Fusaka: FVM Integration

Feasibility Analysis, Divergence Mapping & Implementation Plan

Germina Labs

December 5, 2025

Note: "Fusaka" is the official codename for the Ethereum upgrade succeeding Pectra. It supersedes the scope previously identified as "Pectra Phase 2" in earlier Ethereum development roadmaps.

Abstract

This report outlines the results of the "Fusaka" investigation, aiming to align the Filecoin Virtual Machine (FVM) with Ethereum's recent Hard Fork (activated Dec 3rd, 2025). The upgrade focuses on scaling data availability (PeerDAS) and enhancing execution capabilities (secp256r1, CLZ). Our analysis concludes that while FVM should adopt the execution-layer improvements to maintain EVM equivalence for developers, the data availability and gas-layer changes are architecturally incompatible with Filecoin's native storage and metering models. We recommend implementing the three adopted features as **three separate Filecoin Improvement Proposals (FIPs)** to facilitate modular governance and activation.

Contents

1 Executive Summary

The "Fusaka" upgrade represents a significant divergence point for Ethereum, introducing complex consensus mechanisms for "Blob" sharding (PeerDAS). For the Filecoin Virtual Machine (FVM), which operates on a fundamentally different storage substrate, blind adoption of these EIPs is neither feasible nor desirable.

However, Fusaka also introduces high-value application-layer features. The **secp256r1** precompile is essential for the next generation of consumer wallets (Passkeys), and the **Count Leading Zeros (CLZ)** opcode is required for compatibility with future Solidity compilers.

Our "Phase 2" analysis has delivered:

- **EIP Applicability Matrix**: A definitive guide on which Fusaka EIPs to Accept, Reject, or Mock.
- **Architectural Specification**: A technical plan for implementing `secp256r1` in the `builtin-actors` Rust workspace.
- **Client Integration Plan**: A design for the `eth_config` RPC method in Lotus, enabling dynamic capability discovery.

We recommend eschewing a monolithic "Fusaka Bundle" FIP in favor of three targeted proposals covering the specific features relevant to Filecoin.

2 EIP Deep Dive & Applicability Analysis

We performed a detailed technical analysis of each EIP in the Fusaka bundle. Below is the comprehensive breakdown of our findings and strategic decisions.

2.1 High-Priority Integrations (Adopt)

2.1.1 EIP-7951: Precompile for secp256r1 Curve Support

Summary: Adds a precompiled contract at address `0x100` (tentative) to perform signature verification on the `secp256r1` (P-256) elliptic curve.

Analysis: Currently, the EVM only natively supports `secp256k1`. However, the standard for hardware secure enclaves (Apple Secure Enclave, Android Keystore) and the WebAuthn/FIDO2 (Passkeys) standard relies on `secp256r1`. Supporting this precompile enables "Smart Accounts" on Filecoin to be controlled directly by a user's biometrics without an intermediate relayer or expensive off-chain ZK proofs.

Strategy: Implement as independent FIP. This provides high direct value to the FVM DeFi and consumer ecosystem. We will implement this using the Rust `p256` crate within the `builtin-actors` layer.

2.1.2 EIP-7939: Count Leading Zeros (CLZ) Opcode

Summary: Introduces a `CLZ` instruction to the EVM.

Analysis: This is a low-level efficiency upgrade. While trivial in isolation, its absence creates a "compiler trap." If the Solidity compiler (versions 0.8.28+) assumes the target chain supports `Fusaka`, it may emit `CLZ` for optimizations. If FVM lacks this opcode, standard contracts will crash.

Strategy: Implement as independent FIP. Strict bytecode compatibility is non-negotiable for developer experience.

2.1.3 EIP-7910: `eth_config` JSON-RPC Method

Summary: Adds a standard RPC method for clients to query chain capabilities.

Analysis: As L2s and alternative L1s (like FVM) diverge from Ethereum Mainnet, wallets and developer tools need a way to discover what features are available. For example, FVM supports `secp256r1` but *not* `Blob Transactions`. Without this RPC, tools have to guess via `chainId` maps, which is brittle.

Strategy: Implement as independent FIP. We will add this to Lotus. It serves as the handshake protocol to inform connected clients that "PeerDAS is disabled" but "FVM features are active."

2.2 Rejections based on Gas & Metering

2.2.1 EIP-7883 & EIP-7823: ModExp Gas Changes

Summary: These EIPs reprice the Modular Exponentiation precompile to match its actual computational cost on Ethereum clients.

FVM Reality: The FVM ignores EVM Gas Tables entirely. When `MODEXP` is called on FVM, it executes a Rust function compiled to Wasm. The FVM Runtime meters the *actual Wasm instructions executed*. If a ModExp operation takes 10x more CPU cycles, the FVM automatically charges 10x more Gas Units. We are inherently immune to the pricing mismatch that EIP-7883 fixes.

Strategy: Reject. Adopting Ethereum's arbitrary gas formulas would be redundant and technically incorrect for our Wasm metering model.

2.2.2 EIP-7935 & EIP-7825: Gas Limits

Summary: Increases the block gas limit and caps individual transaction gas usage.

FVM Reality: Filecoin's throughput is governed by `Expected Consensus` and `Gas Units`. We have our own block limits (approx. 10B Gas Units) and our own "Tipset" mechanics.

Strategy: Reject. We continue to use Filecoin's native network parameters.

2.3 Rejections based on Data Availability

2.3.1 EIP-7594: PeerDAS (Blob Sampling)

Summary: Introduces a complex p2p sampling layer to verify that "Blobs" (ephemeral data attached to Type 3 transactions) are available without downloading them.

Analysis: This solves the "Data Availability Problem" for Ethereum. Filecoin *is* a storage network. We solve Data Availability with cryptographic `Proof-of-Replication` (PoRep) and `Proof-of-Spacetime` (PoSt). We do not need an ephemeral, 18-day blob layer; we have a permanent sector market.

Strategy: Reject entirely. We will not implement the PeerDAS networking layer, and we will **reject** Type 3 transactions at the parser level (Status Quo). Tools attempting to send blobs to Filecoin should receive a clear error rather than a misleading success.

2.3.2 EIP-7918: Blob Base Fee

Analysis: This is market logic for the PeerDAS blob fee market. Since we are rejecting PeerDAS, this market does not exist on FVM.

Strategy: Reject.

2.4 Other Rejections

- **EIP-7642: `eth/69` (Networking): Reject.** Filecoin uses `libp2p` with its own gossip protocols. We do not share the `devp2p` stack with Ethereum.
- **EIP-7917: Deterministic Proposer Lookahead: Reject.** Specific to Ethereum's Beacon Chain. Filecoin uses `EC` (Expected Consensus) and VRF-based leader election.
- **EIP-7934: RLP Execution Block Size Limit: Reject.** Filecoin uses IPLD (DAG-CBOR) for block headers and messages.

3 Implementation Roadmap

The implementation is organized into three parallel workstreams, each corresponding to a specific FIP. This allows for independent development, testing, and activation.

> **Note:** Due to the well-scoped nature of these individual components, we proceeded directly to implementation within the target repositories (`builtin-actors` and `lotus`), bypassing the need for throwaway standalone prototypes.

3.1 Workstream 1: EIP-7951 (secp256r1 Precompile)

Goal: Enable native Passkey verification on FVM.

1. **Rust Implementation (Complete):** A full implementation has been developed in `builtin-actors-eip7951`.
   - **Location:** `actors/evm/src/interpreter/precompiles/secp256r1.rs`
   - **Logic:** Uses the `p256` crate to implement `verify_impl`, taking 160 bytes of input (hash, r, s, x, y) and returning a boolean.
   - **Address:** Registered at `0x00..0100` (aligned with RIP-7212).
   - **Status:** Code reviewed and unit tests with vectors from `daimo-eth/p256-verifier` passed.
2. **Actor Integration:**
   - Import the logic into `builtin-actors/actors/evm/src/interpreter/precompiles/` (Done).
   - Register the new precompile address (e.g., `0x100`) (Done).
3. **Specification:** Deliver **FIP-X: secp256r1 Precompile** describing the gas costs (Wasm metering) and API.

3.2 Workstream 2: EIP-7939 (CLZ Opcode)

Goal: Maintain compiler compatibility with Solidity 0.8.28+.

1. **Rust Implementation (Complete):** Implemented in `builtin-actors-eip7939`.

- **Modifications:** Added `clz` to `actors/evm/src/interpreter/instructions/bitwise.rs`.
- **Registration:** Registered opcode `0xf6` in `execution.rs` and `instructions/mod.rs`.
- **Testing:** Added unit tests for the new instruction.

2. **Actor Integration:**
   - Update `builtin-actors/actors/evm/src/interpreter/execution.rs` to include the new opcode `0xf6` (Done).
   - Map the instruction to the Rust `u256.leading_zeros()` method (Done).
3. **Specification:** Deliver **FIP-Y: CLZ Opcode** detailing the instruction behavior.

3.3 Workstream 3: EIP-7910 (`eth_config` RPC)

Goal: Enable dynamic capability discovery for wallets and tools.

1. **API Design (Drafted):** Defined `EthConfig` struct in `lotus-eip7910/chain/types/ethtypes/eth_typ`
2. **Lotus Integration (Drafted):**
   - Added `EthConfigAPI` interface to `lotus-eip7910/node/impl/eth/api.go`.
   - Defined fields: `chainId`, `peerDAS`, `fvm`.
3. **Specification:** Deliver **FIP-Z: eth_config RPC Method** standardizing the response format for all Filecoin implementations.

4 Conclusion

The "Fusaka" upgrade presents a streamlined opportunity for the FVM. By focusing strictly on the application-layer features (`CLZ`, `secp256r1`) and rejecting the consensus-layer complexities (PeerDAS, Gas), we can deliver a high-value upgrade with reduced engineering risk. The plan is now set for execution.