Kettle

Table of Contents

Overview

This document provides the technical specification for the SUAVE Kettle, one of the primary protocol actors in the SUAVE protocol. The Kettle contains all necessary components to accept, process, and route confidential compute requests and results. Below, the expected behavior of an "honest Kettle" for the Rigil testnet version of the SUAVE protocol is outlined.

Prerequisites

All terminology, functions, and protocol mechanics defined in SUAVE chain , Confidential Data Store , and MEVM are prerequisites for this document and used throughout. Please see the Rigil overview before continuing and use it as a reference.

Kettle Responsibilities

A Kettle has several primary responsibilities within the SUAVE network:

1. Handling Confidential Compute Requests:
2.
   - ProcessConfidentialComputeRequest
3.
   - received from users or other network nodes.
4.
   - Execute transactions in confidential mode, providing access to the usual confidential APIs.
5.
   - Create aSuaveTransaction
6.

    - using the confidential computation request and result.
7.
    - Sign and submit the transaction into the SUAVE mempool.
8. Maintaining the Confidential Data Store:
9.
    - Store and retrieve confidential data used in all SUAPPs.
10.
    - Restrict access to stored data based on the allowed peekers and stores of each bid.
11.
    - Ensure synchronization of confidential data across the network.
12. Propagating Confidential Compute Requests and Results:
13.
    - Forward bundles and blocks emitted from confidential computation to block builders and relays.
14.
    - Broadcast the results of confidential computations to the appropriate parties within the SUAVE network.
15.
    - Ensure the propagation of results adheres to the confidentiality and privacy requirements of the SUAVE protocol.

Becoming a Kettle

Currently, Kettles are not permissioned, but participation in the confidential data store synchronization protocol is. To join the network, you must run the entire Kettle software stack and connect to the existing network via bootnode or known peer. Eventually, Kettles will be associated with an onchain registry.

Kettle Identification

A unique identifier in the form of an Ethereum-compatible public key must be generated for each Kettle to ensure proper tracking and verification of Confidential Computation, as well as responsibility assignment within the network.

Kettle Architecture

SUAVE Kettles house all components necessary to perform confidential computation and result routing. Below is a high-level architectural diagram followed by descriptions of the main components.

RPC

This component is similar to RPCs in any Ethereum or blockchain client and is responsible for mapping user's requests to their appropriate functionality.

SUAVE PoA Chain

The SUAVE chain maintains consensus about smart contract code for SUAPPs. Additionally, it can also be used to store and broadcast data for better censorship guarantees.

Kettles must keep a lively copy of the SUAVE chain state to process Confidential Compute Requests, but these liveliness requirements are currently not defined in the Rigil Kettle.

For more details see the SUAVE chain spec.

MEVM

The MEVM is a new interpreter and a set of precompiles to interact with APIs for the Confidential Data Store and domain-specific services.

For more details see the MEVM spec.

Confidential Data Store

The Confidential Store is a secure and privacy-focused storage system, exposing a key-value store for safeguarding confidential compute and orderflow-related data. Only those with appropriate permissions (peekers) can access the stored data, thus ensuring privacy and control.

For more details, see the Confidential Data Store spec.

Domain-Specific Services

Domain-Specific Services allow Kettles to scale horizontally by hosting nodes for other domains in separate processes that

can be queried at confidential compute time. Kettles currently are not responsible for publicly committing to their domain-specific services, so support for a specific domain is done on Kettle-by-Kettle basis. If a Kettle attempts to process your confidential compute request for a domain it does not support, it will simply return a failure on the computation and will not propagate the failure to the rest of the network so other Kettles can still attempt to process.

For more details on supporting the needed APIs to enable a domain, see below SUAVE Execution API spec.

Containers

The core data types used inside the Kettle.

Confidential Compute Record

This type serves as a record of computation. It's part of both the Confidential Compute Request and Suave Transaction .

type ConfidentialComputeRecord struct

{ Nonce uint64 GasPrice * big . Int Gas uint64 To * common . Address Value * big . Int Data [ ] byte

KettleAddress common . Address ConfidentialInputsHash common . Hash

ChainID * big . Int V , R , S * big . Int }

Confidential Compute Request

This type enables users to request the MEVM to compute over their data via theeth_sendRawTransaction method. After processing, the request'sConfidentialComputeRecord is embedded intoSuaveTransaction.ConfidentialComputeRequest and serves as an onchain record of computation.

type ConfidentialComputeRequest struct

{ ConfidentialComputeRecord ConfidentialInputs [ ] byte } A Kettle's signature is used as the integrity guarantee for the computation's results. Eventually, this can include arbitrary proofs such as zero-knowledge proofs.

Suave Transaction

The final home of compute results and intentionally leaked data from confidential compute requests is a SUAVE transaction; see SUAVE chain specs for more details

type SuaveTransaction struct

{ ConfidentialComputeRequest ConfidentialComputeRecord ConfidentialComputeResult [ ] byte

// Kettle's signature ChainID * big . Int V * big . Int R * big . Int S * big . Int }

Honest Kettle

At present, the protocol relies on the honesty of Kettles, akin to the reliance on honest block builders and relays. An honest Kettle performs the duties above of:

- Handling Confidential Compute Requests
- Maintaining Confidential Data Privacy
- Maintaining the Confidential Data Store
- Propagating Confidential Compute Results

On the Rigil testnet, Kettles do not live inside Trusted Execution Environments. Because of this, a malicious Kettle could alter its source code to censor Confidential Compute Requests and their results.

Confidential Computation

To successfully process a request for confidential computation, Kettles must engage the Confidential Compute Process.

Confidential Compute Process

Confidential compute is defined by the use of an offchain function call , signified in solidity via theview modifier.

These view functions are used with plaintext access to decrypted data to perform confidential computation on the decrypted data. Results and a signature of integrity are propagated along with the initial request. The confidential data is propagated

separately via the confidential data store, but other Kettles do not need the underlying data as they will trust any valid signature attesting to the computation's integrity.

This sequence diagram represents the interactions between various components when processing a Confidential Compute Request (CCR). Here's a description of the flow depicted in the diagram:

- Starting Point
- : The process initiates with a "Confidential Compute Request" directed towards the JsonRPC.
- MEVM Execution
- : Upon receiving the request, the Json RPC triggers the MEVM (Modified Ethereum Virtual Machine) to run. MEVM execution can use multiple APIs depending on the context, with two possible paths:
-
    - Request to an External Domain: The MEVM can make API requests to external domains via Domain-Specific Services.
-
    - Request to the Confidential Datastore: The MEVM can make API requests directly to the "Confidential Datastore" to fetch or store data.
- Suave Chain Interaction
- : Eventually, after processing the request, depending on the SUAPP, the MEVM takes the results and sends a SUAVE transaction.
- Transaction Hash Output
- : Following the Suave transaction, a transaction hash is produced and returned to the request's originator, precisely like in the Ethereum process.

Suave JSON-RPC

SUAVE JSON-RPC can be seen as a super set of Ethereum JSON-RPC. This means the Ethereum JSON-RPC standard remains the same when interacting with the SUAVE chain. Some methods in theeth_ namespace are overloaded to support the confidential compute requests.

eth_sendRawTransaction

Creates new message call transaction or a contract creation for a signedConfidentialComputeRequest .

eth_call

Executes a new message call immediately without creating a transaction on the blockchain. It follows the same format as the defaulteth_call with two extra parameters:

Parameters

- IsConfidential
- : Set to true to execute as a confidential request and access theMEVM
- methods.
- ExecutionAddress
- :address
-
    - (optional) The execution address that performs the execution.

eth_executionAddress

Returns the list of available addresses in the Kettle to execute a confidential compute request. Kettle owners can register multiple addresses to sign transactions with from their Kettle. Users can pick which address the Kettle must use to sign their transactions. Edit this page Previous Rigil Testnet Next Suave Chain