

MAP Protocol

Contract Audit Report

VER 1.0

11th July 2022

No. 2022071115361

Project Summary

1. Project Introduction

MAP Protocol promotes the whole cross-chain ecosystem by building omnichain infrastructure. Any developer using MAP Protocol will automatically inherit the omnichain attribute without requiring the developer to handle cross-chain communication (error-prone), thus reducing the technical threshold for building omnichain DAPPs.

1. MAP Protocol is divided into three layers: protocol layer, cross-chain service layer (MCS), and application layer. 2. MAP Protocol is verified through the smart contract on the chain. 3. The cross-chain authentication network of MAP Protocol adopts light client independent self-authentication

2. Audit Summary

Project Name	MAP Protocol	Platform	N/A
Token	N/A	Token symbol	N/A
Start date	29th June 2022	Language	Solidity
End date	11th July 2022	Website	N/A
Github	https://github.com/mapprotocol/map-contracts/tree/ea53c8931353c6693bdcdd0c996a3a9319d784f8/mapclients/eth/contracts	Whitepaper	N/A

3. Audit Scope

ID	File	SHA-256 checksum
contracts	LightNode.sol	a4b29ddf406554d0c6b798e773048a04e50dad8774a1cc35985f7d963df18319
contracts	LightNodeProxy.sol	1a265a3c24a9ddba8f236f4abb4f7f9e7612a0535024d52a140f295152e48610
contracts	VerifyTool.sol	7d12b26af32038cc9c8a78a4f8a84d88b5700d1e09f9a72e9810fd4bc350c3b3
contracts/lib	MPT.sol	3ab7fb3466ca64f234964fd26daaa507d45700fa193b8adcc1b1362435b83cea
contracts/lib	RLPEncode.sol	cd552d56e3589eeff4fb4286e7c4527630808aa16a817ff903b0fae012a70181
contracts/lib	RLPReader.sol	b06774b7ab7ed057fca9b76f0d719b826d616c8e08ccb5e41e1a8c175a9571b0
contracts/bls	BGLS.sol	fa630679679afad4edc784521df7d25a6f3c391aa1bf9e895b541be19c5b271
contracts/bls	BlsCode.sol	62196c5e1f4e67833d24f3a3a288ef7bae0f9f565e06fdfba25a0180cc256579

4. Code Structure

—— LightNode.sol	#Verification logic of light nodes
—— LightNodeProxy.sol	#ERC1967agency contract
—— VerifyTool.sol	#Contract of MPT and signature verification tool
—— bls	
—— BGLS.sol	#BGLS aggregate signature
—— BlsCode.sol	#BLS Codec
—— interface	#Interface definition
—— IBLSPoint.sol	
—— ILightNode.sol	
—— ILightNodePoint.sol	
—— IVerifyTool.sol	
—— lib	
—— MPT.sol	#lib Verified by MPT
—— RLPEncode.sol	#lib encrypted by RLP
—— RLPReader.sol	#lib read by RLP

Audit Report Summary

1. Audit Methods

The audit was conducted to gain a clear understanding of how the project was implemented and how it works. The audit team conducted in-depth research, analysis, and testing of the project code and collected detailed data. In this report, the audit team will list in detail each issue identified, where it is located, the root cause of the issue, and a description of the issue, and will recommend changes to the issue accordingly.

Audit methods	Static analysis, Manual Review
---------------	--------------------------------

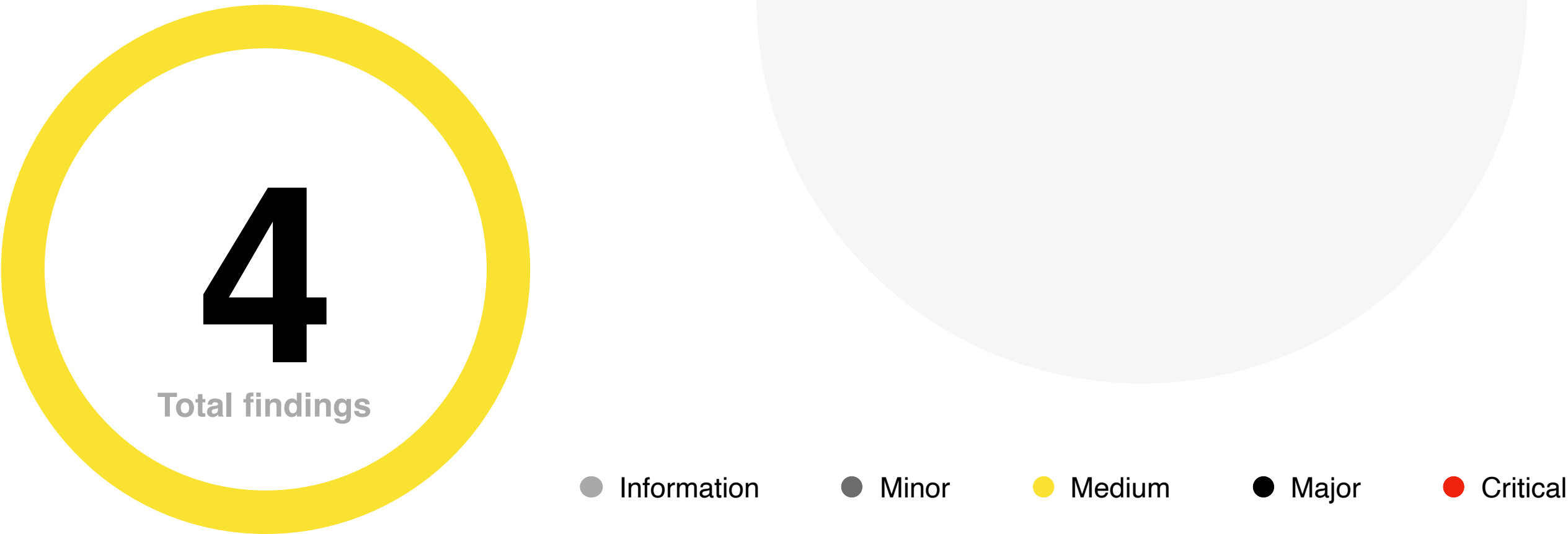
2. Audit Process

Steps	Operation	Description
1	Background	Read project descriptions, white papers, contract source code, and other relevant information the project team provides to ensure a proper understanding of project functions.
2	Automated testing	Scanning source code mainly with automated tools to find common potential vulnerabilities.
3	Manual reveiw	Engineers read the code line by line to find potential vulnerabilities.
4	Logical proofread	The engineer will compare the understanding of the code with the information provided by the project and check whether the code implementation is in line with the project white paper information.
5	Test case	Including test case design, test scope analysis, symbolic execution, etc.
6	Optimization items	Review of projects in terms of maintainability, safety, and operability based on application scenarios, deployment methods, and latest research results.

3. Risk Levels

Risk level	Issue description
Critical	Fatal risks and hazards that need to fixed immediately.
Major	Some high risks and hazards that will lead to related problems that must be solved
Medium	Some moderate risks and pitfalls may lead to potential risks that will eventually need to be addressed
Minor	There are low risks and hazards, mainly details of various types of mishandling or warning messages, which can be set aside for the time being
Information	Some parts can be optimized, such problems can be shelved, but it is recommended that the final solution

4. Audit Results



ID	Audit project	Risk level	Status
1	Reentrancy	None	
2	Injection	None	
3	Authentication bypass	None	
4	MEV Possibility	None	
5	Revert	None	
6	Race condition	None	
7	Insufficient Gas Griefing	None	
8	The major impact of flash loans	None	
9	Unreasonable economic model	None	
10	Predictable random numbers	None	
11	Voting rights management confusion	None	

ID	Audit project	Risk level	Status
12	Privacy leak	None	
13	Improper use of time on chain	None	
14	Improper codes in fallback function	None	
15	Improper identification	None	
16	Inappropriate opcode	None	
17	Inappropriate assembly	None	
18	Constructor irregularities	None	
19	Return value irregularity	None	
20	Event irregularity	None	
21	Keywords irregularity	None	
22	Not following ERC standards	None	
23	Irregularity of condition judgment	None	
24	Risk of liquidity drain	None	
25	Centralization Risk	None	
26	Logic change risk	None	
27	Integer overflow	None	
28	Improper function visibility	None	
29	Improper initialization of variables	None	
30	Improper contract calls	None	
31	Variable irregularities	None	
32	Replay	None	
33	Write to Arbitrary Storage Location	None	
34	Honeypot logic	None	
35	Hash collision	None	
36	Improper logic in receiving awards	None	
37	Use the not recommended method	None	
38	Basic coding principles were not followed	Medium	Acknowledged

*In the above table, if the status column is “**Acknowledged**”, the audit team has informed the project owner of the vulnerability. Still, the project owner has not made any changes to the vulnerability or has not announced to the audit team the progress of the changes to the vulnerability. If the status column is “**Resolved**”, the project owner has changed the exposure, and the audit team has confirmed the changes.

5. Risk and Modification Program

The following section provides detailed information about the risk items learned after the audit, including the type of risk, risk level, location of the issue, description of the problem, recommendations for changes, and feedback from the project owner.

1.Basic coding principles were not followed

Location	Contract file	Risk Status	Risk level
Line 8	LightNode.sol	⚠ Acknowledged	Medium

① Description

Importing a file name that does not exist will result in a compilation failure.

② Recommendation

It is recommended to remove the nonexistent import file name if there is no special logic design.

③ Code

```
JavaScript/**

import "@openzeppelin/contracts/proxy/utils/UUPSUpgradeable.sol";

import "@openzeppelin/contracts/proxy/utils/Initializable.sol";

import "../interface/ILightNode.sol";

//@OKLink Audit Description: import files not created

//@OKLink Audit Solution: Remove the statement


import "../interface/IWeightedMultiSig.sol";

import "../bls/BlsCode.sol";

import "../bls/BGLS.sol";

import "../interface/IVerifyTool.sol";
```


2. Basic coding principles were not followed

Location	Contract file	Risk Status	Risk level
Line 16~18	LightNode.sol	 Acknowledged	Medium

① Description

In solidity, state variables declared as "constant" or "immutable" are constants. The value of the constant modified by "constant" is determined at compile time, while the value of the constant modified by "immutable" is determined at deployment time. Try to use constants, which are part of the contract bytecode and do not occupy storage slots. Using constants saves gas more than variables. When deployed, constants consume less gas.

② Recommendation

It is recommended to change the keywords of these three lines of state variables to "constant"

③ Code

```
JavaScript/**
    * contract LightNode is UUPSUpgradeable,Initializable, ILightNode,BGLS {

//@OKLink Audit Description: 'constnt' is not used
//@OKLink Audit Solution: Use the "constant" keyword to identify constants

uint256 public maxValidators = 20;


uint256 public epochSize = 1000;

uint256 public headerHeight = 0;

address[] public validatorAddressss;

validator[maxValidators] public validators;
```

3. Basic coding principles were not followed

Location	Contract file	Risk Status	Risk level
Line 80	LightNode.sol	 Acknowledged	Medium

① Description

If the defined variable is not used in the code, or it is not used after assignment, it may lead to logical errors in coding or gas consumption in vain. Clearing unused variables is a good specification for coding, and it also helps to improve code readability and maintainability.

After the "hash" variable is defined, the assignment is completed through "verifytool.verifyheader (headerrlp)", but there is no subsequent call to "hash"

② Recommendation

It is recommended to check whether the code logic is correct. If it does not work, it is recommended to clear it.

③ Code

```
JavaScript/**

function verifyProofData(receiptProof memory _receiptProof)

    external

    view

    override

    returns (bool success, string memory message,bytes memory logsHash) {

        logsHash = verifyTool.encodeTxLog(_receiptProof.receipt.logs);

        (success, message) =verifyTool.getVerifyTrieProof(_receiptProof);

        if (!success) {

            message = "receipt mismatch";

            return (success, message,logsHash);

        }

        //@OKLink Audit Description: The local variable "hash" is not used,
and it is not used after the subsequent "hash" assignment

        //@OKLink Audit Solution: It is recommended to check whether the logic
is correct. If it is really useless, delete suggested

        bytes32 hash;

        bytes memory headerRlp =
verifyTool.encodeHeader(_receiptProof.header);

        (success, hash) = verifyTool.verifyHeader(headerRlp);

        if (!success) {

            message = "verifyHeader error";

            return (success, message,logsHash);

        }

        istanbulExtra memory ist =
verifyTool.decodeExtraData(_receiptProof.header.extraData);

        success = checkSig(_receiptProof.header, ist, _receiptProof.aggPk);

        if (!success) {

            message = "bls error";

        }

        return (success, message,logsHash);

    }
```

4. Basic coding principles were not followed

Location	Contract file	Risk Status	Risk level
Line 22	LightNode.sol	⚠ Acknowledged	Medium

① Description

The specification of initialization of state variables is to assign values in the initialization function or "constructor" function to avoid possible repeated assignments. Assigning values directly when defining state variables here will cause excessive gas consumption during contract deployment and may lead to subsequently repeated initialization. At the same time, directly creating a new contract object here does not meet the specification.

② Recommendation

It is recommended to use the interface to create or reference to avoid excessive gas consumption, or use the function of "Blscore" through the pipeline of Lib (Blscore is more like lib contract), instead of creating a Blscore contract for each LightNode contract.

③ Code

JavaScript/**

```
contract LightNode is UUPSUpgradeable,Initializable, ILightNode,BGLS {
```

```
    uint256 public maxValidators = 20;
```

```
    uint256 public epochSize = 1000;
```

```
    uint256 public headerHeight = 0;
```

```
    address[] public validatorAddressss;
```

```
    validator[maxValidators] public validators;
```

```
    IVerifyTool public verifyTool;
```

//@OKLink Audit Description: State variables should not be initialized at the time of definition

//@OKLink Audit Solution: Assign or inherit in the initialization function, and it is better to use address and interface to represent

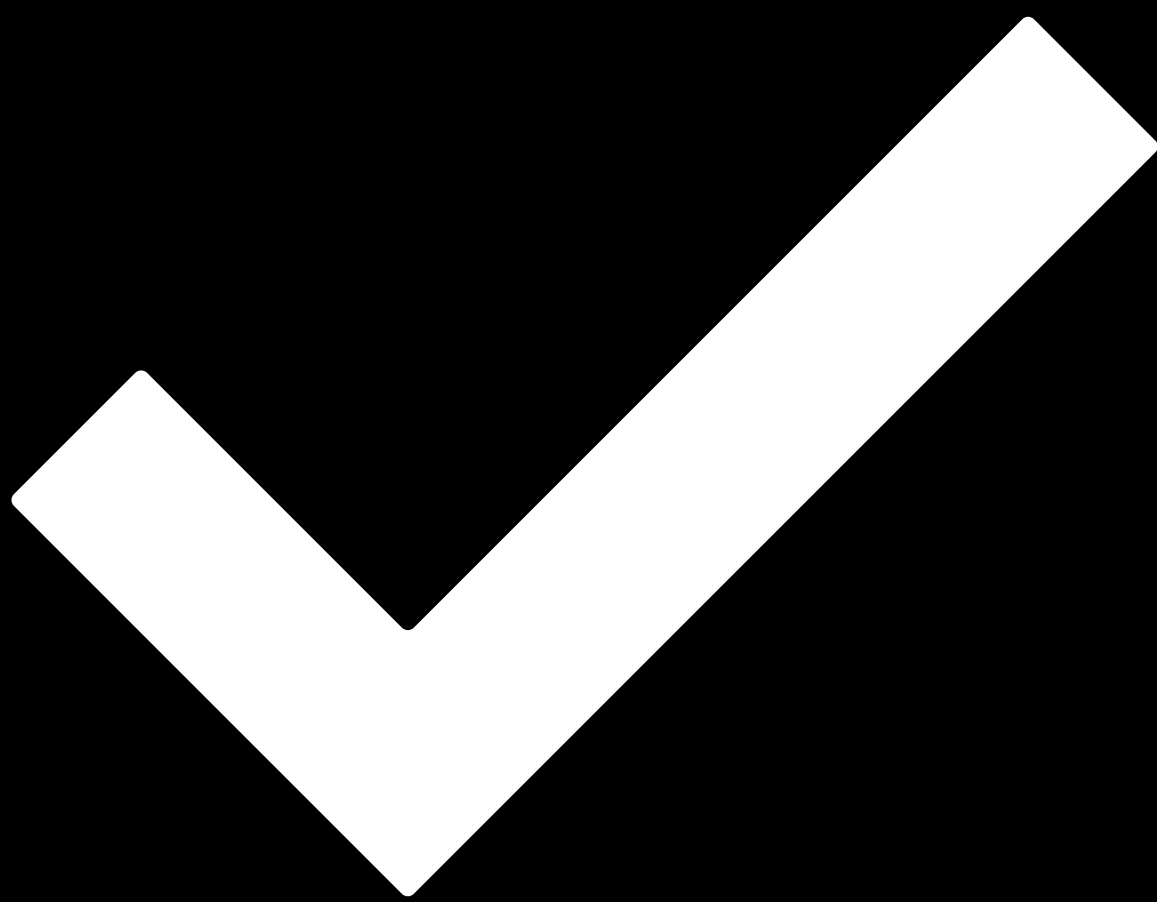
```
    BlsCode blsCode = new BlsCode();
```

```
    ~~~
```


Disclaimer

- i. This audit report focuses only on the types of audits identified in the final report issued. Other unknown security vulnerabilities are not part of this audit, and we do not accept responsibility for them.
- ii. We shall only issue an audit report based on an attack or vulnerability that existed or occurred before the issuance of the audit report. We cannot determine the likely impact on the security posture of our projects for new attacks or vulnerabilities that may exist or occur in the future, and we are not responsible for them.
- iii. The security audit analysis and other elements of our published audit report shall be based solely on documents and materials (including, but not limited to, contract codes) provided to us by the Project Party before the release of the audit report. Such documents and materials shall not be untrue, inaccurate, uninformative, altered, deleted, or concealed, and if the documents and materials provided by the Project Party are false, inaccurate, uninformative, changed, deleted or hidden, or if the documents and materials provided by the Project Party are untrue, inaccurate, uninformative, altered, deleted or concealed, or if the documents and materials provided by the Project Party are uninformative, uninformative, altered, deleted or hidden. If the records and information provided by the Project Party are untrue, inaccurate, uninformative, altered, deleted, or concealed, or if changes to such documents and information are made after the issuance of the audit report, we shall not be liable for any loss or adverse effect arising from any inconsistency between the reflected and actual conditions.
- iv. The Project Parties are aware that our audit report is based on documents and information provided by the Project Parties and relies on the technology currently available. However, due to the technical limitations of any organization, there is a possibility that our audit report may not fully detect all risks. Our audit team encourages the project development team and any interested parties to conduct subsequent testing and audits of the project.
- v. The project owner warrants that the project for which we are engaged to provide audit or testing services is legal, compliant, and does not violate applicable laws. The audit report is for the project owner's reference only, and the contents, manner of obtaining, use of, and any services or resources involved in the audit report shall not be relied upon for investment, tax, legal, regulatory, or advisory purposes of any kind, and we shall not be liable therefor. The Project Party shall not refer to, quote, display, or send the Audit Report in whole or in part to any third party without our prior written consent. The Project Party shall bear any loss or liability arising from that place. We assume no responsibility for any reliance on or use of the audit report for any purpose.
- vi. This audit report does not cover the compiler of the contract or any areas beyond the programming language of the Smart Contract. The risk and liability of the audited Smart Contract arising from references to off-chain information or resources is the sole responsibility of the project party.

- vii. Force Majeure. Force majeure means an unforeseen event whose occurrence and consequences cannot be avoided and cannot be overcome by the parties at the time of entering into the contract, including but not limited to natural disasters such as war, typhoon, flood, fire, earthquake, tidal wave, lightning, natural disaster, strike, nuclear explosion, epidemic and other unforeseen events such as changes in laws, regulations and policies and governmental acts, whose occurrence and consequences cannot be prevented or avoided, and which contains, affects or delays the performance by either party of all or part of its obligations under the contract.
- viii. Suppose either party believes that the occurrence of force majeure affects the performance of its obligations under this Agreement. In that case, it shall promptly notify the other party and, depending on the extent of the effect of the event on the performance of the Agreement; the parties shall consult to determine whether to terminate the Agreement or partially relieve itself of its obligations to perform the Agreement, or to extend the performance of the Agreement.
- ix. In force majeure, neither party shall be deemed in breach or non-performance of its obligations under this Agreement. Any financial commitments existing before the event shall not be affected, and the project party shall make payment for work performed by us.



Passed.

Date 11th July 2022

Audit Team 歐科雲鏈

The purpose of this audit is to review the ETH light node MPT verification protocol written by the MAP Protocol cross-link bridge project based on the solid language, study its design and architecture, find potential security risks, and try to find possible vulnerabilities.