ANASTASIA LABS

VIA Cardano Smart Contracts Security Audit

Date: August 5, 2026
Project: VIA Cross-Chain Messaging Protocol — Cardano Contracts
Version: 1.1.0

Contents

Disclosure

Disclaimer and Scope

A code review represents a snapshot in time, and the findings and recommendations presented in this report reflect the information gathered during the assessment period. It is important to note that any modifications made outside of this timeframe will not be captured in this report.

While diligent efforts have been made to uncover potential vulnerabilities, it is essential to recognize that this assessment may not uncover all potential security issues in the protocol.

It is imperative to understand that the findings and recommendations provided in this audit report should not be construed as investment advice.

Furthermore, it is strongly recommended that projects consider undergoing multiple independent audits and/or participating in bug bounty programs to increase their protocol security.

The assessment is limited to the 30 Aiken source files enumerated by `scope-manifest.sha256` at revision `b10f038190e7991153c630a3f459317ccae586f8`. Findings are scoped exclusively to Cardano smart-contract behavior. TypeScript clients, transaction builders, deployment scripts, emulator infrastructure, generated `plutus.json`, other chains, and operational key management are excluded from reportable findings. Off-chain code was consulted only to understand transaction shapes and deployment assumptions. Explicit test modules, test-only validators, `always_fails.ak`, `reference_validator.ak`, and the redeemer-indexing harness are excluded. No mainnet deployment was exercised.

Assessment overview

Anastasia Labs performed a line-referenced security assessment of VIA's Cardano Aiken contracts at the pinned revision. The review covered all deployable minting policies and spending validators, including both reference integration clients and the Midnight-compatible lock/release variant, plus every transitive on-chain library.

The work used an eUTxO-specific context-closure process before finding discovery. It mapped validator purposes, datum state machines, singleton assets, reference inputs, redeemer-list coupling, value movement, authorization boundaries, and cross-script transaction composition. Potential issues were retained only when a concrete accepted transaction shape and line-referenced validation path could be shown.

Phases included:

- **Planning** — pin the revision, lock scope, and identify specifications.
- **Context** — map all entrypoints, branches, state nodes, control assets, and trust boundaries.
- **Discovery** — sweep Cardano-native datum, address, double-satisfaction, minting, value, authorization, replay, arithmetic, and state-management risks.
- **Validation** — execute the complete Aiken suite and focused audit regressions.
- **Reporting** — separate confirmed findings from deployment assumptions and defense-in-depth notes.

Assessment components

Manual revision

Our manual code auditing is focused on a wide range of attack vectors, including but not limited to:

- Datum identity, inline datum decoding, and state-transition integrity
- Script address binding, staking-credential ambiguity, and output selection
- Double satisfaction and cross-script transaction composition
- Singleton NFT continuity, mint/burn cardinality, and token confinement
- Value preservation, destination accounting, and liquidity extraction
- Owner, admin, relayer, project, and cryptographic signer authorization
- Replay-list ordering, duplicate insertion, and message-domain binding
- Redeemer-list indexing and recipient-script participation
- Integer bounds, amount serialization, thresholds, and fee arithmetic
- State contention, unbounded data, configuration safety, and upgrades

Executive summary

Protocol architecture

The Cardano system implements shared cross-chain messaging infrastructure and two reference asset integration models. A gateway singleton stores chain-wide signer, relayer, fee, pause, and replay-policy configuration. An ordered project registry stores per-recipient relayers, project signers, thresholds, and fee overrides. Outbound requests are represented by authenticated `send_request` UTxOs and assigned monotonic identifiers through a nonce-tracker singleton. Inbound messages are signature-checked and inserted into an ordered replay list before the recipient script may mint or release assets.

The mint/burn reference client burns local assets for outbound messages and mints them on authenticated delivery. The lock/release reference client moves a fixed asset through lane NFTs and a mutable source-route allowlist. Its Midnight variant additionally rejects outbound amounts above `2^64 - 1`.

Result

At the pinned revision, we confirmed **one acknowledged Minor finding**. No Critical, Major, Medium, or Informational finding met the final evidence threshold. The issue permits negative VIA and chain signature thresholds at gateway genesis and during owner updates. Because signature counts are compared directly against those thresholds, a negative value makes the corresponding layer pass with zero signatures.

The immediate impact depends on privileged misconfiguration: the gateway seed holder or current owner must first install the invalid threshold. Once present, an unprivileged submitter can omit those signature layers; if the project layer and relayer gate are also configured permissively, forged inbound delivery can reach a registered recipient's mint or release branch. The issue is therefore rated Minor under the template taxonomy despite the downstream asset impact.

The report also preserves **five historical findings that were resolved before the audited revision**: three Major, one Medium, and one Informational. These cover insert-only replay consumption, missing source-route binding in the mint/burn client, cross-chain replay-key collisions, destination checks delegated to integrations, and pause semantics during outbound request creation. Historical items are not counted as current vulnerabilities in the release recommendation.

Assurance notes

The review retained three non-finding operational assumptions. Replay genesis placement is not bound to the replay spending validator. The declared `max_lane_count` is guidance rather than an enforced bound. Several state transitions intentionally preserve non-lovelace value while allowing lovelace to vary; deployments should avoid accumulating surplus ADA at shared state UTxOs. These conditions are documented and do not provide an independently reachable unprivileged exploit at the reviewed revision.

Build and validation status

`aiken check` completed with **151 of 151 tests passing**. Two temporary audit-only tests also passed: one constructed gateway genesis with negative VIA and chain thresholds, and one demonstrated that zero supplied signatures satisfy both layers. The temporary fixtures were removed after execution, leaving the source tree unchanged. The report itself was compiled with Typst from the Anastasia Labs audit template.

Five permanent regressions corresponding to the historical findings were also re-run individually during report revision and passed.

Release recommendation

Enforce `0 <= required <= length(signers)` both at gateway creation and in both signer-update branches. Validate the entire genesis `GatewayDatum`, including non-negative fees, canonical signer/relayer arrays, and explicit protocol rules for whether a zero threshold is permitted. Add permanent negative-threshold regressions before mainnet deployment.

Code base

Repository

`https://github.com/Anastasia-Labs/via-drivers-cardano`

Commit

`b10f038190e7991153c630a3f459317ccae586f8`

Files audited

SHA256 Checksum	Files
`8d8a8a8053b25339da3f0abd9e7720f3881ffcb4935706659ba65525d52072256`	`sha256 — 30 Aiken files`

Severity Classification

- **Critical**: This vulnerability has the potential to result in significant financial losses to the protocol. They often enable attackers to directly steal assets from contracts or users, or permanently lock funds within the contract.
- **Major**: Can lead to damage to the user or protocol, although the impact may be restricted to specific functionalities or temporal control. Attackers exploiting major vulnerabilities may cause harm or disrupt certain aspects of the protocol.
- **Medium**: May not directly result in financial losses, but they can temporarily impair the protocol's functionality. Examples include susceptibility to front-running attacks, which can undermine the integrity of transactions.
- **Minor**: Minor vulnerabilities do not typically result in financial losses or significant harm to users or the protocol. The attack vector may be inconsequential or the attacker's incentive to exploit it may be minimal.
- **Informational**: These findings do not pose immediate financial risks. These may include protocol optimizations, code style recommendations, alignment with naming conventions, overall contract design suggestions, and documentation discrepancies between the code and protocol specifications.

Finding severity ratings

The following table defines levels of severity and score range that are used throughout the document to assess vulnerability and risk impact.

Level	Severity	Findings
5	Critical	0
4	Major	3
3	Medium	1
2	Minor	1
1	Informational	1

Findings

ID	Title	Severity	Status
VIA-CAR-H01	Replay identities could be consumed without executing the recipient	Major	Resolved
VIA-CAR-H02	Mint/burn delivery did not bind the authenticated source route	Major	Resolved
VIA-CAR-H03	Cross-chain replay-key collision	Major	Resolved
VIA-CAR-H04	Destination-chain binding was delegated to recipient contracts	Medium	Resolved
VIA-CAR-001	Negative gateway signature thresholds disable security layers	Minor	Acknowledged
VIA-CAR-H05	Outbound request creation ignored the gateway pause state	Informational	Resolved

VIA-CAR-H01 Replay identities could be consumed without executing the recipient

Severity	Status
Major	Resolved

Description

A previously reviewed `ViaTxRedeemer.Insert` branch authenticated an inbound message and inserted its replay identity without proving that the recipient contract executed in the same transaction. Any submitter who obtained a valid signed envelope could front-run delivery with an insert-only transaction. The replay list would then permanently reject the legitimate delivery of that envelope.

Violated invariant: consuming a replay identity and executing the authenticated recipient action must be atomic.

Adversarial transaction shape: consume a covering replay node, mint the new replay token, emit the valid continuation and inserted nodes, and supply the valid gateway, registry, relayer, and signature evidence, but omit every witness for the envelope's `recipient`. The earlier policy accepted this shape and irreversibly consumed the identity.

Historical evidence: parent revision `263f4728f5cbeffb04d01c26b34a0582bb378182` validates the complete Insert path without any recipient-participation condition. The remediation is commit `3f58f81b4a316f1392ce3b436af9b769498b148b`.

Current controls: `recipient_script_participates` resolves Mint, Spend, or Withdraw purposes to the authenticated recipient and excludes the replay policy's own witness (`minting_via_tx.ak:78-96, 613-616`). Non-delivery branches in both reference clients reject replay-policy activity, preventing a recipient script's unrelated branch from satisfying the atomicity proof.

Recommendation

Retain the insert-only regression and the non-delivery branch exclusions. Every future recipient integration should bind its delivery branch to the exact VIA Insert redeemer and reject replay activity in all other branches.

Resolution

Resolved in `3f58f81b`. The replay policy now requires a distinct witness belonging to the authenticated recipient. A permanent Aiken regression, `via_tx_insert_rejects_insert_only_transaction`, and an emulator regression cover the original transaction shape.

VIA-CAR-H02 Mint/burn delivery did not bind the authenticated source route

Severity	Status
Major	Resolved

Description

During pre-commit review, the mint/burn reference client accepted an inbound VIA envelope based on its signatures, destination recipient, and payload but did not require the envelope's `(source_chain, sender)` pair to belong to an authorized remote token contract. A message emitted by an unrelated source contract could therefore target the Cardano mint policy and request supply creation if it obtained the normal VIA approvals.

Violated invariant: only explicitly paired remote contracts may authorize minting of the local bridged asset.

Adversarial transaction shape: submit a valid VIA Insert whose recipient is the mint/burn policy and whose payload requests an arbitrary positive amount, but whose source chain and sender describe an unrelated origin contract. The reviewed implementation had no local allowlist comparison.

Historical evidence: this issue was identified against the March 11, 2026 pre-commit working tree. Commit `071a4989486aa9f2c7ee7efc41ace4b370f0e389` introduced the route binding and its negative regression before the affected work was committed.

Current controls: `MintFromVia` reads the singleton mint/burn state and `route_is_supported` requires exact equality of both source chain and sender (`minting_client_test_v4.ak:391-415, 520-560`). State updates remain administrator-authorized.

Recommendation

Keep route state explicit and review every administrative route change as a security-sensitive asset-pairing operation. Preserve exact byte equality for sender identifiers; do not normalize or truncate them inside the validator.

Resolution

Resolved in `071a4989` and retained through the mutable-state refactor. `mint_from_via_rejects_unsupported_source_route` proves that an otherwise valid message from an unlisted chain/sender pair cannot mint.

VIA-CAR-H03 Cross-chain replay-key collision

Severity	Status
Major	Resolved

Description

Replay token names and ordered-list keys were previously derived from `keccak256(uint256(tx_id))`. If two source chains assigned the same message sequence number, processing either message consumed the single shared replay identity and permanently prevented the other valid message from executing.

Violated invariant: replay identity uniqueness must cover the full domain in which message identifiers are allocated.

Adversarial transaction shape: obtain two independently valid signed messages `(chain A, n)` and `(chain B, n)`. Execute the first normally. The second then cannot find an insertion interval because its hashed key and token name are identical to the first, despite the signed envelopes being distinct.

Historical evidence: parent revision `bac0ca1bce0794853d4bcfbe697e074c4a99e855` defines `hash_tx_id(tx_id)` and uses it for insertion. Commit `74db1538983d8fd94f365dac6cc27eb377f3dc7e` changed both token and list-key derivation.

Current controls: the policy hashes the unambiguous fixed-width encoding `uint64(source_chain) || uint256(tx_id)` and rejects negative values (`minting_via_tx.ak:477-505, 618-619`).

Recommendation

Treat the fixed-width replay encoding as a protocol constant across all on-chain and off-chain implementations. Any future format change requires a migration that preserves every historical identity.

Resolution

Resolved in `74db1538`. The permanent regression `replay_identity_hash_distinguishes_source_chain` proves that equal sequence numbers on distinct source chains produce different identities.

VIA-CAR-H04 Destination-chain binding was delegated to recipient contracts

Severity	Status
Medium	Resolved

Description

The shared VIA replay policy originally authenticated message signatures and replay insertion without comparing `dest_chain` to the local gateway chain. The bundled reference clients performed their own comparison, but a new integration that omitted it could accept a valid message intended for a different destination and consume its replay identity on Cardano.

Violated invariant: a shared message-processing policy must reject every envelope whose destination domain is not the current chain.

Adversarial transaction shape: use a correctly signed envelope naming a different destination chain, target an integration that relies on the shared processing policy, and construct an otherwise valid replay insertion plus recipient action. Before the fix, the shared policy supplied no destination guarantee.

Historical evidence: revision `071a4989486aa9f2c7ee7efc41ace4b370f0e389` lacked the shared comparison. Commit `50765cb768dd7458408cc13acea81a731b150336` added it.

Current controls: Insert requires `dest_chain == gateway_datum.chain_id` before replay mutation (`minting_via_tx.ak:600-619`).

Recommendation

Keep destination binding in the shared policy even when recipient contracts repeat the check. Shared validation should provide a complete authenticated envelope contract to integrations.

Resolution

Resolved in `50765cb7`. The permanent validator regression `via_tx_insert_rejects_wrong_destination_chain` exercises the rejected cross-domain Insert transaction.

VIA-CAR-001 Negative gateway signature thresholds disable security layers

Severity	Status
Minor	Acknowledged

Description

Gateway genesis only proves that the output datum decodes as `GatewayDatum`; it does not validate either signature threshold. The owner-only signer update paths enforce only `required <= list.length(signers)` and also accept negative values. Inbound processing accepts a layer when its valid-signature count is at least the threshold, so zero signatures satisfy every negative threshold.

Violated invariant: every VIA and chain signature threshold must be non-negative before it participates in message authorization.

Adversarial transaction shape:

1. The gateway seed holder initializes the singleton with `via_signers_required = -1` and/or `chain_signers_required = -1`, or the current gateway owner installs the same value through `SetViaSigners` or `SetChainSigners`.
2. An attacker submits `ViaTxRedeemer.Insert` with empty signature lists and an otherwise structurally valid replay-list transition.
3. `verify_signatures` returns zero for the empty list, and the affected comparison evaluates as `0 >= -1`.
4. If project signatures are configured with threshold zero and relayers are not required, the forged envelope may reach a registered recipient's mint-from-VIA or release-from-VIA branch.

Affected code: `contracts/cardano/validators/minting_message_gateway_v4.ak:32-40`, `contracts/cardano/validators/spending_message_gateway_v4.ak:83-103`, `contracts/cardano/validators/spending_message_gateway_v4.ak:235-259`, and `contracts/cardano/validators/minting_via_tx.ak:393-449`.

Evidence: two focused Aiken tests passed. The first initialized the gateway with both thresholds set to `-1`; the second called the security-layer check with no VIA, chain, or project signatures and observed acceptance.

Recommendation

Enforce one shared gateway-datum invariant at genesis and every update. Require `0 <= required && required <= list.length(signers)` for both signer sets, and validate non-negative fees, canonical signer/relayer arrays, and any required minimum threshold. Add permanent genesis, update, and empty-signature tests.

Resolution

Acknowledged by the project team. The finding remains present at the audited revision. The repository documentation identifies non-negative thresholds as an operational requirement, but the requirement is not enforced by the reviewed on-chain code.

VIA-CAR-H05 Outbound request creation ignored the gateway pause state

Severity	Status
Informational	Resolved

Description

Source-side request minting previously permitted users to lock or burn assets while `gateway_datum.system_enabled` was false. Later indexing and delivery remained paused, so this did not bypass message authorization or permit asset theft, but it allowed users to enter a workflow that could not progress until an administrator re-enabled the gateway.

Operational transaction shape: mint a valid `send_request` and execute the client lock/burn branch while referencing a disabled gateway. The request remained pending because downstream processing enforced the pause.

Historical evidence: the March 11, 2026 review found the missing gate in shared request creation and both reference integrations. Commit `071a4989486aa9f2c7ee7efc41ace4b370f0e389` aligned request creation with the downstream circuit breaker.

Current controls: `validate_mint_send_request` resolves the authoritative gateway datum and rejects a disabled system before fee validation (`send_request.ak:171-191, 205-239`).

Recommendation

Preserve the pause check in shared request minting and document whether any future pause mode intentionally permits queueing. If queueing is desired, it should be an explicit state rather than an accidental inconsistency.

Resolution

Resolved in `071a4989`. The permanent regression `mint_send_request_rejects_when_gateway_disabled` covers the original shape.