
slug: mevm-suave-centauri-and-beyond title: The MEVM, SUAVE Centauri, and Beyond authors: [robert] tags: [suave, sgx, orderflow auction, building, roadmap] image: /img/posts/suave/centauri-cover.png hide_table_of_contents: false description: In this post we preview the upcoming SUAVE Centauri release and introduce the MEVM. forum_link: https://collective.flashbots.net/t/mevm-suave-centauri-and-beyond/2007

In this post, we preview the *SUAVE Centauri* release defined in the November 2022 roadmap in the [Future of MEV](#). *SUAVE Centauri* consists of two parts: A privacy-aware [orderflow auction](#) (“OFA”), which we released with [mev-share's beta](#); and a SUAVE Devnet. This post aims to explain, at a high level, several key ideas that will help understand SUAVE and Flashbots more broadly. Moreover, we invite the community to experiment alongside us with the upcoming devnet launch detailed in the post.



Making SUAVE Programmable

We are excited to introduce the **MEVM**, a powerful modification of the EVM with new precompiles for MEV use cases. Using the MEVM, developers can program MEV applications as smart contracts within an expressive, familiar, and flexible programming environment - just like the normal EVM. The ambition of the MEVM is to offer every primitive of the MEV supply chain as a precompile, allowing any centralized MEV infrastructure today to be transformed into a smart contract on a decentralized blockchain. Below we will briefly review the MEVM and how it supports the goals of SUAVE before walking through a code example along with the reference architecture for SUAVE.

The MEVM helps achieve several of SUAVE's goals. Drastically lowering the barriers to creating new MEV applications maximizes **competition** for different mechanisms. For example, we hope to see a proliferation of different block building (e.g. [PROF](#)) or orderflow auction mechanisms (e.g., MEVBlocker) on SUAVE. This open field for innovation will help drive better user outcomes and blocks for proposers. The MEVM is maximally **expressive** in that it allows for using every primitive in the MEV supply chain in smart contracts. Lastly, it helps to **decentralize** the MEV supply chain by enabling centralized infrastructure (builders, relays, centralized RFQ routing, etc.) to be programmed as smart contracts on a decentralized blockchain.

Example of writing MEVM smart contracts

Block builders today run off-chain infrastructure that accepts bundles of transactions from searchers and individual transactions from users. From these inputs, the builder tries to find the ordering of transactions in a block that optimizes the block's value. The most simple example is the following algorithm: 1. Simulate all transactions or bundles of transactions to derive their *effective gas price* (defined as total payment / total gas used). 2. Add the highest gas price transaction or bundle first. 3. Try adding the next highest gas price transaction and see if there are any conflicts. If there are, discard the transaction or bundle. If there are not, add the transaction or bundle to the pending block. 4. Repeat (3) until you run out of bundles and transactions or reach the block's gas limit.

Doing this in today's centralized infrastructure takes a lot of code and an extensive infra setup. If you're interested in the details, check out the open-source [Flashbots Builder](#).

Let's take a look at how we can replicate the above effective-gas-price building algorithm as a smart contract using the MEVM instead:



Here the solidity is precisely the same as normal solidity but with additional special functions that can be called to aid in the block-building process. In this example, we can get Ethereum mainnet state, simulate bundles, and make blocks available to proposers using precompiles like `getState()`, `simulateBundle()`, and `exportBlock()`. These special precompiles are used alongside standard solidity functionality to define and execute a simple block-building process that resembles what the Flashbots Builder has long executed.

SUAVE as a Market For Mechanisms

The above example features just a few of the precompiles we've added to the standard EVM. The guiding principle is to give developers superpowers that **enable them to build new applications that they cannot build on Ethereum in a decentralized way today**. Initially, we are targeting support of a few types of new applications: * Applications that require **private data**, e.g., auctions, block building * Applications that require **coordination within block times**, e.g., block building, trade routing and filling * Applications that require **access to fresh off-chain data**, e.g., trading strategies conditional on centralized exchange prices or transactions that are conditional on other transactions * Applications that are too **expensive to do on-chain** because of how much compute they use, e.g., block building

SUAVE is a decentralized platform designed for developing these applications with the properties they need: low latency, privacy, credible computation, and composability. It allows application developers from the existing MEV supply chain to rebuild their applications as smart contracts on SUAVE, where they can leverage its decentralization and credible compute. The MEVM is at the core of this by offering a simple, flexible, and familiar interface for developers: the EVM and its tooling ecosystem (solidity, foundry, etc).


Another important capability that the MEVM offers is privacy. Without privacy, sensitive data cannot be shared due to the risk of frontrunning and MEV stealing. The MEVM offers privacy and efficiency by moving compute on sensitive data into execution nodes that run off-chain. Flashbots or another 3rd party will initially run execution nodes in a trusted fashion. In the next release, execution nodes will be in SGX to lower the trust needed in centralized operators. Eventually, we want to augment SGX with cryptography so that all compute on sensitive data is encrypted and trust requirements are as low as possible.

The permissionless nature of SUAVE, the superpowers it gives developers, and the drastically lower barriers to developing and deploying new MEV applications make SUAVE a *market for mechanisms*. It allows for a proliferation of new mechanisms that users can choose and benefit from. In turn, users will get better outcomes, and proposers will get better blocks.

SUAVE Centauri

Flashbots plans to launch a SUAVE Devnet in Q4 of this year (2023), thus completing the Centauri release and offering an initial version of the MEVM. This devnet will enable community experimentation and stress testing before a mainnet as a part of the next release, *SUAVE Andromeda*.

At a high level, the initial *Centauri* release has the following architecture:

 * **Execution node**: a node that provides credible and private off-chain compute that can be used in smart contracts on SUAVE through special precompiles. Initially, Flashbots or another 3rd party will run an execution node. In the future, trusted execution environments and/or cryptography will be used, obviating the need for trust in centralized parties. * **Confidential data store**: a place where confidential data can be stored for usage in execution nodes. * **Bids**: a new transaction type in SUAVE that contains confidential data the user wants executed and a list of contracts that are allowed to access the user's confidential data. * **SUAVE Chain**: a modified EVM chain with MEV-specific precompiles designed for usage alongside credible off-chain execution facilitated by execution nodes. The chain is used for the deployment of smart contracts and for coordination between parties. In the future, the chain will also escrow funds used to pay for preferences, for oracle updates, and data availability.

There are four primary stakeholders of SUAVE initially: * **Developers** create smart contracts on SUAVE Chain that define rules for MEV applications like OFAs and block building. Developers benefit from a flexible and efficient way of creating their applications and access to user orderflow they otherwise wouldn't have access to. * **Users** send bids to SUAVE. These comprise confidential data and a set of contracts the user allows to interact with its bid. Confidential data is stored in a confidential data store off-chain. Users benefit from SUAVE's market of mechanisms competing to provide them best execution, and they can access MEV applications without having to switch RPCs as they do today. * **Executors** attempt to execute bids using smart contracts that define how they can interact with users' confidential data, e.g., using a backrun contract to place an arbitrage transaction behind a user. Executors benefit by making money from executing bids and from being able to leverage orderflow they otherwise wouldn't have access to. * **Proposers** listen to SUAVE for full blocks. Proposers benefit from having access to valuable blocks. More technical details - and open source code - can be expected soon (™) before the launch.

Andromeda and Beyond

Again, the initial release with *Centauri* is intended to act as a platform for experimentation and hardening SUAVE before a mainnet. As a part of this, we will showcase how existing MEV infrastructure - such as the Flashbots Builder or MEV-Share - can be developed as smart contracts and invite others to develop their own MEV applications alongside us.

The *Centauri* release initially leverages trust in Flashbots to provide privacy and credibility in execution nodes. We think of this release as a clean interface reference, or in cryptography terms, an *Ideal Functionality*, since the cryptography and TEE are kept out of this release completely. As a part of the 2nd release, *Andromeda*, we will run execution nodes inside of trusted execution environments ("TEEs"). In doing so, we endow off-chain computation with the privacy and integrity that the vision of SUAVE requires. Moreover, this allows far more parties to run execution nodes, as TEEs can provide integrity and privacy guarantees even in the presence of a malicious operator. This architecture is also intended to be modular and future-proof, allowing for execution nodes to be run using pure cryptography when that is ready.

Lastly, we plan to work with other domains to showcase how they can integrate SUAVE and use it as a plug-and-play builder by leveraging block-building smart contracts on SUAVE.

Call to Action

We are looking for others to join us in hardening and experimenting with SUAVE: * **MEV infrastructure operators** (builders, OFAs, relays, etc.): Help us understand your requirements for precompiles and SUAVE as a platform. Work closely with us to port your existing off-chain applications to SUAVE. Build net-new applications that require privacy and credibility at the time of block building. * **Other domains (rollups, L2s, etc.)**: Reach out to us to talk about how you can integrate and leverage SUAVE. * **Users**: Switch your RPC to the [Flashbots Protect RPC](#) to receive MEV payments via leveraging MEV-Share. * **Wallets**: Switch your RPC to the [Flashbots Protect RPC](#) to extend privacy and redistribution coverage to all of your users via leveraging MEV-Share. * **dApps**: Switch your RPC to the [Flashbots Protect RPC](#) to extend privacy and redistribution coverage to all of your users. Reach out to us to talk about how you can replicate your centralized off-chain infrastructure (e.g. relays) as smart contracts on SUAVE. * **Searchers**: Search on [mev-share](#) orderflow. Listen to SUAVE smart contracts for new bids and use their hints to search. Leverage verifiable, transparent, and credible rules for block building.

Thanks to [Georgios](#), [Luke](#), [Jon](#), Michał, [Uma](#), and many Flashbots mates for their review and comments on early drafts of this post. Special shout out to [Andrew Miller](#) for the close collaboration on this blog post and many insights related to the MEVM.