Code Review and Security Assessment
For
MACI/QFI

Initial Delivery: July 15, 2022
Final Delivery: September 30, 2022

Prepared For
Cory Dickson | Ethereum Foundation
Barry Whitehat | Ethereum Foundation
Chao Ma | Ethereum Foundation
TheF3llowship | Ethereum Foundation

Prepared by
Er-Cheng Tang | HashCloak Inc
Mikerah Quintyne-Collins | HashCloak Inc

Table Of Contents

Executive Summary **4**

Overview **5**

Methodology **5**

Claim Validity **5**

Findings **6**

Data are not fully verified during state update 6

Unconstrained variables 6

Doesn’t take into account ERC20s with blacklisting abilities 7

Token for top-up is a free resource 7

Integer overflow problem & improper bit length restrictions 7

Missing re-initialization 8

MessageQueue in PollFactory is uninitialized 9

Doesn’t take into account ERC20s that are deflationary or charge a fee upon transfer 9

Transactions do not revert when there aren’t enough funds 9

Inconsistent restriction on voice credit upper bound 10

Redundant functions and redundant checks 10

Potential for Re-entrancy in QFI 11

Potential for Re-entrancy in MACI 11

Potential for Re-entrancy in Poll 12

Return values of transferFrom calls are not checked 12

Miscellaneous mistakes 12

The QFI owner is powerful enough to decide the payout distribution 13

Subsidy functionality is incomplete 13

Incorrect error messages 13

Return value of ExtContracts.maci.mergeStateAq is ignored in Poll contract 14

Anyone can pass mismatched length encryption keys and message batches to PublishMessageBatch 14

Naming conflict with transferMatchingFunds in FundsManager and GrantFactory 14

isAfterDeadline, topup and publishMessage in Poll rely on the block timestamp 15

Boolean conditions in require statements can be simplified 15
State variables are incremented within a loop in AccQueue.mergeSubRoots 15
_decimals in TopUpCredit and stateTreeDepth in MACI are not constant 15
ERROR_VK_NOT_SET and ERROR_SB_COMMITMENT_NOT_SET should be moved to PollProcessorAndTallyer 16

General Recommendations 16

Set public functions to external if they have no internal calls 16
Use consistent Solidity versions across files 16
Do multiplications before divisions 16
Check return values of external calls 17

Executive Summary

The Ethereum Foundation's Applied ZKPs team engaged HashCloak Inc for an audit of MACI and QFI which are voting and funding related infrastructures. The audit was done by 2 auditors from July 4, 2022 to July 18, 2022. The relevant codebase was the MACI and the QFI repositories, assessed at commits 749eec…319366 and eab14a…3e7b95 respectively. During the first week we familiarized ourselves with the underlying design and the codebase of MACI and QFI. In the following weeks we investigated the security of the codebase through various efforts. From September 26, 2022 to September 30, 2022, we re-assessed the codebases at the following commits and pull requests:

- https://github.com/privacy-scaling-explorations/maci/pull/522
- https://github.com/privacy-scaling-explorations/maci/pull/523
- https://github.com/quadratic-funding/qfi/commit/b0777ec15ec1165adbb01971768f348797c4fc9e

All of the issues we identified have been rectified as of those commits and pull requests or are slated to be fixed in a future release of MACI or the QFI repositories.

We found several issues ranging from critical to informational during the audit.

Severity	Number of Findings
Critical	3
High	5
Medium	1
Low	7
Informational	10

Overview

Minimal Anti-Collusion Infrastructure (MACI) aims to provide a quadratic voting system that disincentives collusion behavior. It is deployed as Ethereum smart contracts where users can sign up and make anonymous votes. The work of processing these votes is delegated to a coordinator, who has the privilege to read the encrypted votes and has to prove its correct execution to the smart contract.

MACI is now updated to support 2 additional features. First, users can top-up their voice credits as they engage in various polls. Second, the coordinator is in charge of subsidy calculation, which could be verified in smart contracts. These changes would give MACI wider applicabilities and enhanced guarantees.

Quadratic Funding Infrastructure (QFI) is an application of MACI to the crowd funding scenario. Users obtain voice credits by transferring tokens to the grant pool, and they place their votes using MACI. At the end of each grant round, the grant receivers can claim funds from the grant pool. The amount one can claim is in proportion to the square of the total votes that one gets, which explains the name quadratic funding.

Methodology

We checked through common Ethereum vulnerabilities manually according to __ , and we ran the __ analyzer to aid our analysis. We further investigated potential attack surfaces based on our hacking experience.

Claim Validity

Our investigation shows that the following properties made in the MACI docs are valid:

- Collusion resistance
- Receipt freeness
- Privacy
- Uncensorability

Findings

Data are not fully verified during state update

Type: Critical
Files affected: maci/circuits/processMessages.circom

Description: Among the goals of MACI is ensuring correct execution by the coordinator. MACI uses zk-SNARK to verify the correctness of each execution step. The verification should check that the new state is the result of executing several user messages on the current state. We found that the verification in `processMessages.circom` is incomplete. In particular, `topupStateLeaves` and `topupStateLeavesPathElements` are never verified against the current state, while `topupStateIndexes` and `topupAmounts` are never verified against the message root. However, all of these fields will affect the new state through `ProcessTopup`.

Impact: A malicious coordinator can change the MACI state arbitrarily, meaning that it can change the voice credit and the voting public key of any user to any value. Many of the security claims of MACI would be violated because of this issue.
Suggestion: Assign the index and the amount with `msgs[i][1]` and `msgs[i][2]` in lines 332, 333. Assign the state leaf and the path elements with `currentStateLeaves` and `currentStateLeavesPathElements` in lines 336, 337.
Status: This has been rectified as of commit 6df6a4054da926b07f35c5befab4f1f8af33dcc6.

Unconstrained variables

Type: Critical
Files affected: qfi/contracts/GrantRound.sol, qfi/contracts/QFI.sol

Description: The value `_tallyCommitment` in the function `claimFunds` is provided by the caller. There is no guarantee that the caller will use the correct value; the variable should be replaced with `tallyHash` which is provided by the trusted coordinator. One can further assume less trust from the coordinator by getting the tally commitment from the `tallyCommitment` field of the `PollProcessorAndTallyer` contract. Likewise, the value `_alphaDenominator` in the function `finalize` is provided by the QFI owner, which is only assumed to be correct without verification.

Impact: A malicious recipient can claim an arbitrary amount of tokens from a grant round by calling `claimFunds` using wrong tally result, tally commitment, and tally proof.
Suggestion: Replace the parameter `_tallyCommitment` with the state variable `tallyHash`, or get the value from `PollProcessorAndTallyer.tallyCommitment()`. Make the `alpha` calculation part of the contract logic to ensure correctness.
Status: This issue has been acknowledged by the development team and will be included in a future release.

Doesn't take into account ERC20s with blacklisting abilities

Type: Critical
Files affected: qfi/GrantRound.sol, qfi/QFI.sol
Description: In the QFI contract, an owner can choose to use any ERC20 token they want. As such, an owner can choose to use an ERC20 token such as USDC that has blacklisting capabilities. Once a particular address has been blacklisted, the funds owned by that address can no longer be moved. As such, it is possible that some of the funds within the GrantRound contract are stuck in the event of a blacklisting.

Impact: Funds controlled by the QFI contract can be frozen.
Suggestion: Document this risk to users who may want to support such tokens for example USDC or USDT.
Status: This issue has been acknowledged by the development team and they have provided extra clarity within their documentation around using ERC20s with blacklisting abilities.

Token for top-up is a free resource

Type: High
Files affected: maci/contracts/TopupCredit.sol

Description: Users can top up their voice credits with `TopupCredit` tokens. However, the airdrop functionality of `TopupCredit` allows anyone to receive an arbitrary amount of tokens by calling the function multiple times. As a result, users can receive unlimited voice credits for free.

Impact: The voting system would have no control over the distribution of voice credits.
Suggestion: Set access control over the functions `airdrop` and `airdropTo`, e.g. add the modifier `onlyOwner` to both functions.

Status: This has been rectified at commit ee0c8a6a654d136f95180e6728c9cec283c1659b.

Integer overflow problem & improper bit length restrictions

Type: High
Files affected: maci/circuits/circom/float.circom

Description: In circom, the largest number consists of `253` bits. The integer in line 16 can be as large as `2^(2n)`, but `n` is only required to be less than `253` in line 11. Hence, the circuit `IntegerDivision` might use an overflowing divisor, so that the output can be incorrect. Also, the use of `assert` keywords in circom does not contribute to verification constraints. Thus, the size assertions on `a`, `b` in lines 12, 13 does not actually prevent a malicious coordinator from using out-of-size values. A better approach is to actually write down circuits that verifies the bit length of the variables. We suggest that various bit-length checks in `float.circom` be made more carefully. The bit length bounds shall be clearly documented, and one shall check if the bounds are indeed satisfied in the application codes. For example, we found that the assertion `(b < 2**n)` in line 13 can be violated during subsidy calculation, since `n` is set as `64` and `b` is calculated from the votes which can be `127` bits. An honest coordinator will fail to update the subsidy commitment in this case. Moreover, the division by zero check is absent in `IntegerDivision`.

Impact: A malicious user can affect the honest coordinator's effort of calculating the subsidy in various ways: (1) make the result incorrect (2) intercept the calculation.
Suggestion: Ensure that the circuits actually put down bit length constraints instead of merely assertions. Ensure that the circuits check whether the divisor is non-zero. Make clear documentation on acceptable bit lengths. Ensure that the contracts will only use the circuits with values that satisfy the bit length constraints.
Status: This has been rectified as of commit c8eb37ad593ee671652f11458909df2a95db3581.

Missing re-initialization

Type: High
Files affected: qfi/contracts/QFI.sol

Description: QFI allows multiple rounds of grant funding through the function `acceptContributionsAndTopUpsBeforeNewRound`. During this process, the status of

`contributors` was not reinitialized. As a result, previous contributors are mistakenly regarded as contributors for the current round.

Impact: A malicious user can withdraw tokens from a grant round even if it did not contribute to the grant round.
Suggestion: Reinitialize `contributors` within the function `acceptContributionsAndTopUpsBeforeNewRound`.
Status: This has been rectified at commit https://github.com/qu55b512b4342aa060295dd58a543d4a079b8f6da7.

MessageQueue in PollFactory is uninitialized

Type: High
Files affected: maci/contracts/Poll.sol, maci/contracts/MACI.sol
Description: The message queue stores the state of messages to be processed by a coordinator within MACI. Its underlying implementation relies on a quintary binary tree. At leaf zero, this tree should be initialized with a default “nothing up my sleeve” value. The “nothing up my sleeve” value is defined within the MACI contract. However, the message queue is now stored within the Poll contract and as such should be initialized within the PollFactory contract.

Impact: A malicious user can initialize the message queue with a value that they know how to decrypt but that takes a very long time to generate a proof for. This would effectively be a DoS attack on the coordinator.
Suggestion: Move `NOTHING_UP_MY_SLEEVE` from Maci.sol to Poll.sol within the PollFactory contract. Add `messageAq.enqueue(NOTHING_UP_MY_SLEEVE)` to deploy.
Status: This has been rectified at commit 04f21b358b9efc17cffb8732c96f338ec56462d3.

Doesn’t take into account ERC20s that are deflationary or charge a fee upon transfer

Type: High
Files affected: qfi/contracts/GrantRound.sol, qfi/contracts/QFI.sol
Description: The GrantRound contract enables an owner to use any token they want for their grant rounds. As such, an owner can choose to use a token that charges a fee

upon transfer or deflationary. This affects the amount that the grant recipient gets as it is not currently taken into account during the calculations for grant distributions.

Impact: Users may receive less funding than they were allocated.
Suggestion: Check that the associated balance before and after a transfer is the expected amount.
Status: This has been rectified at commit b33b89a63f3e284bce0fe376bafc91c6de195e2c.

Transactions do not revert when there aren't enough funds

Type: Medium
Files affected: qfi/contracts/GrantRound.sol

Description: One shall use `assert` in replacement of `if` in line 307, so that the transaction will be reverted in case the funds are insufficient, and that the recipient will not be mistakenly marked as having received the payout.

Impact: When the grant round is canceled, the recipient might lose the chance to receive its payout if there aren't enough funds.

Suggestion: Replace `if` with `assert` in line 307.
Status: This issue has been rectified at commit [a116e1c88d92c0d048b8ea84a57f5df28877ffb0](a116e1c88d92c0d048b8ea84a57f5df28877ffb0).

Inconsistent restriction on voice credit upper bound

Type: Low
Files affected: maci/contracts/MACI.sol, maci/contracts/Poll.sol

Description: There is an upper bound on the maximum number of voice credits in `MACI` contract line 227. The preceding comment says that this bound is also enforced in the `MessageValidator` circuit, but this is not the case. Meanwhile, the `topup` function in `Poll` allows users to increase their voice credits without limitations. This shows that the upper bound constraint is inconsistent within the codebase.

Impact: The inconsistency can mislead users and developers.
Suggestion: Make the upper bound constraint consistent across the codebase.

Status: This has been rectified at issue 7a8c5c190793032ad10370da9da0d2256abdd999.

Redundant functions and redundant checks

Type: Low
Files affected: qfi/contracts/QFI.sol

Description: First, the function `closeVotingAndWaitForDeadline` in line 413 seems redundant since there is no real difference between `VOTING_PERIOD_OPEN` and `WAITING_FOR_FINALIZATION` stages. One can still vote in the latter stage. Second, the checks in lines 438 and 442 in the function `finalizeCurrentRound` seems redundant, as the values on the left hand side of the checks are function parameters, which can always be set to satisfy the checks.

Impact: These redundant checks can mislead users and developers into thinking that the stages have indeed changed or that the functions are safe under the checks.
Suggestion: Either remove the redundant codes, or fix the codes if they are not doing their jobs properly.
Status: This has been rectified as of commit 2c2338d7da23d9e64f04c2c59df12b63fa2af84e. The development team noted that `closeVotingAndWaitForDeadline()` works as intended and as such kept the implementation the same.

Potential for Re-entrancy in QFI

Type: Low
Files affected: qfi/contracts/QFI.sol

Description:
In QFI.sol, the functions `finalizeCurrentRound()`, `deployGrantRound()` and `contribute()` may be susceptible to re-entrancy attacks that may affect the state of the QFI contract resulting in miscalculation of allocation of funds. In `initialize()`, the checks-effects-interaction pattern is not enforced, as such it may be possible for an owner to take advantage of the fact that the contract helpers have not been initialized yet.

Impact: May be able to take advantage of state variables affecting the allocation of funds through re-entrancy.

Suggestion: Apply the checks-effects-interaction pattern to initialize(), contribute(), `finalizeCurrentRound()` and `deployGrantsRound()`.
Status: This has been partially rectified at commit 999f79cb99cbf79c9abfd91000a3735c2e74dfc1.

Potential for Re-entrancy in MACI

Type: Low
Files affected: qfi/contracts/MACI.sol

Description:
In initialize(), the checks-effects-interaction pattern is not enforced, as such it may be possible for an owner to take advantage of the fact that the contract helpers have not been initialized yet. In `signUp()`, the `numSignUps` state variable is incremented after several external calls are made. In particular, a malicious signUpGatekeeper instance might implement a register with recursive calls to signUp in MACI and thus the same user can be added multiple times but the numSignUps variable is not updated. In `deployPoll()`, the `polls` and `nextPollId` state variables are modified after a call to the `pollFactory.deploy()` function.It may be possible to do an re-entrancy attack in which these values are not updated.

Impact: May be able to fake the number of sign ups for a MACI instance.
Suggestion: Apply the checks-effects-interaction pattern to `initialize()`, `signUp()`, and `deployPoll()`.
Status: Has been rectified at commit 6f1fa85299ebbc8fe10e30691afe8f036b8c68d1 and [d62c7c710ba126ced713b8d32190408dbf5fa29f](d62c7c710ba126ced713b8d32190408dbf5fa29f) . In particular, `deployPoll()` still doesn't completely enforce the checks-effects-interactions pattern. This has been noted in issue 504.

Potential for Re-entrancy in Poll

Type: Low
Files affected: maci/contracts/Poll.sol

Description:
In each of `mergeMaciStateAq()`, `publishMessage()` and `topup()` in Poll, state variables are updated after external calls to functions that can take advantage of the order of execution of these functions.

Impact: In `topup()`, a malicious caller may be able to top up their credits for the same message multiple times before `numMessages` increments. Since `numMessages` is used to keep track of the total number of messages, this will result in the top up credits being inflated. A similar issue arises in `publishMessage()`. For `mergeMaciStateAq()`, a malicious caller can attempt to get the account tree associated with a specific pollId multiple times. However, `merge()` in `AccQueue.sol` will always return the same tree of the same depth as long as it contains the same accounts between re-entrant calls.
Suggestion: Apply the checks-effects-interaction pattern to `topup()`, `publishMessage()` and `mergeMaciStateAq()`.
Status: Has been rectified at commit 6f1fa85299ebbc8fe10e30691afe8f036b8c68d1. Further commit a0b07b99489109f6aa937f6f815dfb83686ce589 has added a test case in order to ensure that the behavior of merge() is as expected if mergeMaciStateAq() were to be re-entered.

Return values of transferFrom calls are not checked

Type: Low
Files affected: maci/contracts/Poll.sol and qfi/contracts/GrantRound.sol
Description: In `Poll.sol`, the `topup()` function calls transferFrom on a `TopUpCredit` token. Since the `TopUpCredit` token uses the default transferFrom implementation, it will properly revert in the case of an error and return true if everything was executed properly. However, in `GrantRound.sol`, the ERC20 token provided can have any implementation of transferFrom and as such, may have unintended consequences on any calls that are made to it.

Impact: Not checking the return value of `transferFrom()` calls may result in unexpected behavior.
Suggestion: Check the return values of the transferFrom calls in order to ensure the proper execution.
Status: This has been rectified at commit 6f1fa85299ebbc8fe10e30691afe8f036b8c68d1. Further, GrantRound.sol already uses `safeTransferFrom()` which handles this internally.

Miscellaneous mistakes

Type: Low
Files affected: qfi/contracts/QFI.sol

Description: The payout token in line 298 of QFI should use `nativeToken` instead of an externally given ERC20 token. The state variable `contributorCount` should be updated not only in `contribute()` but also in `withdrawContribution()` and `acceptContributionsAndTopUpsBeforeNewRound()`.

Suggestion: Fix the mistakes accordingly.
Status: This has been rectified at commits https://github.com/quadr367033a2ee0ec431091a39d00c9b9acf5ad2304b and https://github.com/quadratic-fund55b512b4342aa060295dd58a543d4a079b8f6da7.

The QFI owner is powerful enough to decide the payout distribution

Type: Informational
Files affected: qfi/contracts/GrantRound.sol

Description: The `QFI` owner will always be the `GrantRound` owner. It can `cancel` the grant round and `transferMatchingFunds` to pay the funds out in an arbitrary way.

Suggestion: We consider the owner as given a lot of power, and we suggest making this situation explicit in the documents.
Status: This has been acknowledged by the development team. They will reconsider a new design and implementation in order to minimize this risk in a future release.

Subsidy functionality is incomplete

Type: Informational
Files affected: maci/contracts/Poll.sol

Description: Subsidy is only calculated but not distributed in the `Poll` contract. This means that the subsidy design is not fully implemented.

Suggestion: Implement the subsidy distribution functionality.
Status: This has been acknowledged and will be completed in a future update.

Incorrect error messages

Type: Informational
Files affected: qfi/contracts/QFI.sol

Description: Some functions can be called only when the contract is in some specific stages. When this is not the case, there will be error messages. In lines 416 and 461, the error messages are incorrect about the current stage.

Suggestion: Fix the error messages accordingly.
Status: This has been rectified at commit https://githa15489336450ee9df27737215068b9e853aa7cda.

Return value of ExtContracts.maci.mergeStateAq is ignored in Poll contract

Type: Informational
Files affected: maci/contracts/Poll.sol
Description: In `mergeMaciStateAq` in the Poll contract, `ExtContracts.maci.mergeStateAq` is called in order to merge the State queue in the MACI contract. This function returns the new root of the merged tree. However, this value is ignored when it should be used to set the `mergedStateRoot` variable.

Suggestion: Set `mergedStateRoot = extContracts.maci.mergeStateAq(_pollId)`.
Status: This issue has been rectified at commit 76c991a2c4f580c353f526375daf138fbb66ec92.

Anyone can pass mismatched length encryption keys and message batches to PublishMessageBatch

Type: Informational
Files affected: qfi/contracts/GrantRound.sol

Description: In `PublishMessageBatch` in `GrantRound` implicitly assumes that `_messsages` and `_encPubKeys` have the same number of elements. However, if there are more elements in `_encPubKeys` than `_messages`, then the for loop will not take into account the other elements in `_encPubKeys`. A similar situation occurs if there are more elements in `_messages` than there are elements in `_encPubKeys`.
Status: This has been rectified at commit a15489336450ee9df27737215068b9e853aa7cda.

Naming conflict with transferMatchingFunds in FundsManager and GrantFactory

Type: Informational
Files affected: qfi/contracts/FundsManager.sol, qfi/contracts/GrantFactory.sol

Description: In FundsManager and GrantFactory, there is a transferMatchingFunds function. However, the functionality of both differ. As such, this can cause confusion when reading the contracts.
Status: This has been rectified at commit a116e1c88d92c0d048b8ea84a57f5df28877ffb0.

isAfterDeadline, topup and publishMessage in Poll rely on the block timestamp

Type: Informational
Files affected: maci/contracts/Poll.sol
Description: In isAfterDeadline, topup and publishMessage within the Poll contract, block timestamps are used to enforce conditions regarding the validity of when various portions of a voting period occur. As block timestamps are controlled by miners, it is possible to manipulate when the voting period starts/ends.
Status: This issue has been acknowledged by the development team and will be taken into account within their documentation.

Boolean conditions in require statements can be simplified

Type: Informational
Description: In many contracts across both repositories, require statements tests whether a boolean value is true or false. This is unnecessary as the boolean constant variable can be used itself to make up a valid boolean condition.
Status: Has been rectified at commit 6f1fa85299ebbc8fe10e30691afe8f036b8c68d1.

State variables are incremented within a loop in AccQueue.mergeSubRoots

Type: Informational
Files affected: maci/AccQueue.sol
Description: Within the implementation of mergeSubRoots, the nextSubRootIndex counter is incremented within a for loop. As this requires an SLOAD operation for each for loop, it is quite gas intensive. Instead, a local variable that keeps the state of the nextSubRootIndex should be used within the for loop and then nextSubRootIndex can be updated outside of the for loop with this local variable.
Status: Has been partially rectified at commit 6f1fa85299ebbc8fe10e30691afe8f036b8c68d1.

_decimals in TopUpCredit and stateTreeDepth in MACI are not constant

Type: Informational
Files affected: maci/TopUpCredit.sol, maci/MACI.sol
Description: _decimals in TopUpCredit.sol and stateTreeDepth in MACI.sol are set to specific values that are not changed throughout the contract. As such, these should be set to constant in order to minimize gas costs.
Status: This has been rectified at commit f6caf665127a86504c4d163c34575a92bb2ebe04

ERROR_VK_NOT_SET and ERROR_SB_COMMITMENT_NOT_SET should be moved to PollProcessorAndTallyer

Type: Informational
Files affected: maci/contracts/Poll.sol
Description: ERROR_VK_NOT_SET is an error message to indicate that the verifier key registry is not set to be used within the Poll. And, ERROR_SB_COMMITMENT_NOT_SET is an error message to indicate that the state leaves and ballots commitment has not been set. However, in this new Poll contract, the verifier key registry is no longer needed and checking the sbCommitment is no longer done within the poll contract. As such, both are only needed within the PollProcessorAndTallyer contract. As such, this error message should be moved to the PollProcessorAndTallyer contract and the appropriate check done.

Suggestion: As there is no use for the ERROR_SB_COMMITMENT_NOT_SET error message, it can be safely removed. ERROR_VK_NOT_SET should be moved to the PollProcessorAndTallyer contract and the appropriate check be made.
Status: This has been addressed at commit [6f1fa85299ebbc8fe10e30691afe8f036b8c68d1]

General Recommendations

Set public functions to external if they have no internal calls

Public functions which are solely called from external (other smart contracts or externally owned accounts) should be marked `external` instead of `public`, since it saves gas costs.

Use consistent Solidity versions across files

In general it is always recommended to use the fixed solidity version and the same solidity version. When using ^0.8.0 pragma, it might use the nightly version of solidity, which might have experimental features.

Do multiplications before divisions

When dealing with floating/fixed point numbers, it is recommended that multiplication operations are done before division in order to preserve as much precision as possible.

Check return values of external calls

This codebase makes heavy use of external contracts and as such makes many external calls. We highly recommend that the return values of external calls are checked consistently throughout the codebase in order to fail gracefully and properly handle exceptions and reverts.