understanding-the-confidential-computing-layer)

Confidential Computing Layer

A technical deep-dive into how Secret can provide confidential computation to dApps on other blockchains by leveraging cross-chain communication and Secret contracts. This section will go over 4 different design options with highlighted use cases. This section should hopefully spark interest into using Secret's Confidential Computing Layer (or "CCL") as a whole and help developers understand the possibilities and limitations of this contract development paradigm.

Understanding the Confidential Computing Layer

The CCL in its most basic form is the idea that one can create products with privacy or additional utility for users who are not on Secret Network itself while using Secret contracts in the backend for the application.

Do you want to tap into Ethereum's liquidity, the polygon NFT scene or just want to add additional encryption-powered use cases to your current techstack? Secret's CCL might be a perfect fit.

The below design paradigms are separated based on the data flow in the app. The simplest apps only take data to and from Secret, while more complex apps either keep all the data separate, or take private data in and return public output.

1. Query Secret (ex. Secret-VRF)

The first design pathway only takes data from a Secret contract but is non-interactive beyond that. This means implementing use cases in this path can be considered relatively simple by leveraging IBC, Axelar GMP or other cross-chain messaging solutions.

In this dApp design scheme, we see Secret as a blackbox that has data accessible useful to any other chain on request. The requested data will have to be transferred publicly. The best example is on-chain randomness viaSecret-VRF .

By leveraging a slightly altered version of Tendermint and the Secure Enclave of every validator, Secret managed to create a truly random number generator on-chain. Every block and every contract call, the number is different and the numbers are verifiable via a callback after the block finalises. A developer on another blockchain can design a lottery completely on-chain by determining the random outcome using Secret-vrf. Simply make a contract on the consumer chain that requests a random-number (over IBC) from a Secret contract. Make sure the lottery entries close before requesting the number as it will arrive over public channels after the block closes on Secret. The number is now used within the contract on the consumer chain to assign the winner; no data has to return to the Secret contract.

Want to leverage Secret-VRF for your application on EVM or IBC? Read more here!

1. Authentication via Secret contracts (ex. web-Auth login, Threshold wallets)

For the second design pathway, data flows towards Secret in an encrypted format with the output merely verifying if the input matches a pre-set pattern or any other defined criteria. Secret contracts here are mainly a decentralized and immutable data-storage mechanism with easily definable access controls.

The best example for this design pathway is web2-esque authentication for application login or other access controlled services. The first dApp leveraging this to its full extent was ALTER, a p2p encrypted communication platform enhanced with Secret smart contracts. The ALTER dApp allows users to store their session encryption key on the Secret Network inside of a Secret contract. Access to this contract and the key is then gated by a Wallet login functionality that ties one's crypto wallet public key to this contract - users can now sign in with their wallet and start messaging. However, this could be taken much further into the web2 realm by allowing sign in via Google web-Auth, password or even biometrics. You can also have usecases where we merely check if a user is who they say they are against identity, credit scoring or other personal information oracles.

The Authentication use case however can be taken much further. A Secret contract could not only check the input against a string it contains, but also influence actions on another chain. The first use case in this region is Unstoppable wallets, a form of on-chain Threshold wallets with unparalleled access control and ownership designed by SCRT labs. For unstoppable wallets, a Secret contract governs one part of a larger private key (MPC) and by programming access control policies into the contract, it will decide to sign or not sign a specific message. Every wallet signer commits the signature for the proposed TX to the Secret contract, the contract checks if the set policies are met and returns a valid signature that can be posted to Ethereum or any other blockchain for that matter. One could even decide to provide the Secret contract with a majority share

of the private key allowing it to process transactions on its own for automated strategies or evenWalletless applications where users can sign TXs with web2 authentication methods instead.

Want to learn more about Unstoppable wallets? - Watchthis talk from the SCRT labs CEO and Secret Network founder Guy Zyskind at Gateway to Cosmos Prague 2023.

1. Sister contracts (ex. Private data for NFTs, verified p2p communication)

In a "sister contract" design scheme the Secret contract and the CCL consumer chain act as two separate entities with which the user interacts separately. This allows developers to work with encrypted output, something none of the other CCL design flows allow.

Sister Contracts is the idea that one can accompany applications on other networks with access controlled data or functions by leveraging a Secret contract. The simplest usecase to explain here is in the realm of NFTs, certificates or other non-fungible personally assigned data objects. It is of paramount importance to many dApps that their usecase is accessible by many, hence why developers often choose for EVM or specifically Ethereum as their home network. With Sister contracts, one can have NFTs, certificates, Invoices or any other item be created on this user-rich chain while simultaneously linking it to a counterpart on Secret network. At any moment in time, the user who owns the item on Ethereum can verify their ownership to the Secret counter-party contract to reveal additional information only visible to them.

The above scenario requires users to interact with two networks to get the full experience of the respective dApp even if cross-chain messaging is already used by the developer. This is because private metadata can only remain private if the user interacts directly with the Secret blockchain without intermediaries like public bridging solutions. Developers can abstract many of these complexities by making intuitive UIs that allow users to use the same wallet for both networks (Metamask for ex.).

Do you want to flare up your NFT with additional private metadata, have access-controlled content for your marketplace or content platform, or store sensitive information related to invoices, IDS or certificates all while keeping your assets or dApps in the most liquid markets? -Then start building a Sister contract with Data privacy for your application now!

1. Private input with delayed public output (ex. private voting, sealed bid auctions, Game-state obscuring)

Lastly, a hybrid approach is possible between the previous design flows, one where there is a private input from any other chain but what is returned is a delayed public output so that all interactions for the user remain to a single chain. As explained before, we can only guarantee output privacy for a user if they are directly interacting with Secret network, but sometimes privacy is not needed for the output - it is only needed for the input.

One such example is private voting. Voting is inherently a public system, the result matters to everyone who voted and maybe beyond that. However, it is important participants can vote at their discretion with full privacy so there is an honest and non-biased outcome. In blockchain, this is not the case: close to all DAOs and chain governance is public by default.

One can design a dApp that allows Ethereum (or any other chain) DAOs to vote privately by voting via a Secret contract instead. Users encrypt their vote and their VP with the TX encryption key of the Secret network contract. The contract remains open for a set amount of time to collect votes after which it tallies results and emits the output. The output can be sent back to the consumer chain to complete the official voting process.

This same concept applies to many different use cases like sealed bid auctions and obscuring the game-state of turn-based games. The first mainnet application on Polygon calledBidshop is using Secret Network in this way to enable a gamified auction platform.To enable these use cases, cross-chain messaging is needed from either Axelar GMP, IBC or other solutions. Bidshop will be allowing integrations for other dApps on their novel trustless message bridge as well, want to get connected? Let us know!

Want to get started with CCL or explore the possibilities but need some extra help?Get in Contact! Previous Gas Evaporation & Tracking Next Fiat/Crypto Onboarding Last updated6 months ago