certora

Security Assessment & Formal Verification Final Report

Liquity

Bold

December 2024

Prepared for Liquity

Table of content

Project Summary 3
Project Scope 3
Project Overview 3
Protocol Overview 3
Formal Verification 4
Verification Notations 4
General Assumptions and Simplifications 4
Summarization 4
Formal Verification Properties 5
BorrowerOperations 5
P-01. Sum of the trove debts within a batch equals the total batch debt 6
P-02. Batch troves have the same interest rate as the batch 6
P-03. Troves in a batch always accrue interest at the same rate 7
P-04. When a trove is in a batch, it uses the batch for its debt calculation 7
P-05. Troves in a batch share the same upfront fee 7
P-06. Troves in a batch share the same management fee 8
P-07. Batch trove collateral adjustment effects 8
P-08. Batch trove debt adjustment effects 9
P-10. Trove and Batch Trove Equivalences – Actions other than debt/col changes 10
P-11. Trove and Batch Trove Equivalences – Debt/col change actions 11
Disclaimer 12
About Certora 13

Project Summary

Project Scope

Project Name	Repository (link)	Latest Commit Hash	Platform
BOLD		{Hash}	EVM

Project Overview

This document describes the specification and verification of BOLD using the Certora Prover. The work was undertaken from September 9, 2024 to November 22, 2024.

The following contract list is included in our scope:

```
contracts/src/{BorrowerOperations.sol, TroveManager.sol, ActivePool.sol}
```

The Certora Prover demonstrated that the implementation of the Solidity contracts above is correct with respect to the formal rules written by the Certora team based on the inputs of the Liquity team.

Protocol Overview

Liquity V2 is a collateralized lending protocol wherein users may add collateral to positions called Troves. Crucially, Liquity V2 offers a feature that permits Trove owners to delegate management of the Trove to a Batch. Batches may manage a collection of Troves on behalf of multiple users. A core focal point of this verification effort was demonstrating that individual Troves (without this delegation) are observationally equivalent to Troves in batches.

Formal Verification

Verification Notations

Formally Verified	The rule is verified for every state of the contract(s), under the assumptions of the scope/requirements in the rule.
Formally Verified After Fix	The rule was violated due to an issue in the code and was successfully verified after fixing the issue
Violated	A counter-example exists that violates one of the assertions of the rule.

Verification Methodology

Verification of the Liquity protocol was done using the Certora verification tool which is based on Satisfiability Modulo Theories (SMT). In short, the Certora verification tool works by compiling formal specifications written in the Certora Verification Language (CVL) and Liquity's implementation source code written in Solidity. More information about Certora's tooling can be found in the Certora Technology Whitepaper.

If a property is verified with this methodology it means the specification in CVL holds for all possible inputs. However specifications must introduce assumptions to rule out situations which are impossible in realistic scenarios (e.g. to specify the valid range for an input parameter). Additionally, SMT-based verification is notoriously computationally difficult. As a result, we introduce overapproximations (replacing real computations with broader ranges of values) and underapproximations (replacing real computations with fewer values) to make verification feasible.

General Assumptions and Simplifications

Summarization

For verification performance reasons, we introduce abstractions that represent the following functions as returning an arbitrary value on each invocation and having no side-effects:

- `SafeERC20._callOptionalReturn`
- `SortedTroves._findInsertPosition`
- `StabilityPool._computeCollRewardsPerUnitStaked`
- `TroveManager._sendGasCompensation`
- `TroveManager._computeNewStake`
- `fetchPrice`

In addition, we model `BorrowerOperations._calcUpfrontFee` abstractly as an unconstrained function (repeated invocations of calcUpfrontFee with the same parameter values return the same value, but the return value is otherwise arbitrary).

We also model `SafeERC20.safeTransfer` and `SafeERC20.safeTransferFrom` using our own (straightforward) stateful abstractions for SafeERC20.

Formal Verification Properties

BorrowerOperations

Module General Assumptions

- The TroveManager is not shutdown (i.e. TroveManager.shutdownTime == 0)
- Reward snapshots are up to date. That is,
  - `troveManager.rewardSnapshots[_troveId].coll == troveManager.L_coll && troveManager.rewardSnapshots[_troveId].boldDebt == troveManager.L_boldDebt`;

NOTE: This assumption that rewards snapshots are always updated is **not** strictly true as, for example, the rewards snapshots are not updated when batch trove interest rates are adjusted. However, without this assumption there will be changes in the debt updated due to redistribution gains which will: 1) contradict the debt change aspect P-08, P-09, although redistribution gains are beyond the intent of the property and 2) cause differences between batch and individual troves which will cause counterexamples in P-10, however this is also out of scope for the intent of the property.

- `troveManager.batches[batchAddress].debt` **equals** `batchData.entireDebtWithoutRedistribution`
- The two storage locations where batch manager addresses are stored are equal:
  - `troveManager.Troves[troveId].interestBatchManager == borrowerOperations.interestBatchManagerOf[troveId];`
- We assume the following relationships between batch shares, batch debt, and trove shares within any given batch:
  - The total batch shares is zero if and only if the total batch debt is zero
  - A batch trove's total debt is zero if and only if it's batch shares is zero
  - The batch total shares is greater or equal the shares of any trove belonging to the batch

Note: Any assumptions in tables for specific rules apply only to the rule in that table.

Module Properties

P-01. Sum of the trove debts within a batch equals the total batch debt.	
Status: Verified	Assumptions: ● For tool performance reasons, we restrict to the case where a batch has exactly 2 troves in it. We expect this property is true for the general case (>2 troves) as well.

Rule Name	Status	Description	Link to rule report
sum_of_trove_debts	Verified	*Sum of a given batch's individual Trove entire debts sans redistributions (recorded debts + accrued interest) equals the batch's recorded debt plus its accrued interest plus management fees.* *(We use modular verification to establish that batch troves have the same Annual Interest Rate (AIR) as the batch they belong to. That is we assume his for this rule and prove it is true separately)*	[Report]

P-02. Batch troves have the same interest rate as the batch

Status: Verified

Rule Name	Status	Description	Link to rule report
sameInterestRateForBatchTroves	Verified	*We proved this so it can be used for modular verification. (See note in P-01)*	*Report*

P-03. Troves in a batch always accrue interest at the same rate

Status: Verified

Rule Name	Status	Description	Link to rule report
troves_in_batch_accrue_interest_at_same_rate	Verified	*Any two troves in the same batch will accrue interest at the same rate.*	*Report*

P-04. When a trove is in a batch, it uses the batch for its debt calculation

Status: Verified

Rule Name	Status	Description	Link to rule report
troves_in_batch_use_batch_structure	Verified	*When a trove is a member of a batch its recorded debt is calculated as the batch debt normalized by its fraction of the total shares.*	Report

P-05. Troves in a batch share the same upfront fee

Status: Verified

Rule Name	Status	Description	Link to rule report
troves_in_batch_share_upfront_fee	Verified	*For a given average system interest rate, Troves in a given batch always pay the same upfront fee (as percentage of their debt) upon premature interest rate adjustments by the manager*	Report

P-06. Troves in a batch share the same management fee

Status: Verified

Rule Name	Status	Description	Link to rule report
troves_in_batch_share_management_fee	Verified	*Troves in a given batch are charged the same management fee (as percentage of their debt)*	[Report]

P-07. Batch trove collateral adjustment effects

Status: Verified

Rule Name	Status	Description	Link to rule report
troves_in_batch_share_management_fee	Verified	*When any borrower with a batch Trove i adjusts its coll by x:* *-All of the batch's accrued interest is applied to the batch's recorded debt* *-All of the batch's accrued management fee is applied to the batch's recorded debt* *-Trove i's pending redistribution debt gain is applied to the batch's recorded debt* *-Trove i's pending redistribution coll gain is applied to the batch's recorded coll* *-Trove i's entire coll changes only by x* *-Trove i's entire debt does not change*	[Report]

P-08. Batch trove debt adjustment effects	
Status: Verified	Just for the two bullet points about debt changes: we assume the total number of batch debt shares is a scalar multiple of the total batch debt and that the scalar is in the set [1, 5] U {1e9, 1e9-5} where 1e9 is the maximum total debt shares in the system. The motivation for these scalars is that usually the number of debt shares should be close to the total shares (the range 1-5 covers this common case), and we also cover the extreme case and a number slightly below the maximum

Rule Name	Status	Description	Link to rule report
debt_adjust_effects	Verified	*When any borrower with a batch Trove i adjusts its debt by x:* *-All of the batch's accrued interest is applied to the batch's recorded debt* *-All of the batch's accrued management fee is applied to the batch's recorded debt* *-Trove i's pending redistribution debt gain is applied to the batch's recorded debt* *-Trove i's pending redistribution coll gain is applied to the batch's recorded coll* *-Trove i's debt change x is applied to the batch's recorded debt* *-Trove i's entire coll does not change*	Report
withdraw_debt_change	Verified	- Trove i's entire debt changes only by x (withdraw case)	Report
repay_debt_change	Verified	- Trove i's entire debt changes only by x (repay case)	Report

P-10. Trove and Batch Trove Equivalences – Actions other than debt/col changes	
Status: Verified	Assumptions: We assume the total number of batch debt shares is a scalar multiple of the total batch debt and that the scalar is in the set {1, 1e9} where 1e9 is the maximum total debt shares in the system. The motivation for these scalars is that usually the number of debt shares should be close to the total shares (the range 1-5 covers this common case), and we also cover the extreme case and a number slightly below the maximum

Rule Name	Status	Description	Link to rule report
trove_batch_trove_eq_closing	Verified	*For individual Trove i and batch Trove j in the same branch where i != j, and i and j have equivalence E:* *E:(recorded_coll_i,recorded_debt_i, accrued_interest_i, stake_i, redistribution_debt_gain_i, redistribution_coll_gain_i) = (recorded_coll_j, recorded_debt_j, accrued_interest_j, stake_j, redistribution_debt_gain_j, redistribution_coll_gain_j)* *Then, the following pairs of simultaneous actions maintain equivalence E:* *-Closing Trove i and closing batch Trove j*	Report
trove_batch_trove_eq_redeem	Verified	*...* *-Redemption from Trove i and from batch Trove j of the same BOLD amount*	Report
trove_batch_trove_apply_pending_debt	Verified	*...* *-Applying pending debt to Trove i and to Trove j's batch*	Report

trove_batch_trove_eq_liquidate	Verified	… *-Liquidation of another Trove k in the same branch by redistribution (only when stake_i = stake_j)* *-Liquidation of another Trove k by SP offset*	*Report*

P-11. Trove and Batch Trove Equivalences – Debt/col change actions	
Status: Verified	Assumptions: These assumptions are introduced because otherwise batch and non-batch trove executions will differ: ● Batch annualManagementFee == 0 ● Annual interest rate for both troves are equal ● The non-batch trove's redistDebtGain equals the batch trove's redistDebtGain scaled by its portion of the total debt shares This assumption is introduced for performance reasons, just for proving the equivalence holds for debt values after debt change actions ● The total number of batch shares is a scalar multiple of the total batch debt and the number of shares is in {1, 1e9}. (See P-08 for the rationale behind these scalars). Assumptions / Notes about calcInterest: For performance reasons we need to modify the Liquity code so that the calculation done by calcInterest is inlined for proving the equivalence for recordedDebt. For collateral and stake we underapproximate calcInterest assuming it always returns either 0 or 1 (in separate report links). We note that this restriction is only on newly added interest during the function call and not on interest already accumulated before the call. We also note this restriction only applies to this rule and so calcInterest is covered by other properties, namely P-05, P-07, P-08

Rule Name	Status	Description	Link to rule report
troveBatchTroveEquivalenceAddColl_{recordedDebt, gainsCol, stake}	Verified	*For individual Trove i and batch Trove j in the same branch where i != j, and i and j have equivalence E:* *E = (recorded_debt_i, entire_coll_i, stake_i)* *Then, the following pairs of simultaneous actions maintain equivalence E:* *-Adding the same amount of collateral to Trove i and batch Trove j*	Debt: *Report* Collateral: *Report* *Report* Stake: *Report* *Report*
troveBatchTroveEquivalenceWithdrawColl_{recordedDebt, gainsCol, stake}	Verified	*…* *-Withdrawing the same amount of collateral to Trove i and batch Trove j*	Debt: *Report* Collateral *Report* Report Stake: *Report* *Report*
troveBatchTroveEquivalenceWithdrawBold_{recordedDebt, gainsCol, stake}	Verified	*…* *- Withdrawing the same amount of bold to Trove i and batch Trove j*	*Debt, scalar=1, 1e9* *Report* *Collateral* *Report* *Report* *Stake* *Report* *Report*
troveBatchTroveEquivalenceRepayBold_{r recordedDebt, gainsCol, stake}	Verified	*…* *- Repaying the same amount of bold to Trove i and batch Trove j*	*Debt, scalar=1, 1e9* *Report* *Collateral* *Report* *Report* *Stake* *Report* *Report*

Disclaimer

Even though we hope this information is helpful, we provide no 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.

About Certora

Certora is a Web3 security company that provides industry-leading formal verification tools and smart contract audits. Certora's flagship security product, Certora Prover, is a unique SaaS product that automatically locates even the most rare & hard-to-find bugs on your smart contracts or mathematically proves their absence. The Certora Prover plugs into your standard deployment pipeline. It is helpful for smart contract developers and security researchers during auditing and bug bounties.

Certora also provides services such as auditing, formal verification projects, and incident response.