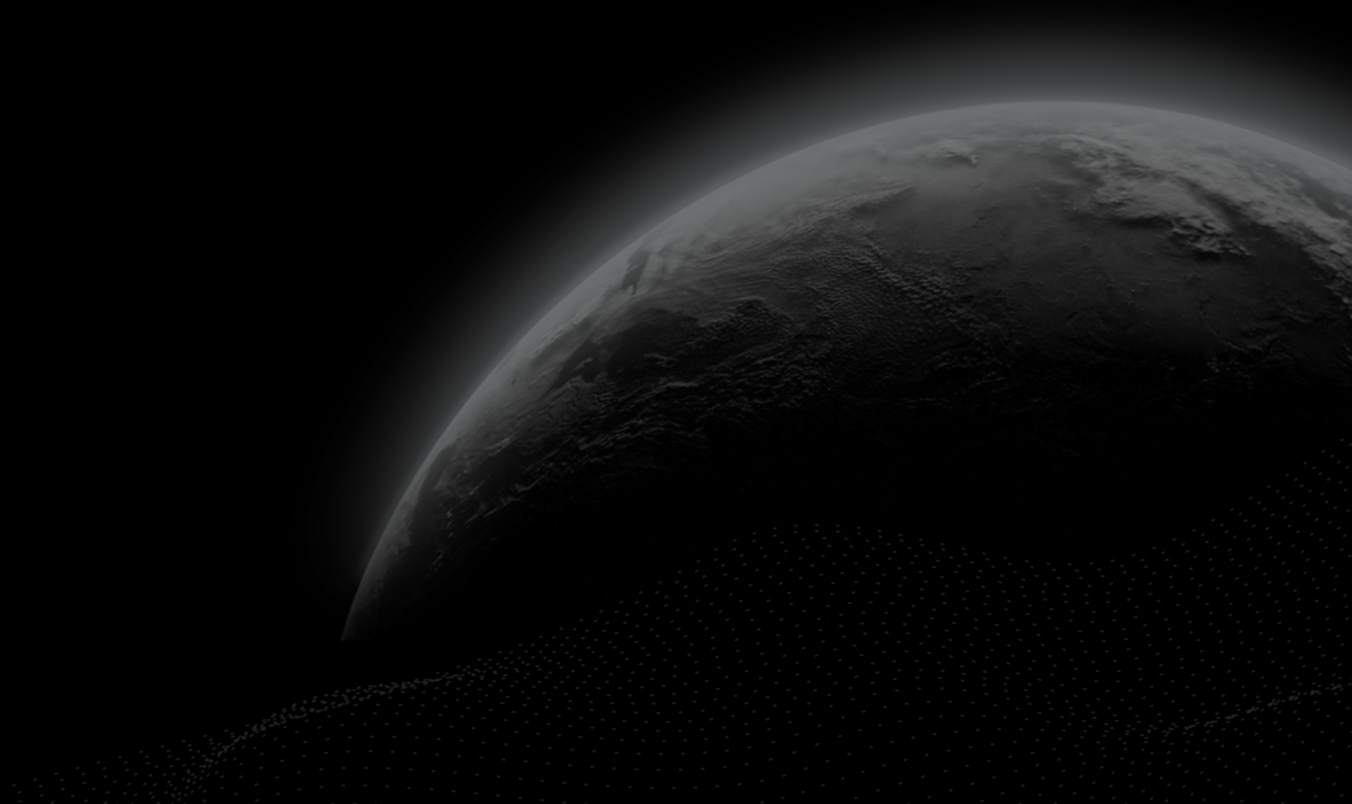
CERTIK

Security Assessment

Arcana Network

CertiK Verified on Dec 28th, 2022



CertiK Verified on Dec 28th, 2022

Arcana Network

The security assessment was prepared by CertiK, the leader in Web3.0 security.

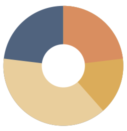
Executive Summary

TYPES	ECOSYSTEM	METHODS
KMS(Key Management System)	Other	Manual Review, Static Analysis
LANGUAGE	**TIMELINE**	**KEY COMPONENTS**
Golang	Delivered on 12/28/2022	N/A

CODEBASE
https://github.com/arcana-network/dkgnode/
...View All

COMMITS
1dd34fb3b33380ea26ffadf4f848d765c34ec9ad
...View All

Vulnerability Summary



13	12	0	0	1	0	0
Total Findings	Resolved	Mitigated	Partially Resolved	Acknowledged	Declined	Unresolved

🟥	0	Critical		Critical risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks.
🟧	3	Major	3 Resolved	Major risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project.
🟨	2	Medium	2 Resolved	Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.
🟨	5	Minor	4 Resolved, 1 Acknowledged	Minor risks can be any of the above, but on a smaller scale. They generally do not compromise the overall integrity of the project, but they may be less efficient than other solutions.
🟦	3	Informational	3 Resolved	Informational errors are often recommendations to improve the style of the code or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

TABLE OF CONTENTS | ARCANA NETWORK

CODEBASE | ARCANA NETWORK

Repository

https://github.com/arcana-network/dkgnode/

Commit

1dd34fb3b33380ea26ffadf4f848d765c34ec9ad

AUDIT SCOPE | ARCANA NETWORK

27 files audited ● 14 files with Acknowledged findings ● 5 files with Resolved findings ● 8 files without findings

ID	File	SHA256 Checksum
● AUX	keygen/message_handlers/aba/aux1_handler.go	459495d77a4418ebdbcbcbf0f56377000d00d6c8b6fd8602f1c6275b42f7d2f5
● AU2	keygen/message_handlers/aba/aux2_handler.go	722d8fe2626bae75949f275a5b8ea880d4e4b57a61e46e6589340a30fcfb257b
● AUS	keygen/message_handlers/aba/auxset_handler.go	7d5ae15c335584c745f5160299b731c5bad5114a127a884490adcf3eaa197141
● COI	keygen/message_handlers/aba/coin_handler.go	0bdcf527db61209d90739368169053a6d03d199662cbfbb931c1961927f8aacf
● CON	keygen/message_handlers/aba/coin_init_handler.go	3723934859b0454faa4d02fd70360656deced3e8f3630886eda7f71560a4a238
● EST	keygen/message_handlers/aba/est1_handler.go	06a3740da84912c433a5a2a5f0e9ae2f8954ba47f5eab26fda69c89f945c64b3
● ES2	keygen/message_handlers/aba/est2_handler.go	94f144f1c984500059cd2735e0771bc8abf7e7b28b6c713c3df1ddb413678aaa
● INI	keygen/message_handlers/aba/init_handler.go	d1964bac95cace70520b03c59329cfe97f81af78ca5013f3ccc3c719d8664594
● ECH	keygen/message_handlers/acss/echo_handler.go	5cd6e3341bbeb6430aeb09102b66dd79cc3f17125fdfeba63509f6477e9f0d59
● OUT	keygen/message_handlers/acss/output_handler.go	f613181d9b0a38d4a2a21cc3f7fe92f1f217c9539ab9454fb4917660c5287e70
● PRO	keygen/message_handlers/acss/propose_handler.go	8944e54e465734d20ecc6e23fe9f510d90c018474249783245349fce0708418c
● REA	keygen/message_handlers/acss/ready_handler.go	ba2fa80d9b2c48d5869343600511c7f6ab35e631a3abee048a8aff2a24aaf4c8
● SHA	keygen/message_handlers/acss/share_handler.go	9d1e4b042a702e182098b68c5a7db78af5aaa150a51578f76389d527875750e3

ID	File	SHA256 Checksum
● KEG	keygen/keygen_service.go	31efbf6a0101cb1f21558b7ffba0040c8c7a0789da6225a275e2a3b1bbc1a420
● ACS	keygen/common/acss/acss.go	6986358c889f5c8954d809a37fbdff8fd114e1b433751ff7650b6a5d89e5cb5d
● ECO	keygen/message_handlers/keyset/echo_handler.go	d69230f89bbc44111d5cc5dd1f701ae57d8fc1a8f3049dc25e6913de330e629e
● PRP	keygen/message_handlers/keyset/propose_handler.go	23a2fb472a83ff2a46fa8f6a11fd2199260ccc11f2f94b184aa40f7ce7d9e5e9
● RED	keygen/message_handlers/keyset/ready_handler.go	9d3c7f6901cf6c28546a6c9160784e913584e32a85fa1f622607a0f8a51c66dd
● TRA	keygen/transport.go	505884fc5412290ce587498e746c8318652509909653d1d4137fd633b1ef8185
● COM	keygen/common/common.go	fe782976ccf5d9cf2bfe0ffd93862379df008fcf3de32c8dd2b0e07bd2272dbb
● COO	keygen/common/common_test.go	c1349f15f5820cee99874d1f6dd75e2d3254e912360a95d8392cd84bb327ce4d
● ABA	keygen/common/aba/aba.go	916ceae01a04294315931f39f6955abf3a7e4ee1e94930438458bdd5aca110e0
● COK	keygen/message_handlers/keyset/common.go	e443f47b07cd0f6a9dbf7a83cbc7ed887197f6b98048e6caeb843cbcb5368efd
● INT	keygen/message_handlers/keyset/init_handler.go	e1c4750152f9cf7224e1e5b9d9b4bd5ed3c34af2ad5b5a219716b325ed9158a4
● OUP	keygen/message_handlers/keyset/output_handler.go	75f5f41c43b92c8563616ba21400138b6f63a35b6e8824131b0252c1d0941174
● MEA	keygen/messages/messages.go	b601fd742fee7d9f4b0510d8fd949e16686bbb2fb70acf6db9a01a1896040e51
● SIG	keygen/signature.go	c1f8e2eb6d68e6a272383fc1f0be099d0b5644f3dbefa408099ee7391b8f7e79

APPROACH & METHODS | ARCANA NETWORK

This report has been prepared for Arcana Network to discover issues and vulnerabilities in the source code of the Arcana Network project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Manual Review and Static Analysis techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Testing the smart contracts against both common and uncommon attack vectors;
- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

REVIEW NOTES | ARCANA NETWORK

Messages Handlers

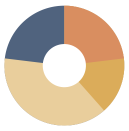
Function	Descriptions	PASS
process ShareMessage	1. Receive a ShareMessage from its own node. 2.Create random secrets and ephemeral keypair. 3. Generate shares and commitments of corresponding length according to n, k and secret respectively. 4. Encrypt each share with node respective generated symmetric key, add to share map. 5. Create and Send ProposeMessage	✔
process ProposeMessage	1. Check the received Message. 2. Verify self share against commitments. 3. If verified, the data in the Message is encoded into n shares and also hashed, and the EchoMessage is constructed with node specific share and hash and self sent by each node.	✔
process EchoMessage	1. Check the received Message. 2. Count the echo messages received. 3. If the number of echo received satisfies BFT or reconstructed need, Create and broadcast ReadMessage	✔
process ReadyMessage	1. Check the received Message. 2. Count the ready messages received, and store the share. 3. If the number of ready received satisfies BFT, verifies all the shares received against m.Hash. 4. If verified, create and send the OutputMessage	✔
process OutputMessage	1. Check if this round completeness has been accounted in current adkg. 2. If not, recover shared key and data, verifies if the share fits the polynomial commitments. 3. If verified, store share in session store 4. If the number of stored share equals k, go on the next Round	✔

Security Concerns

Function	Descriptions	Pass
Protocol Soundness	Proper execution of the protocol and therefore the sub protocols under the assumption of n >= 3t + 1. Here t is the assumed number of malicious parties, every phase of the ADKG should follow a happy flow when the mentioned assumption is met.	✔

Function	Descriptions	Pass
ACSS Soundness	The first phase of the ADKG involves the ACSS, which makes up the sharing of secrets involved in the generation of the key. Since the ACSS is the secret sharing phase, we need to make sure that this sub protocol has been implemented correctly and securely, so that any attempt at exploiting public messages to gain more than the intended information about the Key shares fails.	✔
Cryptographic Primitives	Correct implementation of Encryption, Decryption, NIZK, Commitments.	✔
RBC Finality	The second phase, and even the ACSS, uses a Reliable Broadcast (RBC) protocol. We need to make sure this RBC has been put together correctly, in-order to avoid any potential stall in the middle of the protocol that might subsequently end up stalling the whole key generation process	✔
ABA Finality	The third phase, Asynchronous Binary Agreement (ABA) takes votes and involves some communication amongst the nodes. In some cases, a node might need to wait some iterations of the ABA to end, before it decides to input on other ABAs. This might cause a delay if enforced incorrectly. We need to make sure that the ABA phase always terminates.	✔
Asynchronous Resilience	We wish to ensure that the protocol performs correctly under realistically unstable (asynchronous) network assumptions for each end node. Essentially, random delays in connections and varying network speeds should not force the protocol to abort or stall, unless a significant majority of the network is experiencing severe difficulties.	✔

FINDINGS | ARCANA NETWORK



13	0	3	2	5	3
Total Findings	Critical	Major	Medium	Minor	Informational

This report has been prepared to discover issues and vulnerabilities for Arcana Network. Through this audit, we have uncovered 13 issues ranging from different severity levels. Utilizing the techniques of Manual Review & Static Analysis to complement rigorous manual code reviews, we discovered the following findings:

ID	Title	Category	Severity	Status
ACS-01	Logic Issue On `acss.Decode()`	Logical Issue	Major	● Resolved
MES-01	Potential `nil` Pointer Dereference	Volatile Code	Major	● Resolved
OUT-01	Missing Validation For Sender	Logical Issue	Major	● Resolved
COI-01	Unlocked Session Store	Volatile Code	Medium	● Resolved
MES-02	Unreasonable Defer Code Order	Logical Issue	Medium	● Resolved
EST-01	Missing Necessary Lock	Volatile Code	Minor	● Resolved
KEG-01	Hardcode In `Start()`	Volatile Code	Minor	● Resolved
KEY-01	Dead Code	Logical Issue	Minor	● Resolved
KEY-02	Lack Of Log Tracking	Coding Style	Minor	● Acknowledged
REA-01	Confusing Logic	Logical Issue	Minor	● Resolved

ID	Title	Category	Severity	Status
KEG-02	The `TODO` Comment	Coding Style	Informational	● Resolved
MES-03	Typo	Coding Style	Informational	● Resolved
MES-04	Incorrect Comments	Coding Style	Informational	● Resolved

ACS-01 | LOGIC ISSUE ON `acss.Decode()`

Category	Severity	Location	Status
Logical Issue	● Major	keygen/common/acss/acss.go: 217~222	● Resolved

Description

1. We understand that due to the logic of `FEC.Encode()`, it is necessary to append `msg` so that the length of `msg` is divisible by `k`. Therefore, the logic of the function `acss.Decode()` will remove the `0` byte at the end.
2. The `m.Data` would be `acss.Encode()` in the `propose_handler.go` and `acss.Decode()` in the `ready_handler.go`.
3. But if `m.Data` ends with 0, `acss.Decode()` may remove the 0 belonging to `m.Data`. The result of `acss.Decode()` will be different from `m.Data`. So the L135 of `acss/ready_handler.go` will be `false` and the following program will not work. In this case, it may cause the program to be blocked.

Recommendation

We recommend modifying the solution of `acss.Encode()` and `acss.Decode()` to avoid validation failures.

Alleviation

Arcana team heeded the advice and resolved the finding in the commit hash 364c142df8dfb285abba61b5760dea507970880b.

MES-01 | POTENTIAL `nil` POINTER DEREFERENCE

Category	Severity	Location	Status
Volatile Code	● Major	keygen/message_handlers/aba/coin_init_handler.go: 47~52; keygen/message_handlers/keyset/propose_handler.go: 55~61	● Resolved

Description

In file `./keygen/message_handlers/aba/coin_init_handler.go` , if the `err` handled at line 47 is not `nil` , the statement at line 48 will cause to panic because the pointer of `ADKGSession` we got will be `nil` .

```
47        s, err := common.GetSessionStoreFromRoundID(m.RoundID, p)
48        s.Lock()
49        defer s.Unlock()
50        if err != nil {
51            return
52        }
```

This issue also happens in file `./keygen/message_handlers/keyset/propose_handler.go` at line 55.

```
55        store, err := common.GetSessionStoreFromRoundID(m.RoundID, p)
56        store.Lock()
57        defer store.Unlock()
58        if err != nil {
59            log.Infof("Could not get session store from roundID, err=%s", err)
60            return
61        }
```

Recommendation

We recommend handling the error `err` before using the pointer `s` .

Alleviation

Arcana team heeded the advice and resolved the finding in the commit hash 79588d652273b3f0a7f8dc0ccd3e8cee1768c579.

OUT-01 | MISSING VALIDATION FOR SENDER

Category	Severity	Location	Status
Logical Issue	● Major	keygen/message_handlers/acss/output_handler.go: 42	● Resolved

Description

The `process()` function in `acss.output_handler.go` does not check the `senderIndex` of the `output` message, which is sent by the `ready_handler`. However, the `ready_handler` only sends this message to the self-node. If the `output_handler` does not include a self-check, an attacker could exploit this vulnerability by constructing an `output` message and sending it to the node, potentially disrupting the protocol process. It is important to ensure that the `output_handler` includes a self-check to prevent such attacks.

Recommendation

To add a sender check to the `process()` function in `acss.output_handler.go`.

```
	if sender.Index != self.ID() {
		return
	}
```

Alleviation

Arcana team heeded the advice and resolved the finding in the commit hash 2db0683b968a60c461ade2e1610f38a51aaf82c2.

COI-01 | UNLOCKED SESSION STORE

Category	Severity	Location	Status
Volatile Code	● Medium	keygen/message_handlers/aba/coin_handler.go: 47	● Resolved

Description

The session store `s`, which is obtained at line 47 in the file `./keygen/message_handlers/aba/coin_handler.go`, is unlocked.

Recommendation

We recommend that locks are required for both reading and writing.

Alleviation

Arcana team heeded the advice and resolved the finding in the commit hash 819bf73a9595eb100f17a73e5566aa2558ccc2aa.

MES-02 | UNREASONABLE DEFER CODE ORDER

Category	Severity	Location	Status
Logical Issue	● Medium	keygen/message_handlers/acss/echo_handler.go: 79~81; keygen/message_handlers/keyset/echo_handler.go: 81~83	● Resolved

Description

The `defer keygen.Unlcok()` comes after `defer func() { keygen.State.ReceivedEcho[sender.Index] = true}()`. We all know that the order of execution of the defer function is `The Last Execute First`. So `keygen.Unlock()` will be executed before `func() { keygen.State.ReceivedEcho[sender.Index] = true}()`.

It doesn't make sense to write data after the lock is released. So it might be better to change the order of these two procedures.

Recommendation

We recommend swapping the order of these two lines of code.

Alleviation

Arcana team heeded the advice and resolved the finding in the commit hash 819bf73a9595eb100f17a73e5566aa2558ccc2aa.

EST-01 | MISSING NECESSARY LOCK

Category	Severity	Location	Status
Volatile Code	● Minor	keygen/message_handlers/aba/est1_handler.go: 48~53	● Resolved

Description

The file in the `aba` package does not add a read lock to `ABAStore` . Since it has a write lock on `ABAStore` , concurrency problems may occur when multiple threads read and write to the same variable.

For example, in `est1_handler.go` (L48 and L53), L48 checks if the store contains `senderIndex` and L53 adds `senderIndex` to the store. The same `senderIndex` may be added twice in L53 if there are multiple threads with the same `senderIndex` entering the process function. So, to be more secure, we recommend that locks are required for both reading and writing.

Recommendation

We recommend adding a lock for to the `ABAStore` .

Alleviation

Arcana team heeded the advice and resolved the finding in the commit hash 819bf73a9595eb100f17a73e5566aa2558ccc2aa.

KEG-01 | HARDCODE IN `Start()`

Category	Severity	Location	Status
Volatile Code	● Minor	keygen/keygen_service.go: 46~47	● Resolved

Description

```
42 func (service *KeygenService) Start() error {
43     ChainMethods := service.broker.ChainMethods()
44     selfIndex := ChainMethods.GetSelfIndex()
45     selfPubKey := ChainMethods.GetSelfPublicKey()
46     currNodeList := ChainMethods.AwaitCompleteNodeList(1)
47     currEpoch := ChainMethods.GetCurrentEpoch()
```

On line 46, the fixed value `1` is passed to the `ChainMethods.AwaitCompleteNodeList()` method, is this test code?

We think it would be better to execute the `ChainMethods.GetCurrentEpoch()` method on line 47 and then pass the resulting value `currEpoch` as a parameter to the `ChainMethods.AwaitCompleteNodeList()` method.

Recommendation

We recommend swapping the order of codes on lines 46 and 47 and using the resulting value `currEpoch` as an argument to the `ChainMethods.AwaitCompleteNodeList()` method.

Alleviation

Arcana team heeded the advice and resolved the finding in the commit hash 364c142df8dfb285abba61b5760dea507970880b.

KEY-01 | DEAD CODE

Category	Severity	Location	Status
Logical Issue	● Minor	keygen/keygen_service.go: 118~123, 357~359; keygen/message_handlers/aba/aux2_handler.go: 113~115; keygen/transport.go: 24~30	● Resolved

Description

1. Un-reached Case

   It seems to us that `receive_BFT_message` is only reached when executing the `ReceiveBFTMessage()` method in the file `common/service_broker.go` , but we did not find the location of the `ReceiveBFTMessage()` method call.

2. Unused Method `ReceiveBroadcast()`

   The method `ReceiveBroadcast()` for `KeygenTransport` is only used in the un-reached case "receive_BFT_message"(item 1 in this description), therefore this method can be removed.

3. Unused Method `ProcessBroadcastMessage()`

   The method ProcessBroadcastMessage() `for` *KeygenNode `is only called in unused method` ReceiveBroadcast()`(item 2 in this description), therefore this method can be removed.

4. Redundant `if` statement

   The `if` statement in file `./keygen/message_handlers/aba/aux2_handler.go` at line 113 will catch the occurred error and return without printing log and the `if` statement at line 116 will never be reached. Based on the contents of the log output in the if statement at line 116, we believe that the `if` statement at line 113 is redundant and should be removed.

Recommendation

We recommend reviewing the logic to ensure it meets the design intent.

Alleviation

Arcana team heeded the advice and resolved the finding in the commit hash 364c142df8dfb285abba61b5760dea507970880b.

KEY-02 | LACK OF LOG TRACKING

Category	Severity	Location	Status
Coding Style	● Minor	keygen/keygen_service.go: 128~130, 258~261; keygen/message_handlers/aba/aux1_handler.go: 85~87; keygen/message_handlers/aba/aux2_handler.go: 106~108, 152~154, 163~165; keygen/message_handlers/aba/auxset_handler.go: 76~78; keygen/message_handlers/aba/coin_handler.go: 43~45, 48~50, 56~58, 84~86, 88~90; keygen/message_handlers/aba/coin_init_handler.go: 50~52, 55~57, 75~77; keygen/message_handlers/aba/est1_handler.go: 63~65, 80~82; keygen/message_handlers/aba/est2_handler.go: 62~64, 80~82; keygen/message_handlers/aba/init_handler.go: 57~59; keygen/message_handlers/acss/echo_handler.go: 106~108, 117~119; keygen/message_handlers/acss/output_handler.go: 110~112; keygen/message_handlers/acss/propose_handler.go: 53~55; keygen/message_handlers/acss/ready_handler.go: 140~142; keygen/message_handlers/acss/share_handler.go: 37~39	● Acknowledged

Description

1. The `Stop()` method will be called when the blockchain was stoped in any case. But it only returns `nil` without any valuable response or log.
2. There is no processing or log output when some errors occur.

Recommendation

We recommend reviewing the logic to ensure it meets the design intent and printing a log to improve the maintainability of the system.

Alleviation

Arcana team acknowledged this finding.

REA-01 | CONFUSING LOGIC

Category	Severity	Location	Status
Logical Issue	● Minor	keygen/message_handlers/acss/ready_handler.go: 119	● Resolved

Description

If the condition `len(keygen.ReadyStore) < (2*f + 1)` is `true` , the for loop does not need to continue, because the condition is always `true` .

1. Let's assume that `len(keygen.ReadyStore) > 2*f + 1` is `true` . In this case, the loop will break on the first entry into the loop. No matter what `len(keygen.ReadyStore)` is, as long as it is greater than `2*f + 1` , the loop will execute only once.
2. But if the condition `len(keygen.ReadyStore) > 2*f + 1` is `false` , each loop will do nothing.

It appears that the for loop is not working. So, what is the design intent of the statement?

Recommendation

We recommend reviewing the logic to ensure it meets the design intent.

Alleviation

`[Arcana]` : The statement is supposed to run only once. That’s expected, if the hash doesn’t match there is no point in going forward, you can check out the practical ADKG paper for more details on the loop.

KEG-02 | THE `TODO` COMMENT

Category	Severity	Location	Status
Coding Style	● Informational	keygen/keygen_service.go: 93~105	● Resolved

Description

There is a TODO comment in the `keygen_service.go` file. Based on the comment, it looks like there is an unfinished function. If not, consider removing the extra TODO comment. A good practice is to have no TODO comments in the product code.

Recommendation

We recommend removing the TODO comment on the above line, or adding the unfinished code and then removing the comment.

Alleviation

Arcana team heeded the advice and resolved the finding in the commit hash 819bf73a9595eb100f17a73e5566aa2558ccc2aa.

MES-03 | TYPO

Category	Severity	Location	Status
Coding Style	● Informational	keygen/message_handlers/acss/echo_handler.go: 84; keygen/message_handlers/keyset/echo_handler.go: 86	● Resolved

Description

The comments should be `// Check if the echo has already been received`.

Recommendation

We recommend correcting these comments to improve the readability of the code.

Alleviation

Arcana team heeded the advice and resolved the finding in the commit hash 819bf73a9595eb100f17a73e5566aa2558ccc2aa.

MES-04 | INCORRECT COMMENTS

Category	Severity	Location	Status
Coding Style	● Informational	keygen/message_handlers/acss/ready_handler.go: 82, 101; keygen/message_handlers/keyset/ready_handler.go: 82, 97	● Resolved

Description

File : ./keygen/message_handlers/acss/ready_handler.go

1. "Make sure the echo received from a node is set to true" in line 82 should be "ready received".
2. "increment the echo messages received" in line 101 should be "ready message".

File : ./keygen/message_handlers/keyset/ready_handler.go

1. "Make sure the echo received from a node is set to true" in line 82 should be "ready received".
2. "increment the echo messages received" in line 97 should be "ready message".

Recommendation

We recommend correcting the comments to improve readability.

Alleviation

Arcana team heeded the advice and resolved the finding in the commit hash 79588d652273b3f0a7f8dc0ccd3e8cee1768c579.

OPTIMIZATIONS | ARCANA NETWORK

ID	Title	Category	Severity	Status
KEG-03	Unused Argument `nodeIndex`	Gas Optimization	Optimization	● Resolved

KEG-03 | UNUSED ARGUMENT `nodeIndex`

Category	Severity	Location	Status
Gas Optimization	● Optimization	keygen/keygen_service.go: 305	● Resolved

Description

The `nodeIndex int` argument is not used in the `NewKeygenNode()` method.

Recommendation

We recommend reviewing the logic to ensure it meets the design intent.

Alleviation

Arcana team heeded the advice and resolved the finding in the commit hash 819bf73a9595eb100f17a73e5566aa2558ccc2aa.

APPENDIX | ARCANA NETWORK

Finding Categories

Categories	Description
Gas Optimization	Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.
Logical Issue	Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.
Volatile Code	Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.
Coding Style	Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

DISCLAIMER | CERTIK

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without CertiK's prior written consent in each instance.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER'S OR ANY OTHER PERSON'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE

FOREGOING, CERTIK PROVIDES NO WARRANTY OR UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

CertiK | **Securing** the **Web3** World

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.