, q^3]$ such that
        2. $D(u_i) = \eta$
        3. $D(v_i) = \eta D(E(x))$
3. All parties calculate $u = \oplus_{i=1}^{t+1} u_i = E(\rho)$ and $v = \oplus_{i=1}^{t+1} v_i = E(\rho x)$, where $\rho = \Sigma_{i=1}^{t+1} \rho_i$
4. Each participant P_i selects a random number $k_i \in_R Z_q$ and $c_i \in_R [-q^6, q^6]$, calculates $r_i = g^{k_i}$ and $w_i = (k_i \times_E u) +_E E(c_i q) = E(k_i \rho + c_i q)$, computes $[C_{2,i}, D_{2,i}] = \mathsf{Com}(\mathsf{r_i}, \mathsf{w_i})$ and broadcasts C_{2_i}
5. Each participant P_i broadcasts
    1. D_{2_i} so that any other parties can calculate $[r_i, w_i] = \mathsf{Ver}(C_{2,i}, D_{2,i})$
    2. A zero-knowledge argument $\Pi_{(1,i)}$ stating the following information:
        1. $\exists \eta \in [-q^3, q^3]$ such that
        2. $g^\eta = r_i$
        3. $D(w_i) = \eta D(u) \bmod q$
6. All parties calculate $w = \oplus_1^{t+1} w_i = E(k\rho + cq)$ where $k = \Sigma_{i=1}^{t+1} k_i$, $c = \Sigma_{i=1}^{t+1} c_i$. And calculate $R = \Pi_1^{t+1} r_i = g^k$, $r = H'(R) \in Z_q$
7. Decrypt w to learn the value $\eta \in [-q^7, q^7]$ such that $\eta = k\rho \bmod q$ and $\psi = \eta^{-1} \bmod q$
8. All parties calculate

$$\begin{aligned}
\sigma &= \psi \times_E [(m \times_E u) +_E (r \times_E v)] && (1)\\
&= \psi \times_E [E(m\rho) +_E E(r\rho x)] && (2)\\
&= (k^{-1}\rho^{-1}) \times_E [E(\rho(m + xr))] && (3)\\
&= E(k^{-1}(m + xr)) && (4)\\
&= E(s) && (5)
\end{aligned}$$

9. Decrypt σ. Let $s = D(\sigma) \bmod q$, then output (r, s) as the signature of m.

DCLP

Since assets on different chains may have a premium or liquidity shortage, we need to design a liquidity pool algorithm to protect this premium and encourage

rebalancing of liquidity pools across different chains. The algorithms involved here are:

Premium coefficient

Note that the balance of the assets on each chain as $\mathbf{b} = (b_0, b_1, ...)$, the premium coefficient $\mathbf{p} = (p_0, p_1, ...)$, then the adjusted balance is

$$\mathbf{b}' = T(\mathbf{b}, \mathbf{p}) = (\frac{b_0}{p_0}, \frac{b_1}{p_1}...)$$

Liquidity Algorithm

Note D be the sum of all balances, i.e. $D = \Sigma b_i'$, take the algorithm

$$\chi D^{n-1} \Sigma b_i' + \Pi b_i' = \chi D^n + \left(\frac{D}{n}\right)^n$$

If we keep this value constant, the price at the time of swapping will be determined by the coefficient χ. In order to keep the price not too far from 1, χ needs to be a dynamic value, the algorithm is as follows:

$$\chi = \frac{A \Pi b_i'}{(D/n)^n}$$

In the ideal equilibrium state, this value will take a constant A, and will tend to 0 in the unbalanced state. Substitute into the previous formula to get

$$An^n \Sigma b_i' + D = ADn^n + \frac{D^{n+1}}{n^n \Pi b_i'}$$

So, when a particular asset liquidity pool $\mathbf{b}$ changes, recalculate D and keep the equation true when swapping.