charon - a privacy enabled cross-chain amm
https://github.com/themandalore/charon

Charon is a privacy enabled cross-chain automated market maker (PECCAMM). It achieves privacy by breaking the link between deposits on one chain and withdrawals on another. Charon works by having AMMs on multiple chains, with one of the assets on each AMM being the charon dollar (CHD), a free-floating token created by deposits on an alternate chain and then a withdrawal via a zero-knowledge proof[i]. To achieve this cross-chain functionality, charon utilizes an oracle to pass commitments (proof of deposits) between chains.

design

Charon can be broken down into three parts:

- an automated market maker (AMM) structure allowing deposits to other chains
- a bridge which passes deposit commitments to other chains
- a mixer structure that mints CHD on the destination chain

the charon AMM

The AMM structure used by charon is a standard two token AMM with single side withdrawals / LP deposits on the CHD side, very similar in many aspects to uniswap[ii] or balancer[iii] The tokens traded are defined as a "base" token (e.g., native chain asset) and then the charon dollar, a newly minted token created by privacy-enabled deposits from another chain. CHD is a synthetic asset that represents deposits on alternate chains. CHD is burned either when a trade happens for the base asset (CHD is burned rather than added to the pool) or when CHD is deposited as a commitment to the other chain. The only way to add CHD to the AMM is through depositing as an LP, either directly with CHD (single side LP) or with both assets in equal proportions (traditional LP deposit). Since deposits to alternate chains are denominated in constant terms (e.g., 1000 CHD, based on the current AMM price), the newly minted CHD will be roughly stable, tracking external demand for CHD with spreads built in for oracle variability and time constraints due to the lock time of the bridging process.

the bridge

Charon utilizes an oracle to get information about other chains. Either a native oracle (e.g. the Polygon data bridge[iv]) or another decentralized oracle structure (e.g. tellor[v]) can be used to pass deposit information from one chain to another. The oracle mechanism is a key point in security as the oracle alone can deposit valid commitments to mint CHD on the destination chain. To achieve security regarding chain finality and allowing time for oracle disputes, users must wait twelve hours before withdrawing CHD on an alternate chain.

the mixer

Charon achieves mixer functionality due to the anonymous nature of withdrawals on alternate chains. This is similar to a standard mixer (e.g. tornado cash[vi]), where a user deposits on one address but withdraws with another. The unique aspect of charon is that deposits are made on

one chain (commitment generated using destination chain logic (the destination chain is part of the secret)), the commitment is passed to many chains, and then the withdrawal happens on the destination chains. Since the commitment is passed to many chains, assuming other deposits and commitments, anonymity is achieved since there is no way to prove which chain is the destination or which withdrawal on the destination chain corresponds to a given deposit.

asset pricing across chains

In order to allow deposits of multiple different currencies on multiple different chains, the AMM itself is used to price the amount of the deposit (e.g.1000 CHD of ETH). This allows the withdrawal on the destination chain to simply mint 1000 CHD regardless of which currency the deposit came in the form of (e.g. 1000 CHD deposit of ETH on Ethereum or 1000 CHD deposit of MATIC will result in the same withdrawal of 1000 CHD on Optimism (assuming these chains are in the system)). Arbitrage across chains should force the price of CHD to be even on each chain, and if set up properly, should act as a stable medium of exchange across the assets in the system. Due to the long delay between deposit and withdrawal however, parties should be incentivized to keep reserves of CHD and/or leave open withdrawals for quick arbitrage should prices diverge. This arbitrage incentive will help further increase liquidity and anonymity of the system.

flow

There are three sets of users for the charon system: LP's, bridge/mixer users, and traders. The flow for each user is as follows:

bridge/mixer usage

- deposit on chain1 with commitment (created for withdraw on chain3)
- oracle passes commitment to chain2, chain3, chain4 (if anonymity desired)
- user waits to add anonymity then withdraws CHD on chain3 using a proof
- The CHD is then traded as a market order on the chain3 AMM

trader

- same as bridge/mixer user, however, will likely not focus or wait for anonymity
- the arbitrage trader will likely sit in the "to withdraw stage" if incentivized but will hold CHD if not.

LP base asset (e.g. chain1 base asset (e.g. ETH))

- user acquires equal amounts of base asset and CHD (via an exchange or cross-chain deposit)
- deposit chain1 base asset and CHD for pool tokens
- withdraw both assets at pair rate when finished

LP CHD (e.g., on chain2)

- deposit on chain1 with commitment (created for withdraw on chain2)
- oracle passes commitment to chain2, chain3, chain4 (if anonymity desired)
- user waits to add anonymity then withdraws CHD on chain2 using a proof

- CHD are deposited in single side LP on the chain2 AMM

circuit and cryptographic function constructions

The cryptographic functions for off-chain use are implemented in the circomlib[vii] library. Much of the mixer structure and solidity implantation of the Merkle Trees are taken from tornado cash, specifically Poseidon-tornado[viii], a fork with a more efficient hashing algorithm. The Solidity implementation of the Poseidon hasher is created by Iden3[ix]. The SNARK keypair and the Solidity verifier code are generated by the authors using SnarkJS, a tool also made by Iden3.

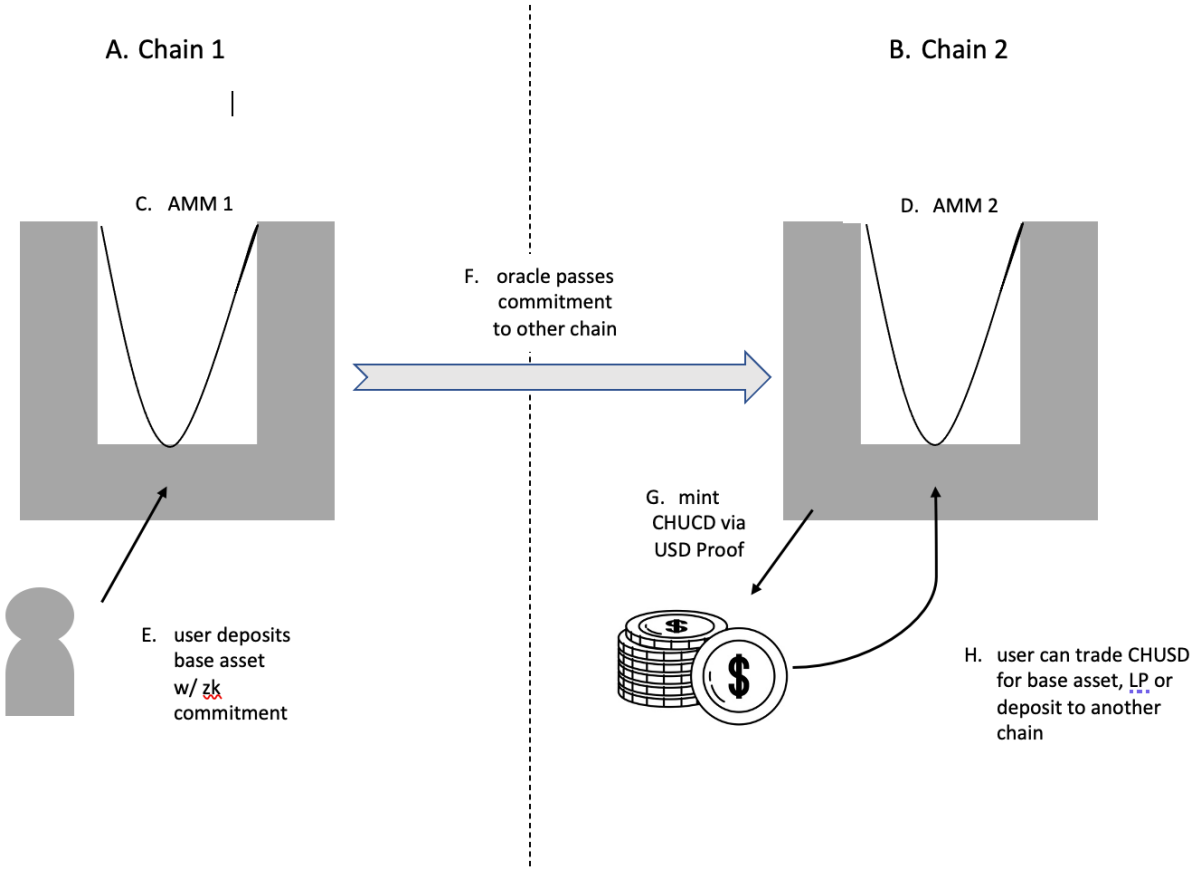
further considerations

oracle methodology

The security of the system relies completely on the oracle mechanism. If the oracle mechanism was broken, false deposits could be created, and the pool could be drained. Due to the potential regulatory issues of charon, the oracle must also be completely trustless and censorship resistant. For this reason, and other aspects, systems such as native data bridges or Tellor will be used as the trustless bridge. For the specific implementation, the use of longer delays for validation can be used to create a safer system. Long delays allow users to see that Tellor (or another oracle) is being attacked, and then can dispute the malicious value to protect the charon deposits. In the worst-case scenario that the oracle is completely compromised (their dispute voting mechanism is overtaken), the delay also enables current exposed users a window to withdraw funds, leaving the attacker with nothing.

---

[i] zero-knowledge proof - https://en.wikipedia.org/wiki/Zero-knowledge_proof
[ii] uniswap - https://uniswap.org/
[iii] balancer - https://github.com/balancer-labs/balancer-core
[iv] polygon data bridge - https://docs.polygon.technology/docs/pos/state-sync/state-sync/
[v] tellor – www.tellor.io
[vi] tornado cash - https://github.com/tornadocash
[vii] circomlib - https://github.com/iden3/circomlib
[viii] poseidon-torndao - https://github.com/ChihChengLiang/poseidon-tornado
[ix] iden3 - https://github.com/iden3


A. Chain 1
B. Chain 2
C. AMM 1
D. AMM 2
F. oracle passes commitment to other chain
G. mint CHUCD via USD Proof
E. user deposits base asset w/ zk commitment
H. user can trade CHUSD for base asset, LP or deposit to another chain