Secret Contracts introduction

This page describes some of the core information required to get up to speed about the state of Secret Contracts in comparison to public (cosmos) networks.

CosmWasm smart contracts

Smart contracts on Secret Network follow the widely usedCosmWasm framework. The framework is built as a module for the Cosmos SDK and allows developers to build smart contracts withRust which compile toWebAssmebly (Wasm). This type of efficient and fast binary, combined with the power of a low-level typed coding language, makes CosmWasm a secure and efficient way for developers to write smart contracts. Contracts on Secret have a public binary just like any other smart contract network, onlythe contract input, output and state is encrypted.

Secret Contracts slightly differ from any standard CosmWasm implementation as they handle encrypted data, secure enclaves, and the key design around that. Secret currently emulates CosmWasm v1.1.9 and is completely compatible over IBC with other V1.0+ Cosmos Smart contract networks. To learn more about IBC compatible contracts you can readthis section .

Encrypted data storage

Private data (contract input, output and state) on Secret is encrypted using an unforgeable contract encryption key which is unique for every contract. Moreover, the user dedicated state in a contract is only visible to the user themselves. Because of this design, contract migration is a privacy risk. If enabled, a contract admin could upload a malicious contract that reads and prints data entrusted to the old contract.CosmWasm native contract migration is therefore disabled on Secret; workarounds are availablehere . Additionally,iterators and raw queries are not supported as one can not iterate over data stored by a different user as there is no access to their dedicated encryption key.

More information about the differences in implementations for a contract between Vanilla and Secret CosmWasm are listedhere.

Available tools and cool features

Secret contract encrypted input and access-controlled output is very powerful and enables many different usecases, combining this with novel development tools creates a powerful framework for developers to leverage. Below some of these tools and ideas are listed to get your brain going.

- Privacy as a Service
-
    - More utility for other blockchains by leveraging Secret contracts cross-chain. Want to build on polygon, ethereum, Sei, Neutron or Aptos but want additional features only possible with data privacy? You can use Secret contracts cross-chain to keep your application decentralized and offer users new and interesting ways to explore your dApp.
- Storage of private/encryption key (fragments) - One can use Secret contracts to store very sensitive info to enable cool usecases like access-controlled content or even cross-chain account ownership for DAOs and unstoppable threshold wallets.
- on-chain randomness
- (secret-vrf) and hidden (game) state - Games on Secret benefit of true on-chain randomness that can remain hiddin inside the contract forever if so desired. Together with a hidden gamestate this brings a whole new dimension and fairness to turn-based games for Web3 like Poker or Mario party and more clarity and ownership to casino games and lootboxes.
- Frontrunning resistant and privatized DeFi - because of an encrypted mempool secret is frontrunning resistant by default. Order size - unknown, Placed bids - unknown, liquidation points - unknown and many other features from orderbooks to dark pools have these privacy features as well.
- Secret.js
-
    - a Javascript library for interacting with the Secret network, Secret contracts, providing encrypted input and creating viewing keys/permits for encrypted output. The library can also be considered as a more complete Cosm.js and can be used on every other cosmos chain. - Alternatives inPython
- ,Kotlin
- ,.Net
- and more exist as well.
- Polar
- ,Fadroma
- ,Secret-toolkit
- (CW-plus) andSecret IDE
-
    - Secret has dedicated tooling sets built by external teams and core development that abstract boilerplate code (for DeFi), help with testing scenarios, simplify contract development nightmares like storage and type handling and

even catch errors as you work in a dedicated IDE.
- IBC, Interchain accounts/Queries/contracts, Axelar GMP, Bidshop messaging and more - Want to build your application crosschain? Secret has many options for both Message and Asset bridging protocols with the most renowned being IBC (current version IBC-go V4).
-

Trusted vs untrusted data and Contract verification

The privacy of Secret Network is reliant on secure enclaves, which impacts how the network performs but also how one should design their contracts to enhance privacy. Firstly, it is important to note that only data originating from inside the enclave is verifiable and any data coming from external resources has to be verified before relying on it for execution. This means data coming from contract callbacks is to be trusted, but the sender address is not. If I allow the sender to create an item on-chain, then only the sender should be able to remove that item. Verifying that it is indeed the sender (data owner) that is requesting this change is done by "input data verification". Explanation of the various network level integrated methods is availablehere.

In practice a Secret contract developer does not have to think too much about this as its abstracted away in the data verification process in the contract module of the network.

Replay/Side-chain attacks - privacy design

When designing contracts for Secret network, one should take into account the usage of so-called replay attacks and othertheoretical attack vectors on secure enclave derived privacy. An example of this is the abundant usage of the Contract hash in Secret contract development. The usage of a contract hash is required to link the input to the specific contract being called. Otherwise one could make a forked chain and replace the contract with one that reads and prints the input parameters. Although encryption for Secret contracts is handled by the protocol, developers should still be mindful of their contract design to optimize for these privacy aspects.

One example of such a privacy optimized design would be the use of the Gas evaporation function as added in the v1.9 upgrade and documented here (link coming soon).

Contract execution result

There are various types of outputs that can be expected, including:

- An updated contract state (i.e., the user's data should update the state or be stored for future computations)
- A computation result encrypted for the transaction sender (i.e., a result should be returned privately to the sender)
- Callbacks to other contracts (i.e., a contract is called conditional on the outcome of a Secret Contract function)
- Send messages to other modules (i.e., for sending value transfer messages that depend on the outcome of a computation).
- Reply - Send more detailed error logs of various submessages as explained here.
-

Want to dive deeper into the CosmWasm framework? - You can read more about it's design in theCosmWasm book.

Summary

- Secret Contracts have private input, output and state. This means all data on-chain is stored as cyphertext and is only accessible by the owner (either the contract or the user). More details are available in theencryption specs
- .
- The binary of any contract is public, meaning the execution logic is verifiable just like any other smart contract blockchain.
- Secret smart contracts are IBC compatible (CosmWasm v1.0+), allowing for cross-chain logic with other IBC-connected chains.
- from v1.11 Secret now supports migrate-able contracts just like vanilla CW, it does not support iterators due to contract privacy design. Workarounds are highlighted understorage
- .
- When designing Secret contracts, the verification of input-data is key to ensure the secure execution of transactions as not all blockchain data is exposed automatically to the enclave.
- Developers need to be mindful of data leakage via replay attacks, improper padding, and more so that the privacy for users of their contracts is optimized.
-

Last updated7 months ago On this page *CosmWasm smart contracts * Encrypted data storage * Available tools and cool features * Trusted vs untrusted data and Contract verification* Replay/Side-chain attacks - privacy design * Contract execution result

Was this helpful? Edit on GitHub Export as PDF