Research Explored: The Composable Virtual Machine — Seamless Cross-Chain Transactions

Composable Foundation

Follow

--

Listen

Share

If you've been keeping up with our Composable Research Forum, you're aware of the transformative capabilities that will soon be brought to life by the Composable Virtual Machine (CVM). For those who find the technical terms daunting, let's unpack the latest CVM research post in an easy-to-understand way.

Read & Engage

The full research post produced by two of Composable's core research contributors, Dzmitry Lahoda and Omar Zaki is available on our forum. We highly encourage all interested and technically savvy followers to explore it:

The Composable Virtual Machine: Technical Overview and Example Case

By Dzmitry Lahoda, 0xbrainjar Introduction User transactions and intents need a language for execution across different…

research.composable.finance

What is the Composable Virtual Machine (CVM)?

CVM is an effective 'tool' that simplifies the complexity of executing transactions across different blockchain ecosystems and chains. Simply put, it allows users to make complex transactions involving multiple blockchains with ease and with a single user-signed transaction — this inturn empowers all IBC connections and applications operating across them to execute user intentions more efficiently.

User transactions and intents need to be universally translatable across different blockchain environments. CVM is an important product to our vision that powers the IBC Protocol to fulfill this universal demand.

An Example:

Imagine a user wants to move their DOT (Polkadot) to Osmosis, swap it for ETH, and then move that ETH to Ethereum, swap it for USDC, and finally move those funds to Composable's Cosmos chain.

Here is what should happen semantically:

With CVM, this entire process can be initiated and executed with a single transaction by the user. Check it out in this simplified depiction:

How Does CVM Work?

In tech-speak, the CVM utilizes opaque contracts to execute non-atomic, asynchronous, trust-minimized, non-custodial, bridge-agnostic, and Turing decidable cross-chain programs. This involves code propagated along with data, accessing de facto native liquidity markets on each chain and abstracting away several underlying transports, encodings, assets systems, and messaging protocols.

In simpler terms, CVM intelligently manages all the intricate requirements to carry out a transaction between multiple different blockchains and ecosystems.

Architecture and Flow

Here is the logical state of the CVM:

CVM Account:

The user (wallet) has an executor (CVM account) on each chain (instantiated on demand).

What does that mean?

For each user and for every blockchain network they interact with, the CVM creates a specialized executor account known as an “CVM account.” This account is responsible for handling all the complex operations required to carry out transactions from one blockchain to another. Essentially, it serves as a kind of automated agent that performs cross-chain transactions on behalf of the user, taking care of the technical details.

The CVM is bridge-agnostic, with full capabilities when the target chain allows permissionless contracts and general message passing, as well as limited shortcuts when no custom code is allowed and only a subset of target cross-chain operations are possible.

Example: a hop from one chain to another chain in detail (additional hops will just repeat part of the image):

What does the image show?

The above diagram starts with a fully CVM-enabled flow.

In the end, it shortcuts VM execution to a standard protocol without doing SubProgram execution. This happens if there is an incentive to do a shortcut (for example, gas costs or limited CVM support on the target) and the subprogram is simple, like a transfer.

All the CVM receives is CVM programs. It then uses routing configuration (of assets, bridges, and their features) to dispatch programs over underlying protocols.

What does that mean?

The diagram shows two ways the CVM can work. First, it can fully run a complex program. Second, it can take a simpler route for easy tasks, like transferring funds, especially if this is more cost effective or more efficient with certain systems. Ultimately, CVM gets program instructions and then decides the best way to carry them out based on the setup of assets and connections it has.

CVM Execution Semantics: Gateway & Router

In the CVM, each blockchain contains a Router and a Gateway that work together to handle transactions. The Gateway serves as a bridge aggregator, while the Router processes incoming programs, manages assets, and controls an Executor instance for carrying out cross-chain transactions. This setup allows for modularity and state maintenance across different protocols.

Other notable features of CVM:

- Bridge-Agnostic
- Multi-Ownership & Security Structure
- Optional Fees
- Global Asset Identifier Support

*make sure you check out the

[full research post

](https://research.composable.finance/t/the-composable-virtual-machine-technical-overview-and-example-case/44)to learn more about these features and benefits

Why Does CVM Matter?

For Users:

CVM streamlines your experience by abstracting away complicated terms and protocols. With our trust-minimized, non-custodial approach, you own your assets at every step. It’s cross-chain asset management, simplified.

For Developers:

CVM offers native support for various liquidity markets, asset systems, and even bridges. This makes the CVM the most versatile cross-chain solution available. All you need is to write your program once and watch it execute anywhere, seamlessly.

Example: Cross-Chain Exchange (swap)

Traditionally, users and applications were trading tokens that were only available on their native chain. If you were operating on a chain X, you would only be able to swap tokens that were registered on X (be it native, ERC20 virtualized, or even virtualized and wrapped tokens like WETH).

Manipulating tokens is very complex already. Users willing to move assets between chains are facing incredible difficulties. Not only is bridging hard, but it is also insecure. In most cases, bridges are centralized and hackable by design. We, at Composable, try to push the blockchain vision forward, trustless from the start to the end.

In this example, we will execute a cross-chain swap through CVM and understand how programs are relayed, instructions are executed, and funds are transferred. Be aware that this use case can be generalized to any DeFi protocol and it is the reason why the CVM makes protocols cross-chain native.

Under CVM, tokens are free to fly between any chain connected to its network. They can not only be traded regardless of their origin, but also are completely abstracted thanks to a globally unique CVM asset identifier.

Alice could submit the following CVM program, along with 250 PICA to execute a cross-chain swap:

1. Alice submits the CVM program and the instruction 1. is executed, resulting in:

i.

the child CVM program, consisting of the instructions [1.1., 1.2., 1.3.] is being submitted within an IBC packet to Osmosis.

ii.

the funds attached to the child program, 250 PICA, are being transferred to Osmosis using anICS20 transfer.

1. An IBC relayer, listening to on-chain events, determine that relaying the IBC packet containing the CVM program is profitable. It proceeds and relays both the funds and the packet to Osmosis.
2. The packet is being submitted by the relayer and subsequently processed on Osmosis, resulting in the child CVM program being executed:

i.

the instruction 1.1. is executed: 25 PICA are transferred to the relayer.

ii.

the instruction 1.2. is executed: 200 PICA are traded against an unknown amount X of OSMO.

iii.

the instruction 1.3. is executed, resulting in:

— a.

the second child CVM program, consisting of the instructions [1.3.1., 1.3.2.] is being submitted within an IBC packet to Picasso.

— b.

the funds attached to the second child program, 100% of the OSMO and 100% of the remaining PICA are being transferred to Picasso using an ICS20 transfer.

1. Finally, an IBC relayer determines that relaying the program is again profitable, the packet and the funds are relayed to Picasso.
2. The packet is being submitted by the relayer and subsequently processed on Picasso, resulting in the second child CVM program being executed:

i.

the instruction 1.3.1. is executed: 100% of the remaining PICA (25 + dust from the swap) is transferred to the relayer.

ii.

the instruction 1.3.2. is executed: 100% of the OSMO is transferred back to Alice.

Closing Thoughts

The Composable Virtual Machine (CVM) will essentially be a universal remote control for users to manage their digital assets securely

across multiple blockchains and ecosystems. It streamlines the entire process, making it easier and more efficient without compromising on security thanks to the power of IBC.

Interested in diving deeper into CVM?

- Detailed Research Post
- CVM Documentation
- CVM Design Model
- CVM GitHub Repo
- CVM Contracts on Composable Cosmos Chain (Mainnet)
- CVM Contracts on Osmosis (Mainnet)