Trusted and untrusted data

This page explains the concept of input data verification, a required mechanism to ensure reliable privacy for secret contracts.

A more in-depth explanation of input-data verification and the Replay attack vectors that come with are displayed in the "potential attacks " section of the Encryption specs. As Secret runs the compute module from within the enclave of the validator node it handles 2 types of data, data coming from within the enclave or outside of it. This data can then be separated as trusted (in-enclave) and un-trusted (outside enclave) meaning you can only rely on the trusted to be true in all cases, the untrusted data can be forged either with replay attacks on forked chains or state modification attacks.

The amount of trusted data options can be increased by additional verification mechanisms.

Input data verification

During execution, some contracts may want to use "external-data", meaning data generated outside of the enclave and sent into the enclave, such as the tx sender address, the funds sent with the tx, block height, etc...

As these parameters are sent to the enclave, they can theoretically be tampered with, and an attacker might send false data; making relying on this dataconceivably risky .

For example, let's say we are implementing an admin interface for a contract, i.e. functionality only for a predefined address. We want to ensure theenv.message.sender parameter provided during contract execution is legitimate, so we will confirm thatenv.message.sender == predefined_address . If this condition is met we can provide admin functionality. If theenv.message.sender parameter can be tampered with — we effectively can't rely on it and cannot implement the admin interface.

In-enclave light client - fork protection

This verification is done by a light-client implemented inside of the enclave which validates all compute transactions and their order. It matches the set of transactions to be executed with the data hash field of the block at the current height. This means we can trust contract executions (Attention: not other modules like Bank), User address, inputs and sent funds are all part of the signed and validated inputs.

The light client does not validate internal contract calls - when your contract calls another during runtime - but these are signed in such a way that it can only be generated by another contract inside a genuine enclave. This can be considered trusted execution as well.

Merkle proofs (coming soon) - State modification attacks

Looking at state modification attacks - these refer to attacks that attempt to modify the internal state of the chain to create behaviour that allows to compromise confidential data. This can be done by changing the internal state values of the chain (even encrypted values can be replaced with other valid encrypted values) and executing the real transactions to see what happens.

For example, imagine a contract that has the following logic:

1. Receive transaction from user with some amount of SCRT
2. Validate that payment is exactly 1000 SCRT
3. If payment is correct, reveal secret value
4.

An attacker can use a modified chain to attack this logic in the following way:

1. Send a transaction on the real chain with 1000 SCRT from an address that has 1 SCRT
2. Wait for the transaction to fail on mainnet
3. On a modified chain fund the attacker's address with 1000 SCRT
4. Replay the real block from mainnet
5. Reveal the secret value
6.

This scenario illustrates how payment systems need to consider these attack values.

Protecting against modified state is being actively developed, and will be solved by validating that values that are read from the chain are part of the merkle tree of the chain itself -Coming end of 2023

Summary of trusted vs untrusted data - from a contract perspective

Trusted = No means this data can easily be forged. An attacker can take its node offline and replay old inputs. This data that isTrusted = No by itself cannot be trusted in order to reveal secrets. This is more applicable toinit andhandle , but know that

an attacker can replay the inputmsg to its offline node.

Althoughquery cannot change the contract's state and the attacker cannot decrypt the query output, the attacker might be able to deduce private information by monitoring output sizes at different times. Seedifferences in output return values size to learn more about this kind of attack and how to mitigate it.

Type Trusted? Note Storage No Can be rolled back - merkle proofs will make it trusted (end 2023) Env yes Protected against fork attack Sender yes verified - fork protected Inputs from sender yes verified - fork protected Funds (input from non-compute module) no Fork attack can replay failed Tx from mainnet Contract callback input yes anything that originates from within the enclave is trusted

Last updated7 months ago On this page *Input data verification * Summary of trusted vs untrusted data - from a contract perspective

Was this helpful? Edit on GitHub Export as PDF