CERTIK

Preliminary Comments

Aki Protocol

Feb 6th, 2022

Table of Contents

Summary

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

- 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.

Overview

Project Summary

Project Name	Aki Protocol
Platform	bsc
Language	Solidity
Codebase	https://bscscan.com/address/0x10586780266E26633a3177cE966DFBE838E9404D
Commit	

Audit Summary

Delivery Date	Feb 06, 2022
Audit Methodology	Static Analysis, Manual Review

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Mitigated	Resolved
Critical	0	0	0	0	0	0	0
Major	4	4	0	0	0	0	0
Medium	1	1	0	0	0	0	0
Minor	2	2	0	0	0	0	0
Informational	9	9	0	0	0	0	0
Discussion	1	1	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
AAC	Address.sol	0228dd7c0a0d1342b88eab6e5a4a07ae4350818ba1650be0a374064b02218f37
CAC	Context.sol	1458c260d010a08e4c20a4a517882259a23a4baa0b5bd9add9fb6d6a1549814a
ECD	ECDSA.sol	98343b7fb83ad932ba41de40411c65f1319654d94905b5564e82a74ee59b7ace
IER	IERC20.sol	6eacf8ca56b41b7636489c0996d8f9608b4d298879a1fd5d876f21ad7a6711f1
IEA	IEnvelope.sol	fe898e427d5c3754a23464baf12bda260fb4c329d9a134481f41480c94ca37f3
MPA	MerkleProof.sol	044f599575ea42c1d720f82576480c69bc994d2e744ca41ed437b1fd8c545d3f
OAC	Ownable.sol	75e3c97011e75627ffb36f4a2799a4e887e1a3e27ed427490e82d7b6f51cc5c9
REM	RedEnvelopeMerkleERC20.sol	a3876c6c3dcb3e5c0c9a7e2bd93b31bc46a81cbea3bd2edd8304e2c95cb34b80
RGA	ReentrancyGuard.sol	aa73590d5265031c5bb64b5c0e7f84c44cf5f8539e6d8606b763adac784e8b2e
SER	SafeERC20.sol	b5a1340c5232f387b15592574f27eef78f6017bdc66542a1cea512ad4f78a0d2
SAC	Strings.sol	8597c62818dcbc6cf85c21179b90b714fb4f70a4347ca2eed23e88c87b08b8a1

Review Notes

The **Aki Protocol** team creates a red envelope application for ERC20 tokens using Merkle tree verifications. Users can add a red envelope for participants with valid passwords to open to gain a random amount of tokens.

External Dependencies

There are a few depending injection contracts or addresses in the current project:

- `signer_` for the contract `IEnvelope`;
- `MerkleERC20Envelope.token` and `approvedTokens` for the `RedEnvelopeMerkleERC20` contract.

We assume these contracts or addresses are valid and non-vulnerable actors and implement proper logic to collaborate with the current project.

Privileged Functions

To set up the project correctly, improve overall project quality and preserve upgradability, the following roles are adopted in the codebase.

In the contract `Ownable`, the role `_owner` has authority over the following functions:

- `Ownable.renounceOwnership()` to renounce the ownership;
- `Ownable.transferOwnership()` to transfer the ownership.

In the contract IEnvelope.sol the role `_owner` has authority over the functions below:

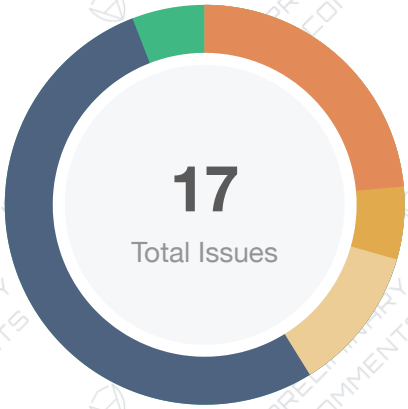
- `IEnvelope.setSigner()` to change the signer address.

In the contract `RedEnvelopeMerkleERC20`, the role `_owner` has authority over the following functions:

- `RedEnvelopeMerkleERC20.approveToken()` to approve a token to be acceptible for creating red envelopes.

To improve the trustworthiness of the project, dynamic runtime updates in the project should be notified to the community. Any plan to invoke the aforementioned functions should be also considered to move to the execution queue of timelock contract.

Findings



Critical	0 (0.00%)
Major	4 (23.53%)
Medium	1 (5.88%)
Minor	2 (11.76%)
Informational	9 (52.94%)
Discussion	1 (5.88%)

ID	Title	Category	Severity	Status
GLOBAL-01	Unlocked Compiler Version	Language Specific	Informational	Pending
AAC-01	Ineffective `isContract()` Check	Volatile Code	Minor	Pending
IEA-01	Centralization Related Risks	**Centralization / Privilege**	**Major**	Pending
IEA-02	Pseudo-random Process	Logical Issue	Major	Pending
IEA-03	Missing Emit Events	Coding Style	Informational	Pending
IEA-04	Unspecified Integer Size	Language Specific, Coding Style	Informational	Pending
IEA-05	Unable to Open an Envelope with Large Balance	Logical Issue	Informational	Pending
IEA-06	Local Variables Could Be Declared `constant` State Variable	Gas Optimization	Informational	Pending
IEA-07	Unused Functions and Structs	Gas Optimization	Informational	Pending
OAC-01	Centralization Related Risks	**Centralization / Privilege**	**Major**	Pending
REM-01	Centralization Related Risks	**Centralization / Privilege**	**Major**	Pending
REM-02	Missing Input Validation	Volatile Code	Medium	Pending

ID	Title	Category	Severity	Status
REM-03	Incompatibility With Deflationary Tokens	Logical Issue	Minor	Pending
REM-04	Unnecessary `payable` Keyword	Coding Style, Logical Issue	Informational	Pending
REM-05	Missing Emit Events	Coding Style	Informational	Pending
REM-06	Function Visibility Optimization	Gas Optimization	Informational	Pending
REM-07	Validations on the `numParticipants` and Available Passwords	Logical Issue	Discussion	Pending

GLOBAL-01 | Unlocked Compiler Version

Category	Severity	Location	Status
Language Specific	Informational	Global	Pending

Description

The contract has an unlocked compiler version. An unlocked compiler version in the source code of the contract permits the user to compile it at or above a particular version. This, in turn, leads to differences in the generated bytecode between compilations due to differing compiler version numbers. This can lead to ambiguity when debugging as compiler-specific bugs may occur in the codebase that would be hard to identify over a span of multiple compiler versions rather than a specific one.

Recommendation

We advise that the compiler version is instead locked at the lowest version possible that the contract can be compiled at. For example, for version `v0.6.2` the contract should contain the following line:

```
pragma solidity 0.6.2;
```

AAC-01 | Ineffective `isContract()` Check

Category	Severity	Location	Status
Volatile Code	Minor	Address.sol: 27	Pending

Description

The implementation of the `isContract()` check can not cover all scenarios. The check can be bypassed if the call is from the constructor of a smart contract or when the contract is destroyed. Because, in that case, the codesize will also be zero.

The "isContract()" function in the OpenZeppelin "Address" library uses the same implementation, but comments mention that "it's unsafe to rely on the check and it can be bypassed". Reference: https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/utils/Address.sol

Recommendation

It is recommended to add the additional `msg.sender == tx.origin` check to cover all the scenarios. Do note that the check still works for the current EVM (London) version, but future updates to the EVM or EIP (ex. EIP-3074) might cause the check to become ineffective.

```
modifier notContract() {
    require((!_isContract(msg.sender)) && (msg.sender == tx.origin), "contract not
allowed");
    _;
}

function _isContract(address addr) internal view returns (bool) {
    uint256 size;
    assembly {
        size := extcodesize(addr)
    }
    return size > 0;
}
```

IEA-01 | Centralization Related Risks

Category	Severity	Location	Status
Centralization / Privilege	Major	IEnvelope.sol: 197	Pending

Description

In the contract IEnvelope.sol the role `_owner` has authority over the functions below:

- `IEnvelope.setSigner()` to change the signer address.

Any compromise to the `_owner` account may allow a hacker to take advantage of this authority and manipulate the protocol.

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multi-signature wallets.

Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement;
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles;
  OR
- Remove the risky functionality.

Noted: Recommend considering the long-term solution or the permanent solution. The project team shall make a decision based on the current state of their project, timeline, and project resources.

IEA-02 | Pseudo-random Process

Category	Severity	Location	Status
Logical Issue	Major	IEnvelope.sol: 260~270	Pending

Description

In the code snippet below, the random variable `rand` is generated by using the `block.difficulty`, `block.timestamp`, and `receiver`, which can be precalculated locally or on-chain since the `getMoneyThisOpen()` function is marked as `public`. Unethical players can guess the best timing by the trial-error method and significantly increase the chance to win big, which may potentially break the game.

```
260         uint16 rand = uint16(
261             uint256(
262                 keccak256(
263                     abi.encodePacked(
264                         block.difficulty,
265                         block.timestamp,
266                         receiver
267                     )
268                 )
269             )
270         );
```

Recommendation

Consider using Chainlink VRF (Verifiable Random Function) as a provably-fair and verifiable source of randomness.

IEA-03 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	Informational	IEnvelope.sol: 197	Pending

Description

The function that affects the status of sensitive variables should be able to emit events as notifications to users.

- `IEnvelope.setSigner()` to change the signer address

Recommendation

Consider adding events for sensitive actions, and emit them in the function.

IEA-04 | Unspecified Integer Size

Category	Severity	Location	Status
Language Specific, Coding Style	Informational	IEnvelope.sol: 170	Pending

Description

The `timestamp` is declared as `uint` without specifying the integer size. Although by default `uint` serves the same as `uint256`, leaving the integer size unspecified leads the code base to be less organized and unified. In addition, it is prone to error when the Solidity version iterates over time.

Recommendation

Consider explicitly specifying the integer size.

IEA-05 | Unable To Open An Envelope With Large Balance

Category	Severity	Location	Status
Logical Issue	Informational	IEnvelope.sol: 275	Pending

Description

It is possible that when the `randBalance` is large enough, the following calculation would revert due to integer overflow, which results in an un-openable envelope.

```
275         uint256 moneyThisOpen = ((maxThisOpen * rand1K) / 1000) + minPerOpen;
```

Recommendation

Consider checking the status of `maxThisOpen * rand1K` first. If it overflows, the calculation could apply division first then multiplication.

IEA-06 | Local Variables Could Be Declared `constant` State Variable

Category	Severity	Location	Status
Gas Optimization	Informational	IEnvelope.sol: 231	Pending

Description

The local variable `MAX_INT` is a constant integer value declared inside the function `IEnvelope.hashPassword()`, which could be declared as a `constant` state variables for gas optimization in a case when the function `IEnvelope.hashPassword()` is frequently called upon.

Recommendation

Consider declaring the variable as a `constant` state variable.

IEA-07 | Unused Functions And Structs

Category	Severity	Location	Status
Gas Optimization	Informational	IEnvelope.sol: 158~161, 163~171, 173~183, 185~195, 201, 219, 226	Pending

Description

The following functions are never used:

- `IEnvelope.recover(bytes calldata signature, string calldata unhashedPassword)`,
- `IEnvelope.initStatus()`,
- `IEnvelope.hashPassword(string memory unhashedPassword)`.

The following Structs are never used:

- `Status` on line 158,
- `Envelope` on line 163,
- `MerkleEnvelope` on line 173,
- `MerkleEnvelopeERC721` on line 185.

Recommendation

Consider removing unused functions and structs for gas optimization.

OAC-01 | Centralization Related Risks

Category	Severity	Location	Status
Centralization / Privilege	**Major**	Ownable.sol: 54, 62	Pending

Description

In the contract `Ownable`, the role `_owner` has authority over the following functions:

- `Ownable.renounceOwnership()` to renounce the ownership;
- `Ownable.transferOwnership()` to transfer the ownership.

Any compromise to the `_owner` account may allow a hacker to take advantage of this authority and manipulate the protocol.

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multi-signature wallets.

Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement;
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles;
  OR
- Remove the risky functionality.

Noted: Recommend considering the long-term solution or the permanent solution. The project team shall make a decision based on the current state of their project, timeline, and project resources.

REM-01 | Centralization Related Risks

Category	Severity	Location	Status
Centralization / Privilege	Major	RedEnvelopeMerkleERC20.sol: 45	Pending

Description

In the contract `RedEnvelopeMerkleERC20`, the role `_owner` has authority over the following functions:

- `RedEnvelopeMerkleERC20.approveToken()` to approve a token to be acceptible for creating red envelopes.

Any compromise to the `_owner` account may allow a hacker to take advantage of this authority and manipulate the protocol.

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multi-signature wallets.

Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement;
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles;
  OR
- Remove the risky functionality.

Noted: Recommend considering the long-term solution or the permanent solution. The project team shall make a decision based on the current state of their project, timeline, and project resources.

REM-02 | Missing Input Validation

Category	Severity	Location	Status
Volatile Code	Medium	RedEnvelopeMerkleERC20.sol: 77~82	Pending

Description

The function `RedEnvelopeMerkleERC20.openEnvelope()` verify the password by passing in the `proof` and `leaf` value to `MerkleProof.verify()`. However, it is recommended to avoid using leaf values that are 64 bytes long prior to hashing, or use a hash function other than keccak256 for hashing leaves. This is because the concatenation of a sorted pair of internal nodes in the merkle tree could be reinterpreted as a leaf value. Hence, a malicious attacker could fake a leaf node by concatenation of a sorted pair of parent nodes to exploit the protocol.

Recommendation

Consider hashing the leaf value inside the contract using `keccak256` and validating the leaf values that are less than 64 bytes long prior to hashing.

REM-03 | Incompatibility With Deflationary Tokens

Category	Severity	Location	Status
Logical Issue	Minor	RedEnvelopeMerkleERC20.sol: 11	Pending

Description

When transferring standard ERC20 deflationary tokens, the input amount may not be equal to the received amount due to the charged transaction fee. For example, if a user adds an envelope with 100 deflationary tokens (with a 10% transaction fee) in `RedEnvelopeMerkleERC20.addEnvelope()`, only 90 tokens actually arrived in the contract. However, the user can still withdraw 100 tokens from the contract, which causes the contract to lose 10 tokens in such a transaction.

Recommendation

Consider regulating the set of approved tokens supported and add necessary mitigation mechanisms to keep track of accurate balances if there is a need to support deflationary tokens.

REM-04 | Unnecessary `payable` Keyword

Category	Severity	Location	Status
Coding Style, Logical Issue	Informational	RedEnvelopeMerkleERC20.sol: 38	Pending

Description

The idea behind the distinction of `address` and `address payable` is that `address payable` is an address that can be sent Ether to, while a plain `address` cannot be sent Ether. In the current codebase, Ether is not involved in the contract logic since the red pocket only accepts approved `ERC20` tokens.

Recommendation

Consider removing the `payable` keyword.

REM-05 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	Informational	RedEnvelopeMerkleERC20.sol: 45	Pending

Description

The function that affects the status of sensitive variables should be able to emit events as notifications to users.

- `RedEnvelopeMerkleERC20.approveToken()` to approve a token to be acceptible for creating red envelopes

Recommendation

Consider adding events for sensitive actions, and emit them in the function.

REM-06 | Function Visibility Optimization

Category	Severity	Location	Status
Gas Optimization	Informational	RedEnvelopeMerkleERC20.sol: 30, 45, 49~57, 77~82	Pending

Description

The following functions are declared as `public` and are not invoked in any of the contracts contained within the project's scope. The functions that are never called internally within the contract should have external visibility.

Recommendation

The audit team advises that the functions' visibility specifiers are set to `external`, optimizing the gas cost of the function.

REM-07 | Validations On The `numParticipants` And Available Passwords

Category	Severity	Location	Status
Logical Issue	Discussion	RedEnvelopeMerkleERC20.sol: 73	Pending

Description

In order to open a red envelope, participants must provide a valid password. After opening the red envelope, the remaining allowed participant number is reduced by 1. In the current logic, `MerkleERC20Envelope.balance >= MerkleERC20Envelope.minPerOpen * MerkleERC20Envelope.numParticipants` and it guarantees that every participant opens the envelope with a `minPerOpen`.

However, there is no clear restriction on the `numParticipants` and available passwords to claim. If the number of available passwords is smaller than the `numParticipants`, the claimed `numParticipants` would be incorrect. We could like to confirm it is the intended design.

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

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.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete.

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

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.

About

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.