REINEIRA LABS LTD.

ReineiraOS

Programmable Confidential Settlement Infrastructure

Version 0.1 — March 2026

Reineira Protocol Team
Reineira Labs Ltd.

LITEPAPER

Abstract

ReineiraOS is an on-chain settlement engine where every value is encrypted and every release condition is pluggable. Built on Fully Homomorphic Encryption, the protocol provides confidential escrow, cross-chain funding, and operator relay as core primitives — while verification logic, risk evaluation, and compliance are delegated to a pluggable interface layer. The architecture is stablecoin-agnostic, bridge-agnostic, and verification-agnostic: any ERC-20 token can be wrapped, any cross-chain bridge can be integrated, and any proof system can serve as a condition resolver. The reference implementation wraps USDC into cUSDC on Arbitrum Sepolia with Circle CCTP V2 for cross-chain settlement. This litepaper presents the design, architecture, and current implementation status.

Contents

ReineiraOS Litepaper

Programmable Confidential Settlement Infrastructure

Built on Arbitrum. Powered by Fhenix FHE. Open to any verification method.

TL;DR

ReineiraOS is an on-chain settlement engine where every value is encrypted and every release condition is pluggable. You define *what* triggers a payment. The protocol guarantees *confidentiality*, *non-custodial execution*, and *provable compliance* – simultaneously.

Three properties. One architecture:

1. **Confidential by default.** Amounts, owners, and payment status are FHE-encrypted. The chain sees that something happened, never what happened.
2. **Provably compliant without disclosure.** Verification plugins produce yes/no answers from encrypted inputs. The auditor gets the result, not the evidence.
3. **Programmable by anyone.** Two interfaces. Implement them however you want. Deploy. Attach to an escrow. Done.

1. The problem

Stablecoin settlement today forces a three-way tradeoff:

Approach	Private	Compliant	Programmable
Traditional finance	Partially (NDAs)	Yes	No (rigid rails)
Public DeFi	No	Difficult	Yes
Privacy mixers	Yes	No	No
Custodial escrow	No	Yes	Partially

Every existing approach sacrifices at least one property. Privacy requires giving up compliance. Compliance requires giving up privacy. Programmability requires building from scratch each time.

Five specific gaps remain unsolved:

1. **Balance visibility.** On-chain stablecoin balances are public. Every counterparty, competitor, and adversary can see your position.
2. **Settlement rigidity.** Each payment scenario – trade, payroll, milestone, subscription – requires custom contract development with no shared primitives.
3. **Verification trust.** Connecting off-chain events (deliveries, payments, approvals) to on-chain settlement requires trusting centralized oracles or manual processes.
4. **Cross-chain fragmentation.** USDC exists on many chains, but settling an escrow from a different chain requires multiple manual steps.
5. **Dispute resolution.** When settlements go wrong, there is no on-chain mechanism for insurance or automated dispute resolution.

2. The solution

ReineiraOS resolves the tradeoff through Fully Homomorphic Encryption (FHE). The protocol computes on encrypted data without decrypting it. This means:

- A compliance plugin can verify “this address is not sanctioned” on an **encrypted address** and return a boolean – without revealing the address to the chain.
- An escrow can verify “the caller is the owner” and “sufficient funds have been paid” using **homomorphic comparisons** – without decrypting any value.
- A failed redemption attempt transfers zero encrypted tokens, which is **indistinguishable from a successful transfer** of any amount.

2.1 The settlement engine

At the core is a programmable escrow primitive. You create an escrow by specifying an encrypted owner and encrypted amount. Payments flow in – including cross-chain via Circle’s CCTP V2. When conditions are met, the owner redeems.

All sensitive fields (owner, amount, paid, redeemed) are FHE ciphertexts. The contract performs arithmetic and comparisons on ciphertext. It never decrypts.

2.2 The plugin architecture

Release conditions and risk evaluation are **not hardcoded**. They are defined by two pluggable interfaces:

Any contract implementing these interfaces can be attached to any escrow. The protocol does not care whether your logic uses:

Verification method	What it proves	What stays private
zkTLS (Reclaim, TLSNotary)	“Payment was received”	Amount, sender, bank details
Oracle (Chainlink, UMA)	“Price crossed threshold”	Position size, counterparty
FHE quorum voting	“Majority approved”	Individual votes, voter identities
Prediction market	“Event outcome resolved”	Stake amounts, positions
Compliance check	“Address is not sanctioned”	The actual address
Multi-signature	“N of M parties signed”	Identity of non-signers
Time-lock	“Deadline has passed”	Nothing (time is public)

This is the fundamental design principle: **the core settlement engine is verification-agnostic.** It provides encrypted escrow, cross-chain funding, and operator relay. Everything else is a plugin.

2.3 The insurance layer

When verification alone is insufficient, economic protection fills the gap. The insurance subsystem provides:

- **Coverage pools.** Underwriters create pools backed by encrypted cUSDC liquidity.
- **Pluggable risk evaluation.** Each pool selects IUnderwriterPolicy contracts that evaluate risk and judge disputes – using the same plugin architecture as escrow conditions.
- **Automated dispute resolution.** When a dispute is filed, the policy contract produces an encrypted boolean verdict. If upheld, the pool pays the claim automatically.

Insurance is optional. Not every settlement needs it. But when it does, the mechanism is native – not a separate protocol bolted on.

3. Architecture

Six subsystems. Each handles one concern:

Subsystem	Package	What it does
Confidential Tokens	@reineira-os/tokens	Wraps USDC into FHE-encrypted cUSDC. 1:1 backed. 6 decimals.
Escrow Engine	@reineira-os/escrow	Creates, funds, and redeems encrypted escrows. Silent failure pattern.
Insurance	@reineira-os/insurance	Coverage pools, policy registry, automated disputes.
Orchestration	@reineira-os/orchestration	Operator staking, task execution, fee distribution, slashing.
Cross-Chain	@reineira-os/escrow	CCTP V2 receiver: mint USDC, wrap to cUSDC, credit escrow – atomically.
Plugins	@reineira-os/plugin-examples	Reference condition resolvers and underwriter policies.

---

4. How it works: a complete example

Alice hires Bob for a design project. They agree on 1,000 USDC, payable on delivery approval.

Step 1: Escrow creation. Alice calls create() with Bob's address and 1,000 USDC – both encrypted. A condition resolver is attached that checks an external approval source. Nobody on-chain can see the amount, the recipient, or the condition logic.

Step 2: Funding. Alice funds the escrow by calling fund(). The payment amount is encrypted. The escrow engine accumulates payments via homomorphic addition on ciphertext.

Step 3: Verification. Bob delivers the work. The approval is verified through the attached condition resolver – which could use any method: a zkTLS proof from a project management API, a multi-sig from reviewers, an oracle feed, or a simple owner approval. The protocol does not prescribe how verification works.

Step 4: Redemption. Bob calls redeem(). The contract verifies three conditions homomorphically: (a) caller is the encrypted owner, (b) paid amount >= escrow amount, (c) not already redeemed, and (d) condition resolver returns true. If all pass, Bob receives the encrypted cUSDC. If any fail, Bob receives zero – silently, with no information leakage.

Step 5 (optional): Insurance. If Alice purchased coverage, she can file a dispute if Bob never delivered. The underwriter policy evaluates the dispute proof and returns an encrypted verdict. If upheld, the insurance pool pays the claim automatically.

Step 6 (optional): Unwrap. Bob calls redeemAndUnwrap() to convert encrypted cUSDC back to plaintext USDC in a single transaction.

5. Technology stack

Layer	Technology	Role
Encryption	Fhenix CoFHE (TFHE)	Homomorphic computation on encrypted data
Settlement	Arbitrum (L2)	Transaction execution and finality
Token	Circle USDC	Underlying stablecoin (1:1 backing)
Cross-chain	Circle CCTP V2	Native USDC burn-and-mint across chains
Smart contracts	Solidity 0.8.25	UUPS upgradeable, ERC-7201 namespaced storage
Operator services	NestJS + TypeScript	Coordinator, relay agents, CLI
Plugin verification	Any	zkTLS, oracles, quorum, prediction markets, custom

6. Use cases

Confidential OTC trading

Two traders agree on a price. The seller creates an escrow. The buyer funds it via CCTP from any chain. A condition resolver verifies asset delivery. Funds release. Neither the trade amount nor the counterparties are visible on-chain.

Milestone-based payments

A platform creates three escrows for a development contract: 1,000 USDC per milestone. Each escrow has a condition resolver tied to an external approval source. When the client approves a milestone, the resolver returns true and the developer can redeem. If a milestone is disputed, insurance adjudicates.

Cross-border payroll

An employer creates recurring escrows for each employee. Amounts are encrypted (salary privacy). Employees redeem and unwrap to USDC on their preferred chain. Compliance plugins verify employment status and jurisdiction requirements without revealing personal data.

Subscription billing

A service provider attaches a time-based condition resolver to monthly escrows. Funds release automatically at each billing cycle. The subscriber can dispute via insurance if the service was not delivered.

---

7. Comparison

Capability	Traditional escrow	Public DeFi	Privacy protocols	ReineiraOS
Amount privacy	No	No	Yes	Yes (FHE)
Programmable conditions	No	Partially	No	Yes (plugins)
Cross-chain native	No	Partially	No	Yes (CCTP V2)
Compliance-compatible	Yes (manual)	Difficult	No	Yes (encrypted verification)
Non-custodial	No	Yes	Yes	Yes
Insurance / disputes	Manual	No	No	Native (on-chain)
Operator network	N/A	No	No	Yes (staked relay)
Time to integrate	Weeks	Days	N/A	Minutes (two interfaces)

---

8. Implementation status

Component	Status	Notes
Confidential Tokens (cUSDC)	**Deployed**	Arbitrum Sepolia
Escrow Engine	**Deployed**	Core + extensions + unwrap
Cross-Chain Settlement	**Deployed**	CCTP V2 receiver + handler
Orchestration Network	**Deployed**	Registry, executor, fees, slashing
Insurance Protocol	**Deployed**	Coverage manager, pools, policies, factory
Off-Chain Operator Services	**Running**	Coordinator, operator agent, CLI
Plugin Examples	**Working**	SimpleCondition, SimplePolicy
Agentic Automation	*TODO*	Autonomous on-chain workflow agents
Compliance Plugins	*TODO*	KYC/KYB/sanctions verification plugins
zkTLS Condition Resolvers	*TODO*	Reclaim Protocol integration for Web2 verification
Multi-chain Expansion	*TODO*	Additional CCTP-supported source chains
SDK	*TODO*	TypeScript SDK for application developers
Mainnet Deployment	*TODO*	Pending security audit

---

9. Roadmap

Phase 1 – Settlement core *(complete)*

Confidential tokens, escrow engine, cross-chain settlement, operator network. The foundation: encrypted money that moves programmably across chains.

Phase 2 – Economic protection *(complete)*

Insurance protocol with coverage pools, pluggable underwriter policies, and automated dispute resolution. Economic backstop for settlement disputes.

Phase 3 – Verification ecosystem *(in progress)*

Condition resolver and underwriter policy plugins: zkTLS verification, oracle integration, compliance checks, FHE quorum voting. The ecosystem that makes the settlement engine useful for real-world scenarios.

Phase 4 – Production readiness

Security audits, mainnet deployment, SDK release, multi-chain expansion, progressive decentralization of governance.

ReineiraOS Litepaper – Version 0.1.0 – March 2026