MultiversX

Ethereum

Powered by
MultiversX Sovereign Chains

Ethereum^{X} – MultiversX Sovereign Chains SDK Enables Next Generation Ethereum Layer 2s

Abstract

Ethereum Layer 2 solutions, including Optimistic Rollups and Zero-Knowledge (zk) Rollups, offer a way to increase transaction throughput and lower transaction fees, effectively acting as bypasses to Ethereum's congested network. However, common issues with existing L2 solutions include centralized sequencers, fees paid in different tokens (not ETH), lengthy withdrawal times back to L1. Moreover, L2s do not inherently solve Ethereum-specific problems, such as re-entrancy attacks, wallet draining or Miner Extractable Value (MEV).

An Ethereum L2 solution running on MultiversX Sovereign Chains brings important improvements and new features. Capable of ~4,000 DEFI/Gaming TPS, with an average of $0.x USD paid in any custom defined native gas token, and 100% EVM compatible, a Sovereign Chains based Ethereum L2 would be state of the art.

What makes it stand out further would be significant improvements to security, such as On-Chain 2FA, no re-entrancy or wallet drain attacks by design. Notably, the architecture does not rely on any sequencers, so such an L2 would be decentralized and impervious to MEV. Withdrawals to L1 would happen in minutes instead of hours, and applications running on it would benefit from Smart Accounts, the ESDT token standard and direct interoperability with the MultiversX blockchain & other Sovereign Chains running i.e. Bitcoin VM, Solana VM, and so on.

Ethereum, with its decade-long establishment, has garnered immense trust, a vast developer community, extensive toolkits, and the economic security underpinned by the widely recognized ETH token. When scaled via the novel MultiversX blockchain architecture that is designed to be impervious to Ethereum's legacy issues, several improvements placed years into the future on Ethereum's roadmap can be realized today.

Here we introduce an architecture document that gives a high level overview about how such a Ethereum Layer 2 solution would look like, with opportunities to double click on details and go down the proverbial rabbit-hole of relevant details for how the innovations in MultiversX underlying technology can make this a reality.

Simplified fact-sheet

Ethereum Layer 2 solutions, including Optimistic Rollups and Zero-Knowledge (zk) Rollups, offer a way to increase transaction throughput and lower transaction fees, effectively acting as bypasses to Ethereum's congested network.

However, common issues with existing L2 solutions include:

- **centralized sequencers**
- **fees paid in a different tokens (not ETH)**
- **lengthy withdrawal times back to L1**

Moreover, L2s do not inherently solve Ethereum-specific problems:

- **re-entrancy attacks**
- **wallet draining**
- **Miner Extractable Value (MEV)**

An Ethereum L2 solution running on MultiversX Sovereign Chains brings important improvements and new features. At a glance:

- **~4,000 DEFI/Gaming TPS**
- **$0.x USD average TX fee**
- **Fees paid in ETH**
- **100% EVM compatible WASM VM**
- **Fast withdrawals to L1 (x minutes v 10h+)**
- **Smart Accounts**
- **ESDT token standard**
- **Direct interoperability with MultiversX & other Sovereign Chains**

Security improvements:

- **~4,000 DEFI/Gaming TPS**
- **$0.x USD average TXfee**
- **Fees paid in ETH**
- **100% EVM compatible WASM VM**

Here we introduce an architecture document that gives a high level overview about how such a, Ethereum Layer 2 solution would look like, with opportunities to double click on details and go down the proverbial rabbit-hole of relevant details for how the innovations in MultiversX underlying technology can make this a reality.

Introduction and Overview

The SovereignChains SDK is the culmination of extensive development, research, and architectural design. It leverages advanced technologies from the MultiversX Layer 1 chain, providing a comprehensive package for developers to create new appChains. This package includes explorers, wallets, bridges, VMs, SDKs, and more, ensuring that SovereignChains inherit the innovative properties of MultiversX.

Innovations and Features

1. **Consensus and Transaction Processing**

SovereignChain employs a novel consensus technology that reduces the Practical Byzantine Fault Tolerance (pBFT) communication rounds from 5 to 3. Transaction processing occurs in parallel across all validators and the leader, dedicating approximately 90% of the time to transaction processing.

2. **Built-in MEV Protection**

SovereignChain includes built-in protection against Miner Extractable Value (MEV) exploitation, ensuring a fairer and more secure environment for users and developers.

3. **Smart Accounts and Security**

All accounts are Smart Accounts, with built in functions such as key value storage at account level.
On-chain 2FA provides protection for user funds, even in the event of a compromised seed phrase.

4. **Relayed Transactions and Paymasters**

Relayed transactions are fully supported, with the relayer covering the gas fees on behalf of the user.
Integrated Paymasters with RelayedV3 further enhance this functionality.

5. **Interoperability and Cross-Chain Operations**

Integrated interoperability is achieved through Native Cross Chain Operation modules, facilitating seamless interactions with the MultiversX mainchain, Ethereum, and other blockchain networks.

6. **Performance and Efficiency**

SovereignChain is designed for high performance, achieving up to 30,000 simple transactions per second (TPS) or ~4,000 complex (DEFI, Gaming, etc) transactions per second, with 1-second block times. This performance is customizable and can be achieved on inexpensive hardware, such as systems with 4CPUs and 16GB RAM. On machines with bigger specifications 100K TPS and >10K Complex DeFi Ops/S can be achieved.

7. **ESDT token standard**

ESDTs (Electronic Standard Digital Tokens) are versatile token standards supporting fungible, non-fungible, semi-fungible, and DeFi assets. These native assets can be launched permissionlessly, providing a robust framework for various applications.

8. **SpaceVM**

SpaceVM is the fastest and most secure VM, running on top of WASM, with the most OPCODES for builders. It provides a comprehensive and easy-to-use API, ensuring fast and secure execution of smart contracts.

9. **Integrated DNS and Liquid Staked Assets**

Integrated DNS at the protocol level (Herotags) and complete support for Liquid Staked assets help decentralization rather than centralization.

10. **On-Chain Governance and Composable Tasks**

SovereignChain supports on-chain governance and complex composable tasks, enhancing the flexibility and functionality of the blockchain.

11. **Re-entrancy Protection and SafeMath**

Re-entrancy protection is built into the design
Integrated SafeMath ensures no overflows.
SpaceVM includes integrated crypto OPCODES, BigFloats, BigInts, and BigDecimals.

12. **Advanced ESDTs and SECP256R1 Crypto OPCODE**

ESDT V2 supports dynamic ESDTs, ideal for games, tickets, and dynamically evolving NFTs.
The SECP256R1 crypto OPCODE enables SC-based accounts controlled through passkeys from phones, providing a complete Web2 experience in Web3.

13. **BLSMultiSigVerifier:**

The BLSMultiSigVerifier enhances security and efficiency in multi-signature verification.

EVM Integration and Enhancements

The EVM interpreter is added as an executor inside SpaceVM, allowing it to access blockchain data more efficiently. This integration improves the performance and security of EVM, making it faster and more secure while maintaining full compatibility with existing dApps and smart contracts. More pre-built contracts can be added to access the special opcodes from the SpaceVM.

Benefits of SpaceVM Abstraction Layer:

- **Execution on cache, not directly on the Patricia Merkle trie, significantly faster.**
- **Opcode optimizations and the addition of more opcodes, SafeMath, crypto libraries, elliptic curves, and more compiled binaries.**
- **Parallel execution of EVM transactions, enhancing performance.**
- **Built-in MultiCalls and batch transactions for safer ERC token interactions.**
- **Linear storage on EVM, providing protection against re-entrancy attacks.**

MultiVM Support

SovereignChains can run multiple VMs (WASM and EVM) in parallel, with seamless composability. Developers can write contracts in over 30 languages and deploy them on WASM, interacting with existing EVM contracts and Ethereum through asynchronous calls and the native cross-chain operation module.

1. SovereignChains can by default run Multiple VMs, not limited to only one.

2. All the VMs added are atomically composable one to the other.

3. The abstraction layer on top of the VMs is atomically changing the VMInput and Output style to the VM which needs it. Developer does not need to know on which VM is running his SC or what VM is the SC he is calling. Completely seamless.

4. New developers who do not know Solidity can easily write their contracts in >30 languages and deploy on WASM and still interact with the existing EVM contracts and even contract on Ethereum as well – through asynchronous calls and the native cross chain operation module.

5. Build complex DeFi apps on top of Ethereum liquidity with ease, with higher security, more operations, easier UX.

6. WASM execution is faster and cheaper, so more operations can be performed per tx.

Resolving Wallet Drain Problems

One of the biggest problems on EVM chains and especially with ERC token (which should not exist) is the wallet drain problems. Users are constantly losing funds by signing increase allowance/approve/permit2 or other endpoints which give access for scammers to drain the users funds. ERC tokens are like a 3rd party deposit box where you need to share your key with apps you want to interact with. 34000 people lost funds during April 2024. As the whole ecosystem depends on ERC tokens (20, 721, 1155), those cannot be changed, but how those tokens are used can be greatly improved to make EVM safer, using the properties of ESDTs and SpaceVM.

SovereignChains address wallet drain problems by using native assets and built-in wrappers. Assets are wrapped to ERC tokens only for transaction execution and unwrapped back to native assets afterward. This approach ensures secure transactions and eliminates 99% of wallet drain problems.

1. The basic idea is to have all assets as native ones, and wrap those towards ERC tokens only for the execution of the transaction, after the execution to unwrap everything back to native assets again - using the power of ESDT.

2. User makes ONE and only ONE transaction and all the execution is atomic, if one of these fails tokens are transferred back and state is reverted
   ESDTTransfer@USDC@1000@wrapToERC@swapOnUniswap@WrappedETH@unWrapResults

3. First the ESDT token is sent to the ESDTToERCWrapper contract (can be written in Solidity thus running on EVM, or written in RUST and run on WASMSpaceVM which has atomic composability with EVM). In the ERC20 contract the user will receive the USDC tokens, this is interpreted as transfer from the wrapper contract to the user. (when unwrap happens inside the USDC ERC20 contract the tokens are moved from the user towards the wrapper contract address).

4. After the user has the funds in the ERC20 contract (or any standard contract) the system will call the approve function towards the dApp the user wants to interact with. In this case the uniswap contract receives approval from the user to use the USDC funds.

5. The next atomic action is the call for the actual execution, meaning the swap towards wrapped ETH. This swap will happen on the ERC20 contracts as Uniswap knows those.

6. The user receives the tokens in the WETH ERC20 smart contract.

7. The spaceVM watches all the actions in which transfer of tokens is called towards the user for ERC tokens and for all those tokens the SpaceVM will automatically call unWrap which technically means that user will send the WETH token inside the ERC20 contract towards the wrapper contract, and the wrapper contract will release the native ESDT WETH token towards the users address

This example will work with MultiESDTTransfer calls as well. Where at step 3 and 4 the transfer and approval will happen for all the tokens. In this way all the user sees and interacts with are native tokens, which are much more secure than ERC tokens, and will eliminate 99% of the wallet drain problems.

In order for projects to be able to launch fully fledged Ethereum apps, without any change on top of the Ethereum SovereingChain, the actual EVM interpreter is not changed. The EVMs interaction with the underlying structure can be greatly improved - i.e.the trie can be changed faster to Verkle Tries.

Taking Ethereum to the power of X

There are several possible directions:

1. Running MultiVM, SpaceVM with WASM and EVM it is quite possible to use the existing rollup/connection methods from the L2 space and leverage the existing stacks.

2. The most used ways to connect are: Optimistic rollups, ZK rollups, or 3rd party bridges.

3. In terms of optimistic rollups the existing open source stack are OptimismSDK, Arbitrum SDK, and from ZK rollups there are a big set of possibilities. Teams can choose any stack and add it to the SovereignChain SDK to have an integrated interoperability.

One possible way to do the ZK rollups is to leverage ZK-EVM Type1 from Polygon.

1. SovereignChain by default makes settlements on the MultiversX mainchain for all the ESDT assets

2. EVM is implemented deep inside the SpaceVM codespace as an executor.

3. As SpaceVM is used, that one already has the ExecuteOnOtherVM function implemented, thus there is a direct composability between WASM and EVM contracts.

4. A wrapper contract standard is built from ESDT to ERC token standards and one wrapper standard from ERC to ESDT. Thus achieving interoperability of tokens.

5. Now there is no need to settle everything that happens on the WASM/protocol side (actually would need to create different ZK circuits for the ESDT transfer opcodes, not only pure WASM) on Ethereum, we could settle only the execution of the EVM codes. This is somewhat the same rationale that you do not need to settle everything on a gaming chain, but the assets.

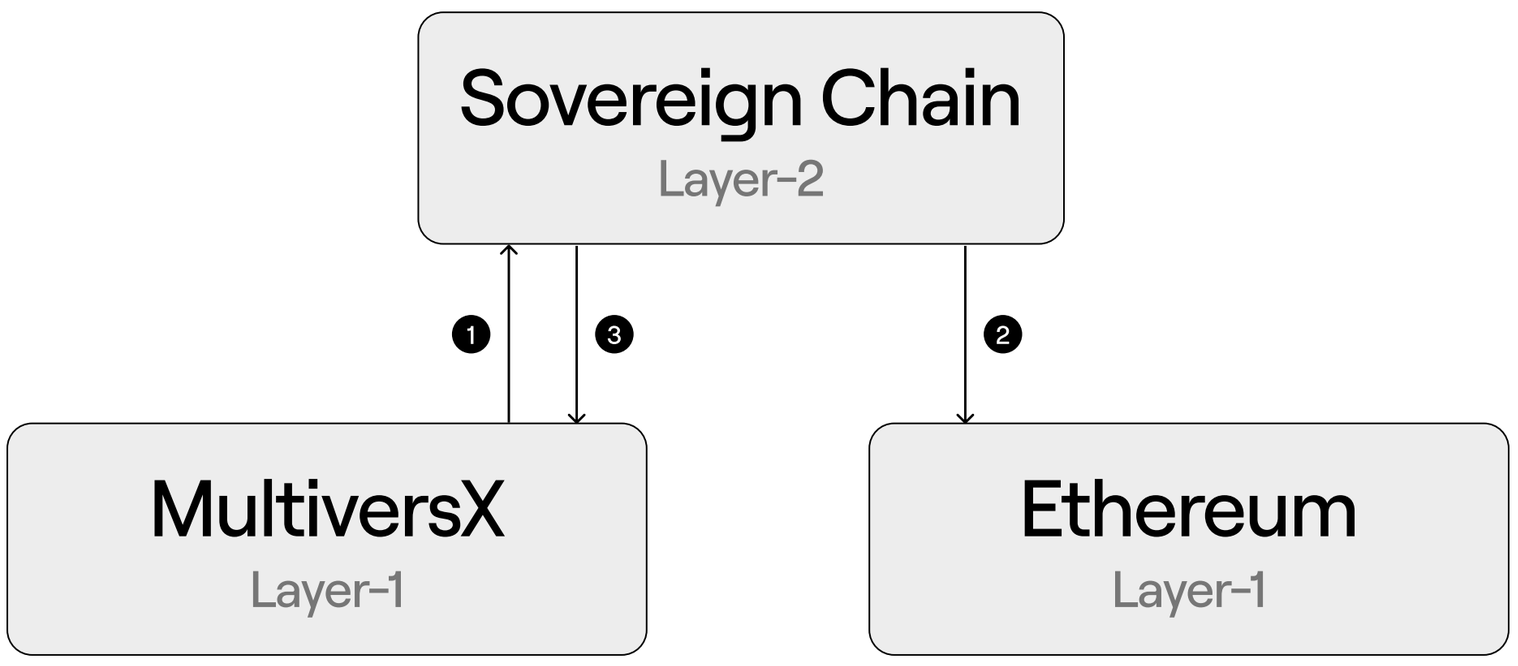
    a. Here comes ZK-EVM type1

        i. **https://polygon.technology/blog/upgrade-every-evm-chain-to-zk-introducing-the-type-1-prover**

        ii. **https://github.com/0xPolygonZero/zk_evm/tree/develop**

    b. This enables for any ecosystem to have its own part of EVM code, running validators and creating blocks to have ONE prover node attached to it which will create the ZK-EVM proofs of execution for that chain. The resulting proof can be sent to the Polygon aggregated layer or directly to Ethereum.

    c. This way a SovereignChain could become an L2 on top of Ethereum as well, this is the choice of the validators who run that Chain.



If someone wants to deploy the SovereignChains without the WASM executor it is completely possible to do it and run with EVM only and ERC tokens only. But they would miss a big chunk of innovations and possibilities.

The current state of decentralisation and security risk on the well known ETH L2s:
https://l2beat.com/scaling/risk

Improved connection, built in interoperability and native cross chain operations towards Ethereum from the SovereignChains

In case of the ZK-EVM type1 integration, there is a need to run only on the ZK rollup creator which will do the proof generation and push that to the Ethereum mainchain. In general these sequencers are centralised in the EVM ZK verifier contracts, but using deep integrations into the SovereignChain SDK this could be heavily enhanced with the following steps:

1. Running MultiVM, SpaceVM with WASM and EVM it is quite possible to use the existing rollup/connection methods from the L2 space and leverage the existing stacks.
2. The most used ways to connect are: Optimistic rollups, ZK rollups, or 3rd party bridges.
3. In terms of optimistic rollups the existing open source stack are OptimismSDK, Arbitrum SDK, and from ZK rollups there are a big set of possibilities. Teams can choose any stack and add it to the SovereignChain SDK to have an integrated interoperability.

Starting from the Native Cross Chain operation module between MultiversX and SovereignChain, and leveraging the learnings from its development, it is possible to establish an integrated native cross-chain operation between SovereignChain and the Ethereum mainchain or all EVM-compatible chains.

This Native Cross Chain operation module can function on top of ZK rollups or Optimistic rollups when engineered correctly. It integrates deeply into the transaction creation, processing, and consensus of SovereignChains, enabling native cross-chain operations not only with MultiversX but also with Ethereum and Bitcoin.

All SovereignChain validators will run an Ethereum light client to monitor events and transactions on the L2 contract, starting with deposit transactions. Validators continuously add Ethereum block headers to the SovereignChain BlockHeader. If an Ethereum block contains successful transactions towards the L2 contracts and is finalized, validators will create an INCOMINGTX and add it to the SovereignChain block.

1. If the leader does not add the INCOMINGTXs, the block is not signed by validators.
2. The next honest leader will add the INCOMINGTX. Each validator generates the INCOMINGTX separately, ensuring the same data and order.
3. The INCOMINGTX can include execution, not just transfers, enabling complex interoperability without relying on third-party bridges and relayers, thus speeding up transaction processing. On the Sovereign Chain, an OutGoingTX SC will be created to facilitate faster fund transfers from Sovereign to Ethereum and to enable complex interoperability between contracts on Sovereign and Ethereum, especially those written in WASM using the CrossChain Async module.

From all successful transactions towards the OutGoingTX SC, validators will create an OutGoingTXMiniblock and sign its hash with their private keys, enabling operations to happen instantly on Ethereum.

1. The current consensus leader must post the OutGoingTXs and the signature on Ethereum promptly, monitored by other validators via their light clients.
2. The hash of the outgoing operations list is signed by all validators, allowing instant execution on Ethereum. This can also be done when ZK-rollup data is sent to Ethereum.
3. The L2 contract on Ethereum will verify the multiSignature over the hash. Validators will register their public keys on the Ethereum contract, and ⅔+1 of the active validators must sign the message. This can be done using BLSMultiKey or ECDSA multiSignature.
    a. On epoch change or validator set change, a bitmap signed by current validators will indicate the active validators for the next epoch.

This method ensures the fastest possible movement of funds and operations between SovereignChain and Ethereum, without relying on third parties, by obliging SovereignChain validators to perform the tasks.

For even more advanced functionality, Asynchronous Composability between SovereignChain smart contracts (SCs) and Ethereum dApps is introduced. A new OPCODE for cross-chain asyncCall is added to SpaceVM, allowing contracts to call cross-chain other contracts and ensuring they receive results or funds in case of errors.

This is particularly effective with WASM-based contracts written in SpaceCraftSDK, enabling developers to tap into existing contracts and liquidity from Ethereum directly from SovereignChains.

1. The SC on SovereignChain calls the crossChain asyncCall towards an Ethereum contract, including funds and execution.

2. The OutGoingTX contract is called, and the OutGoingOperation requested by the SC is included in the OutGoingMiniblock signed by validators.

3. On Ethereum, every outgoing TX is executed. For async execution, the L2 SC gathers all received tokens and results.

4. A deposit call is made automatically inside the L2 contract, monitored by validators' light clients.In case of execution failure on Ethereum, a deposit call is made with the same tokens initially used, including the error message.

5. Validators create an incomingTX of callBack, which reaches the SC on SovereignChain, triggering the defined callBack function.

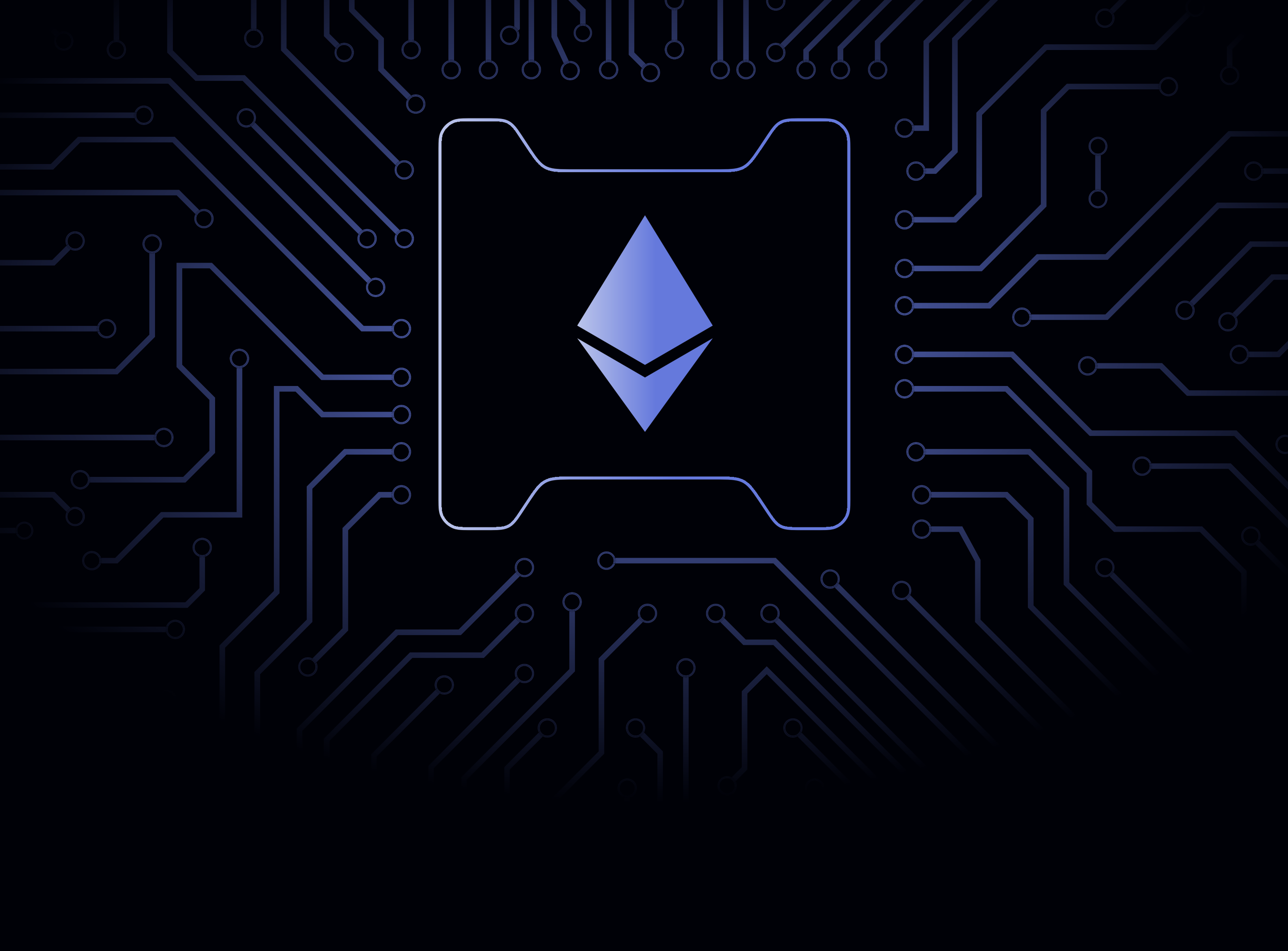
6. The SC on SovereignChain completes its execution.

This approach ensures secure, efficient, and seamless cross-chain operations, enhancing the overall functionality and interoperability.

Conclusion

We have thus introduced a technical path for building the next generation of Ethereum Layer 2 solutions. These will evolve beyond the legacy problems inherent to the original Ethereum architecture. They will also do away with the additional limitations introduced by the current (overcrowded) space of attempts at scaling Ethereum. Better yet, they will be equipped with the new technological capabilities introduced by MultiversX.

Our contribution to the Ethereum and compatible blockchain space is made to build on a vision furthering the existing trust, primitives, tools and capital existing in the space, instead of further fragmenting it.



We call on like minded developers, builders and visionaries to pick up these tools and create something amazing.

It’s time to build!