Formal Verification Report of Aave Delivery Infrastructure

Summary

This document describes the specification and verification of Aave's Delivery Infrastructure (a.DI) using the Certora Prover. The work was undertaken from August 6th to September 18th.

The scope of our verification includes the following contracts:

- CrossChainForwarder.sol
- CrossChainReceiver.sol
- BaseCrossChainController.sol
- CrossChainController.sol
- CrossChainControllerWithEmergencyMode.sol

The Certora Prover proved the implementation correct with respect to the formal rules written by the Certora team. During verification, the Certora Prover discovered bugs in the code which are listed in the tables below. All issues were promptly addressed. The fixes were verified to satisfy the specifications up to the limitations of the Certora Prover. The following section formally defines the high-level specifications of a.DI.

List of Main Issues Discovered

Severity: Low

Issue:	Non-trivial behavior of the receiver contract
Rules Broken:	property #23.

Issue:	Non-trivial behavior of the receiver contract
Description:	`crossChainReceiver` is able to reach a state where the `allowedBridgeAdapters` is nonempty, yet the corresponding `requiredConfirmation` is left uninitialized. In such a state, `receiveCrossChainMessage()` confirms every message on a first try since `newConfirmations >= configuration.requiredConfirmation` always holds. This uninitialized state is equivalent to configuring `newConfirmations = 1`, however this is a non-trivial behavior of the system that can cause havoc in case of misconfigurations. If the owner desires to allow the first bridge that confirms the message to deliver, a more explicit approach is preferred in setting `requiredConfirmation==1`.
Example	Consider the following 2 scenarios: **scenario A**: `CrossChainReceiver.constructor()` and `CrossChainController.initialize()` leave both `allowedBridgeAdapters` empty and `requiredConfirmation==0`. Later, the owner calls `allowReceiverBridgeAdapters()` and adds a bridge but doesn't update `requiredConfirmation`. **scenario B**: `CrossChainReceiver.constructor()` or `CrossChainController.initialize()` are called with some nonempty `receiverBridgeAdaptersToAllow` but an empty `initialRequiredConfirmations`. The functions `_configureReceiverBasics()` and `allowedBridgeAdapters()` may leave `requiredConfirmation` uninitialized as they don't check that its value matches the content of `AllowedBridgeAdapters`.
Mitigation/Fix:	Fixed in PR#130.

Severity: Low

Issue:	Address 0 is accepted as a valid sender
Rules Broken:	Property #11.
Description:	Address 0 can be set as a valid sender, in violation of an explicit requirement of the CrossChainForwarder.
Mitigation/Fix:	Fixed in PR#132.

Disclaimer

The Certora Prover takes a contract as input and a specification and formally proves that the contract satisfies the specification in all scenarios. More importantly, the guarantees of the Certora Prover are scoped to the provided specification, so that it does not check any cases not covered by the specification.

Though we hope that this information is useful, we do not provide warranty of any kind, explicit or implied. The contents of this report should not be construed as a complete guarantee that the contract is secure in all dimensions. In no event shall Certora or any of its employees be liable for any claim, damages or other liability, whether in an action of contract, tort or otherwise, arising from, out of or in connection with the results reported here.

Assumptions and Simplifications Made During Verification

We made the following assumptions during our verification:

- We unroll loops. Violations that require executing a loop more than once will not be detected.
- We do not verify the cryptographic correctness of functions that involves calls to the `keccak256()` function.

CrossChainForwarder

- The contract sends out transactions by emitting them. Since the prover cannot currently access emit's data, we consider calls to `_bridgeTransaction(...)` as transactions that were sent out. We did not check the validity of that function.
- We assume that the contract being called through delegate call are not malicious.

Notations

✔️ indicates the rule is formally verified on the latest reviewed commit.

✔️* indicates that the rule is verified on the simplified assumptions described above in “Assumptions and Simplifications Made During Verification”.

❌ indicates that the rule was violated under one of the tested versions of the code.

⏱ indicates the rule is currently timing out and therefore was not proved and no violations were found.

Properties of a.DI

Below are the properties that were formally check using Certora's prover.

CrossChainForwarder

1. ✔ Only the approved senders can forward a message.
2. ✔ Internal transaction nonces are sequential.
3. ✔ A new Envelope should always be registered.
4. ✔ Forwarding a message should revert if no bridge adapters are registered for the destination chain.
5. ✔ An Envelope can only be retried if it has been previously registered.
6. ✔ An Envelope retry should be on a new Transaction.
7. ✔ A Transaction can only be retried if it has been previously forwarded.
8. ✔ Only the Owner can enable/disable authorized senders.
9. ✔ Only the Owner can enable/disable bridge adapters.
10. ✔ An adapter can not be the address 0.
11. ✔ A sender can not be the address 0. ❌ - found issues in previous commits.

CrossChainReceiver

12. ✔ A Transaction can only be received from authorized bridge adapters.
13. ✔ Only the Owner can set the receiver's bridge adapters.
14. ✔ Only the Owner can set the required confirmations.
15. In order to forward a received Envelope to the final address destination, the envelop needs to be received at least `_requiredConfirmations`. 15.1 ✔ If an envelope has `requiredConfirmations - n` confirmations, then `n` different adaptors must call `receiveCrossChainMessage` in order to change the enevlope's state (checked for `n=2`).

16. An Envelope should be delivered to destination only once. 16.1. ✔️ If envelope has changed state to `Delivered` then `IBaseReceiverPortal.receiveCrossChainMessage()` was called exactly once. 16.2. ✔️ `CrossChainReceiver.receiveCrossChainMessage()` cannot call `IBaseReceiverPortal.receiveCrossChainMessage()` twice with the same envelope.
17. ✔️ An envelope cannot be delivered twice by `deliverEnvelope()`.
18. ✔️ `DeliverEnvelope()` can be triggered by anyone.
19. ✔️ If the bridge is configured such that no confirmations are needed, then no envelope can change its state.
20. ✔️ An envelope state can only go in a single direction: None → confirmed → Delivered.
21. ✔️ `internalTransaction.confirmations` grows by 1 iff `bridgedByAdapter[address]` changes from false to true.
22. ✔️ for an already allowed bridge, `allowReceiverBridgeAdapters()` should not change the bridge's state.
23. ✔️ If there are `allowedBridges` configured, `requiredConfirmation` must be configured to a positive value. ❌ - found issues in previous commits.
24. ✔️ While a transaction is not confirmed yet, every changing-state call to `receiveCrossChainMessage()` increments the confirmation counter by 1.
25. ✔️ If a message was received but not confirmed (# of confirmation did not reach the required conformations) and later `updateMessagesValidityTimestamp()` invalidated its timestamp then it cannot change its state.
26. ✔️ `receiveCrossChainMessage` cannot change envelope state if `requiredConfirmation` is zero.

CrossChainControllerWithEmergencyMode

27. ✔️ For all methods of `CrossChainReceiver`, only Owner and Guardian can invalidate Envelopes.