

# Invisible zkEVVM

The Confidential and Gasless Virtual Blockchain on Arbitrum

## Authors

Sebastian Lujan, Artur Vargas

A virtual blockchain running inside Arbitrum, **confidential**, **zero gas costs** for users, and **mathematical verifiable** using **zkp**.

**Version:** 1.0.0  
**Date:** November 2025  
**Status:** Development

# Contents

## Quick Summary

---

EVVM is a **virtual blockchain** that runs inside Arbitrum, eliminating servers and gas costs for end users. It combines three critical features:

- **Full Encryption:** Complete privacy through Fully homomorphic encryption
- **Fast & Cheap:** 90% cost reduction via Arbitrum Stylus interfaces
- **Mathematically Secure:** Zero-knowledge proofs for verification and Aztec snapshots

**The Vision:** A network where payments and stakes remain completely private, with near-zero costs to users. Achieved by integrating Fhenix's FHE encryption, Arbitrum's Stylus interface, and the zkFisher relayer.

## Core Technologies

- **Fhenix FHE:** Fully Homomorphic Encryption for private computation
- **Arbitrum Stylus:** High-performance Rust smart contracts
- **zkFisher:** Decentralized relayer with zk-proof validation
- **Solidity Core:** Treasury, payments, and staking management

## The Problem We Solve

---

### Privacy Crisis on Public Blockchains

On networks like Ethereum and Arbitrum, transactions are completely transparent:

**What's Publicly Visible:**

- Transaction amounts
- Sender and receiver addresses
- Account balances
- Transaction history and patterns

This transparency creates **critical risks** for:

- **Banking:** Lack of compliance and AML issues
- **Finance:** Exposing trading strategies and portfolio values
- **Payments:** Revealing spending habits and merchant relationships
- **Gaming:** Showing in-game asset values and trades
- **Enterprise:** Disclosing business transaction volumes

## Current Solutions Fall Short

Approach	Limitations	Cost
Mixing Services	Partial privacy	Medium
zk-Rollups	Complex, not fully private	High
Private Chains	Centralized	Variable
Layer 2s	Still public data	Medium-High

Table 1: Limitations of existing privacy solutions

## The EVVM Solution

EVVM creates a **virtual layer** inside Arbitrum where:

- **Everything is private** by default
- **Users pay zero gas**
- **No central servers** required
- **Full Arbitrum security** inherited

## The Architecture

Invisible zkEVVM operates as a **hybrid network** within Arbitrum, leveraging the strengths of each component:

### System Components

#### 1. EVVM Core (Solidity)

The foundation layer handling critical functions:

- **Payments:** Encrypted balance transfers
- **Staking:** Private staking mechanisms
- **Treasury:** Fund management and distribution
- **FHE Integration:** Fhenix precompile calls

**Key Feature:** Homomorphic encryption enables arithmetic operations (addition, subtraction) on encrypted data without revealing actual values.

#### 2. Stylus Interface (Rust)

A lightweight, high-performance layer providing:

- **90% Cost Reduction:** Rust efficiency vs. Solidity
- **Type Safety:** Compile-time encryption verification
- **Memory Safety:** No buffer overflows or data leaks
- **Fast Execution:** Near-native performance

### 3. zkFisher Relayer

The zkFisher relayer serves as the decentralized coordination layer that bridges encrypted user transactions with on-chain verification. Operating autonomously, zkFisher continuously monitors the network for pending encrypted transactions and orchestrates their processing into verified blocks.

The relayer's workflow follows a sophisticated multi-step process. First, it listens for incoming encrypted transactions from users, capturing their transaction data while preserving confidentiality. These transactions are then intelligently grouped into block structures, optimizing for both efficiency and verification cost. For each block, zkFisher generates cryptographic zero-knowledge validity proofs using the Noir proving system, which mathematically demonstrate the correctness of all transactions without revealing their contents. These proofs are subsequently submitted to Arbitrum's verification contracts, where they undergo rigorous validation through the Groth16 backend. Upon successful verification, the encrypted state is updated on-chain, maintaining complete privacy throughout the entire lifecycle.

**Gasless Design:** The relayer and treasury infrastructure cover all gas costs, enabling completely free transactions for end users. This gasless architecture removes economic barriers to privacy-preserving blockchain interactions.

### 4. Fhenix Connection

Integration with Fhenix's FHE infrastructure:

- Contracts call Fhenix precompiles
- Encrypted operations processed off-chain
- Decentralized computation network
- Results verified and committed on-chain

### Architecture Diagrams

The EVVM architecture can be visualized at two levels of abstraction:

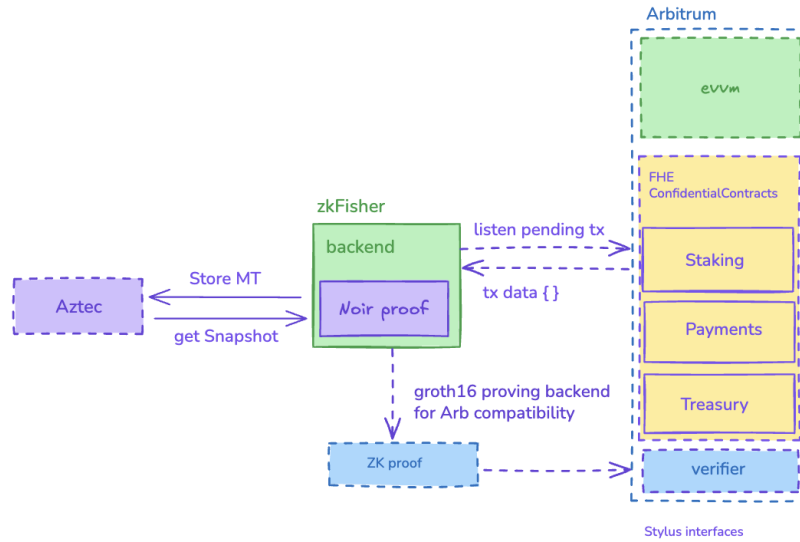


Figure 1: High-level architecture overview showing the main components and their interactions

## Transaction Flow

The EVVM transaction flow creates a seamless bridge between encrypted operations and on-chain verification. We connect the core contracts deployed on Arbitrum (including Staking, Payments, and Treasury) using Fhenix’s Fully Homomorphic Encryption (FHE), with Stylus serving as the high-performance interface layer. The zkFisher relayer acts as the central coordinator, listening to pending transactions and extracting their transaction data.

When a user initiates a transaction, EVVM proposes a **healthy block**, an initially empty block structure ready to be populated with encrypted transactions. The zkFisher relayer continuously monitors for these healthy blocks and begins the batching process. As transactions arrive, zkFisher stores the transaction block header metadata in a Merkle tree structure managed by AZTEC. For this Merkle tree, zkFisher generates a **Merkle inclusion proof**, demonstrating that each transaction is properly included in the block. The root of this Merkle tree is called a **snapshot**, a cryptographic commitment to the state of all transactions at a specific point in time.

At regular intervals defined by epochs (every  $x$  blocks), EVVM captures these snapshots to ensure data integrity and enable efficient verification. The zkFisher then generates a zero-knowledge proof using the **Noir** proving system, which allows for succinct verification of complex computations. To verify this Noir proof on Arbitrum, we employ the **Groth16 proving backend**, a highly efficient zkSNARK verification system that enables Arbitrum smart contracts to validate the cryptographic proofs in minimal gas cost.

For this verification process, we require an **Arbitrum Noir verifier**. This critical infrastructure component is currently being developed as a work in progress by another team and has been approved by the Arbitrum DAO, ensuring community support and alignment with Arbitrum’s ecosystem roadmap. Once the proof is verified through this verifier, the encrypted state is

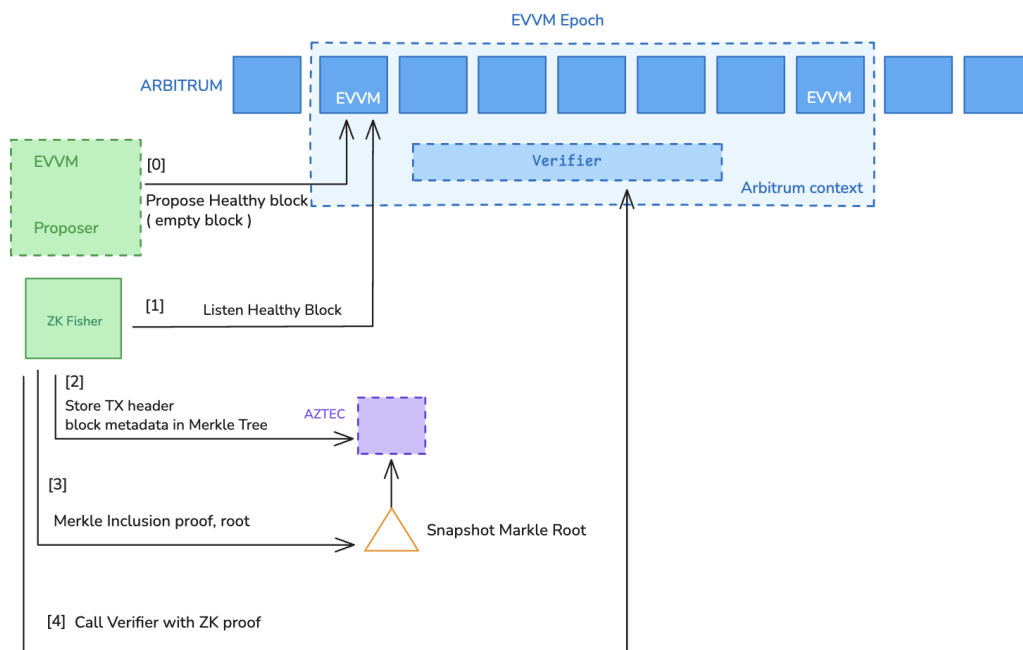


Figure 2: Detailed architecture diagram showing the complete transaction flow and component relationships

committed to the blockchain, and the user receives confirmation, all while maintaining complete privacy throughout the entire process.

**Result:** An independent blockchain integrated into Arbitrum for enhanced security and scalability, without sacrificing privacy. The combination of FHE, Merkle proofs, and Groth16 verification creates a robust, gasless, and fully private transaction system.

## The Benefits

### Total Privacy

- **Encrypted Balances:** No one sees account balances
- **Hidden Transactions:** Transfer amounts remain private
- **Private Recipients:** Destination addresses obscured
- **No Validator Access:** Even validators can't decrypt data

**Perfect for:** Sensitive financial applications, private DeFi, confidential payments, and enterprise solutions.

## Performance

- **90% Cost Reduction:** Stylus efficiency over pure Solidity
- **Fast Blocks:** Seconds, not minutes
- **High Throughput:** Inherits Arbitrum's TPS capacity
- **Low Latency:** Near-instant confirmations

## Security & Upgradeability

- **Audited Code:** Fhenix's FHE implementation is battle-tested
- **Merkle inclusion Proofs:** Snapshots for every epoch
- **Modular Design:** Easy upgrades without breaking changes
- **Arbitrum Security:** Inherits L2 security guarantees

## Scalability

**Arbitrum Foundation:** Leverages Arbitrum's capacity for thousands of transactions per second while maintaining complete privacy.

## Comparison Matrix

Feature	Bitcoin/Ethereum	zk-Rollups	EVVM
Privacy	None	Partial	Full
User Gas Costs	High	Medium	Zero
Speed	Slow	Fast	Very Fast
Scalability	Limited	High	Very High
Security	High	High	Very High
Ease of Use	Medium	Low	High

Table 2: EVVM vs. other blockchain solutions

## Use Case: Confidential Coffee Shop

### EVVMCafhe - Private Payments in Action

EVVMCafhe demonstrates the power of EVVM through a real-world application: a coffee shop where customers pay with cryptocurrency while maintaining **complete financial privacy**. Unlike traditional blockchain payments where transaction amounts and purchase history are visible to anyone, EVVMCafhe ensures that **no one can see** what customers spend or what they buy. This creates a payment experience that combines the benefits of cryptocurrency with the privacy expectations of traditional cash transactions.

## How It Works

The EVVMCafhe payment flow showcases EVVM's elegant integration of privacy technologies. The process begins with **deployment**, where the EVVMCafhe smart contract is deployed on Arbitrum using the Stylus interface, enabling high-performance encrypted operations. When a customer wants to purchase a coffee, they initiate a transaction by **encrypting the payment amount** locally on their device. For example, a \$5 coffee purchase is converted into an encrypted value that only the user can generate but no one can read.

Before the transaction is submitted, the client generates a **cryptographic validity proof** that demonstrates the transaction is legitimate without revealing any sensitive information. The user then **signs the encrypted transaction** with their private key, establishing authenticity and authorization. This signed, encrypted transaction is sent to the EVVM network for processing.

Upon receiving the transaction, the smart contract performs critical **verification steps**, checking both the cryptographic signature and the validity proof to ensure the transaction is authorized and correctly formed. Once verified, the **EVVM Core executes the encrypted payment**, performing homomorphic operations that subtract the encrypted amount from the customer's encrypted balance and simultaneously add it to the coffee shop's encrypted balance. Throughout this entire process, the actual payment amount remains encrypted and hidden from all observers, including validators, block explorers, and even the smart contract itself.

Finally, the system provides **confirmation** to both parties that the transaction has completed successfully. The customer's wallet updates to reflect the payment, the merchant receives confirmation of funds, and all transaction data remains completely private. This seamless flow demonstrates how EVVM makes privacy-preserving payments practical and user-friendly.

## Practical Benefits

- **Merchant:** Receives instant, verified payments without accessing customer data
- **Customer:** Pays with zero gas fees and complete privacy
- **Owner:** Withdraws encrypted funds on demand
- **Privacy:** Transaction amounts never revealed on-chain

### Perfect Applications:

- E-commerce platforms
- Gaming item purchases
- Donation systems
- Subscription services
- Private marketplaces

## Performance Metrics

**Security:** All validations happen on-chain. The system is trustless and cryptographically secure.

Metric	Value
Gas Cost per Order	~10,000 gas
Transaction Time	< 5 seconds
User Cost	\$0
Privacy Level	100%

Table 3: EVVMCafhe performance metrics

## Current Status and Future

### Roadmap

Phase	Timeline	Status
Sepolia Testnet	Q4 2025	✓ Live
MVP Launch	Dec 2025	In Progress
Mainnet Alpha	Q1 2026	Planned
Full Mainnet	Q2 2026	Planned

Table 4: Invisible ZkEVVM development roadmap

### Current Achievements

- ✓ Core contracts deployed on Arbitrum Sepolia
- ✓ Stylus interface implemented and tested
- ✓ zkFisher relayer operational
- ✓ Fhenix FHE integration complete
- ✓ EVVMCafhe demo application live

### Future Applications

EVVM opens the door to entirely new categories of privacy-preserving applications:

#### Private DeFi

- Confidential lending and borrowing
- Private yield farming
- Hidden trading strategies
- Anonymous liquidity provision

## Enterprise Solutions

- Private supply chain tracking
- Confidential payroll systems
- Hidden B2B transactions
- Private procurement

## Gaming & NFTs

- Hidden in-game economies
- Private NFT trading
- Confidential marketplace sales
- Secret item attributes

## Payments

- Privacy-preserving payment rails
- Confidential merchant solutions
- Anonymous subscription services
- Private donation platforms

## Technical Specifications

---

### Network Parameters

Parameter	Value
Base Layer	Arbitrum One (Mainnet) / Sepolia (Testnet)
Block Time	~2-5 seconds
Finality	15 minutes (L1 finality)
TPS Capacity	4,000+ (Arbitrum limit)
Encryption	Fhenix FHE (TFHE)
Proof System	Merkle inclusion proof (Aztec-noir)
Smart Contract Lang	Solidity + Rust (Stylus)

### Contract Architecture

- **EVVMCore.sol**: Main Solidity contract for payments, staking, treasury
- **Stylus Interface**: Rust layer for efficient encrypted operations
- **zkFisher**: Off-chain relayer with proof generation

- **Fhenix Precompiles:** FHE operation handlers

## Summary

---

EVVM represents a **paradigm shift** in blockchain technology: the first fully private, gasless, and scalable virtual blockchain running inside Arbitrum.

### Our Vision:

A future where privacy is the default, not an afterthought.

Where users transact freely without costs or surveillance.

Where businesses build without compromising customer data.

Join us in building this future.

**ZKEVVM: Gasless private infrastructure**