CERTIK

Aergo (HPP) - Vesting

Security Assessment

CertiK Assessed on Feb 3rd, 2026

CertiK Assessed on Feb 3rd, 2026

Aergo (HPP) - Vesting

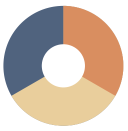
The security assessment was prepared by CertiK.

Executive Summary

TYPES	ECOSYSTEM	METHODS
Staking	EVM Compatible	Formal Verification, Manual Review, Static Analysis

LANGUAGE	TIMELINE
Solidity	Preliminary comments published on 12/02/2025 Final report published on 02/03/2026

Vulnerability Summary



Total Findings	Resolved	Partially Resolved	Acknowledged	Declined
3	0	0	3	0

🟧	1	Centralization	1 Acknowledged	Centralization findings highlight privileged roles & functions and their capabilities, or instances where the project takes custody of users' assets.
🟥	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.
🟧	0	Major		Major risks may include logical errors that, under specific circumstances, could result in fund losses or loss of project control.
🟨	0	Medium		Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.
🟨	1	Minor	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.
🟦	1	Informational	1 Acknowledged	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 | AERGO (HPP) - VESTING

CODEBASE | AERGO (HPP) - VESTING

Repository

123f5838bcee7c6c13734b4ec30ff5ec2155af2b

1f5322f82e706dc4c0bd4f829657bfc6e5852331

Commit

123f5838bcee7c6c13734b4ec30ff5ec2155af2b

1f5322f82e706dc4c0bd4f829657bfc6e5852331

AUDIT SCOPE | AERGO (HPP) - VESTING

hpp-io/contracts
contracts/HPP_Vesting.sol
HPP_Vesting_AIP21.sol

APPROACH & METHODS | AERGO (HPP) - VESTING

This audit was conducted for Aergo (HPP) to evaluate the security and correctness of the smart contracts associated with the Aergo (HPP) - Vesting project. The assessment included a comprehensive review of the in-scope smart contracts. The audit was performed using a combination of Static Analysis, Formal Verification, and Manual Review.

The review process emphasized the following areas:

- Architecture review and threat modeling to understand systemic risks and identify design-level flaws.
- Identification of vulnerabilities through both common and edge-case attack vectors.
- Manual verification of contract logic to ensure alignment with intended design and business requirements.
- Dynamic testing to validate runtime behavior and assess execution risks.
- Assessment of code quality and maintainability, including adherence to current best practices and industry standards.

The audit resulted in findings categorized across multiple severity levels, from informational to critical. To enhance the project's security and long-term robustness, we recommend addressing the identified issues and considering the following general improvements:

- Improve code readability and maintainability by adopting a clean architectural pattern and modular design.
- Strengthen testing coverage, including unit and integration tests for key functionalities and edge cases.
- Maintain meaningful inline comments and documentations.
- Implement clear and transparent documentation for privileged roles and sensitive protocol operations.
- Regularly review and simulate contract behavior against newly emerging attack vectors.

REVIEW NOTES | AERGO (HPP) - VESTING

Overview

The `HPP_Vesting_AIP21` contract implements a linear vesting program. It manages the distribution of HPP tokens to multiple beneficiaries over a fixed 24-month duration.

Key features include:

- **Global Vesting Clock**: Vesting follows a single global `vestingStartTime` for all beneficiaries.
- **Linear Distribution**: Tokens vest linearly over a 730-day period (approx. 24 months) starting from the global start time.
- **Owner-Managed Registry**: The owner is responsible for initializing the start time and adding beneficiaries (individually or in batches).
- **Retroactive Allocation**: Schedules can only be added *after* the vesting start time is set, meaning new beneficiaries immediately have a portion of tokens vested based on the time elapsed since the start.
- **Revocability**: The owner can revoke a beneficiary's schedule, effectively cancelling their right to claim any further tokens (including those already vested but unclaimed).
- **Emergency Controls**: The contract includes mechanisms for the owner to withdraw funds.

External Dependencies

- **OpenZeppelin**: `IERC20` , `SafeERC20` , `EnumerableSet` , `Ownable` , `ReentrancyGuard` .
- The contract integrates with a specific ERC20 token (`hppToken`) set at deployment. The token is expected to follow standard ERC20 behavior.

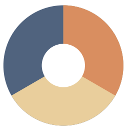
Privileged Functions

Functions callable only by the owner that significantly impact contract state and funds:

- `startVesting` : Permanently sets the global `vestingStartTime` and enables the addition of schedules.
- `addVestingSchedule` , `addVestingSchedules` : Assigns token allocations to beneficiaries. Requires vesting to be started.
- `revokeVestingSchedule` : Deactivates a beneficiary's schedule. This prevents any future claims, including tokens that calculated as vested based on time but were not yet claimed.
- `emergencyWithdraw` , `emergencyWithdrawAll` : Allows the owner to sweep tokens from the contract to the owner address. This capability effectively allows the owner to override token reservations for vesting schedules.

Because the owner has absolute control over creating schedules, revoking claims, and draining contract funds, the owner account represents a single point of failure and should be secured via a multisig or timelock.

FINDINGS | AERGO (HPP) - VESTING



3	0	1	0	0	1	1
Total Findings	Critical	Centralization	Major	Medium	Minor	Informational

This report has been prepared for Aergo (HPP) to identify potential vulnerabilities and security issues within the reviewed codebase. During the course of the audit, a total of 3 issues were identified. Leveraging a combination of Static Analysis, Formal Verification & Manual Review the following findings were uncovered:

ID	Title	Category	Severity	Status
AHV-01	**Centralization Risks In Source**	**Centralization**	**Centralization**	● **Acknowledged**
AHV-03	Global Vesting Timeline Grants Late-Added Beneficiaries Immediate Vesting	Logical Issue	Minor	● Acknowledged
AHV-02	Backdated Vesting Start Lets Late Beneficiaries Claim Allocation Immediately	Design Issue	Informational	● Acknowledged

AHV-01 | Centralization Risks In Source

Category	Severity	Location	Status
Centralization	● Centralization	**HPP_Vesting_AIP21.sol (HPP_Vesting_AIP21.sol): 84, 100, 128, 213, 232, 243**	● Acknowledged

Description

In the contract `HPP_Vesting_AIP21` the role `_owner` has authority over the functions shown in the diagram below (`startVesting` , `addVestingSchedule` , `addVestingSchedules` , `revokeVestingSchedule` , `emergencyWithdraw` and `emergencyWithdrawAll`). Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and add for anyone vestingSchedules or withdraw all the tokens held by this contract.

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., multisignature 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.

Alleviation

[Aergo (HPP), 12/03/2025]: The team acknowledged the issue and decided not to implement the recommended change in the current engagement.

AHV-03 Global Vesting Timeline Grants Late-Added Beneficiaries Immediate Vesting

Category	Severity	Location	Status
Logical Issue	● Minor	contracts/HPP_Vesting.sol (20260128): 34~37, 134	● Acknowledged

Description

The vestingStartTime `/` vestingDuration` are global immutables, and schedules never record their own timing.

```
34      uint256 public immutable vestingStartTime;
35
36      /// @notice Vesting duration (in seconds)
37      uint256 public immutable vestingDuration;
```

The function `_getVestedAmount` relies on the global clock,

```
165     function _getVestedAmount(VestingSchedule storage schedule) internal view
returns (uint256) {
166         if (block.timestamp < vestingStartTime) {
167             return 0;
168         }
169
170         if (block.timestamp >= vestingStartTime + vestingDuration) {
171             return schedule.totalAmount;
172         }
173
174         // Linear vesting calculation (proportional to elapsed time)
175         return (schedule.totalAmount * (block.timestamp - vestingStartTime)) /
 vestingDuration;
176     }
```

So any beneficiary added after the global start—or after the full duration—has their entire allocation considered vested and can claim rewards immediately, defeating staggered or per-user vesting.

Recommendation

Store per-schedule start/duration in `VestingSchedule` and use them in the vesting math, or if a single shared timeline is intended, reject adding schedules after `vestingStartTime` so late additions cannot claim instantly.

Alleviation

[Aergo (HPP), 02/03/2026]: The team acknowledged the issue and decided not to implement the recommended change in the current engagement, providing the following statement:

"Issue acknowledged. I won't make any changes for the current version.

The vesting contract is intentionally designed such that all beneficiaries share the same vesting start time, and vesting for all participants is intended to commence simultaneously.

The ability to add vesting schedules after deployment exists solely as an operational safeguard to address cases where beneficiaries, who were intended to be included at the initial vesting start, were inadvertently omitted. In such cases, the operator may add the omitted beneficiaries at a later time while preserving the original vesting intent.

As this functionality serves a necessary operational purpose, removing or fully disabling the ability to add vesting schedules after vestingStartTime would prevent remediation of such omissions and is therefore not aligned with the intended operational requirements of the system.

Administrative privileges for this function are managed through a multisig wallet, and its usage is governed by internal operational policies to ensure it is exercised only for corrective purposes."

AHV-02 Backdated Vesting Start Lets Late Beneficiaries Claim Allocation Immediately

Category	Severity	Location	Status
Design Issue	● Informational	HPP_Vesting_AIP21.sol (HPP_Vesting_AIP21.sol): 109~116	● Acknowledged

Description

`_addVestingSchedule` stamps every new schedule with the original `vestingStartTime` . Because adding schedules is only permitted after `vestingStarted` is true, any beneficiary onboarded after TGE inherits the past start time, and `_getVestedAmount` treats the entire elapsed period as vested, allowing near-instant withdrawal.

```
    function _addVestingSchedule(address _beneficiary, uint256 _amount) private {
        require(_beneficiary != address(0), "Invalid beneficiary address");
        require(_amount > 0, "Amount must be greater than 0");
        require(!vestingSchedules[_beneficiary].isActive, "Vesting schedule already
exists");
        require(vestingStarted && vestingStartTime != 0, "Vesting not started");
        vestingSchedules[_beneficiary] = VestingSchedule({
            beneficiary: _beneficiary,
            totalAmount: _amount,
            claimedAmount: 0,
            startTime: vestingStartTime,
            duration: VESTING_DURATION,
            isActive: true
        });
        beneficiaries.add(_beneficiary);
        totalVestingAmount += _amount;
        emit VestingScheduleAdded(_beneficiary, _amount, vestingStartTime);
    }
```

Recommendation

(If this is not a intended design)Decouple individual schedule start times from the global start. Either (a) allow schedules to be created before `startVesting` so all beneficiaries share the same true start, or (b) set each schedule's `startTime` when it is created (e.g., `block.timestamp`) and optionally adjust duration or cliff to reflect program rules. Additionally, enforce per-beneficiary cliffs if uniform start alignment is required.

Alleviation

[Aergo (HPP), 12/03/2025]: The team acknowledged the issue and confirmed this is an intended design.

FORMAL VERIFICATION | AERGO (HPP) - VESTING

Formal guarantees about the behavior of smart contracts can be obtained by reasoning about properties relating to the entire contract (e.g. contract invariants) or to specific functions of the contract. Once such properties are proven to be valid, they guarantee that the contract behaves as specified by the property. As part of this audit, we applied formal verification to prove that important functions in the smart contracts adhere to their expected behaviors.

Considered Functions And Scope

In the following, we provide a description of the properties that have been used in this audit. They are grouped according to the type of contract they apply to.

Verification of Standard Ownable Properties

We verified *partial* properties of the public interfaces of those token contracts that implement the Ownable interface. This involves:

- function `owner` that returns the current owner,
- functions `renounceOwnership` that removes ownership,
- function `transferOwnership` that transfers the ownership to a new owner.

The properties that were considered within the scope of this audit are as follows:

Property Name	Title
ownable-transferownership-correct	Ownership is Transferred
ownable-owner-succeed-normal	`owner` Always Succeeds
ownable-renounceownership-correct	Ownership is Removed
ownable-renounce-ownership-is-permanent	Once Renounced, Ownership Cannot be Regained

Verification Results

In the remainder of this section, we list all contracts where formal verification of at least one property was not successful. There are several reasons why this could happen:

- False: The property is violated by the project.
- Inconclusive: The proof engine cannot prove or disprove the property due to timeouts or exceptions.
- Inapplicable: The property does not apply to the project.

Detailed Results For Contract HPP_Vesting (contracts/HPP_Vesting.sol) In Commit 1f5322f82e706dc4c0bd4f829657bfc6e5852331

Verification of Standard Ownable Properties

Detailed Results for Function `transferOwnership`

Property Name	Final Result	Remarks
ownable-transferownership-correct	● True	

Detailed Results for Function `owner`

Property Name	Final Result	Remarks
ownable-owner-succeed-normal	● True	

Detailed Results for Function `renounceOwnership`

Property Name	Final Result	Remarks
ownable-renounceownership-correct	● True	
ownable-renounce-ownership-is-permanent	● Inconclusive	

APPENDIX | AERGO (HPP) - VESTING

Finding Categories

Categories	Description
Logical Issue	Logical Issue findings indicate general implementation issues related to the program logic.
Centralization	Centralization findings detail the design choices of designating privileged roles or other centralized controls over the code.
Design Issue	Design Issue findings indicate general issues at the design level beyond program logic that are not covered by other finding categories.

Details on Formal Verification

Some Solidity smart contracts from this project have been formally verified. Each such contract was compiled into a mathematical model that reflects all its possible behaviors with respect to the property. The model takes into account the semantics of the Solidity instructions found in the contract. All verification results that we report are based on that model.

The following assumptions and simplifications apply to our model:

- Certain low-level calls and inline assembly are not supported and may lead to a contract not being formally verified.
- We model the semantics of the Solidity source code and not the semantics of the EVM bytecode in a compiled contract.

Formalism for property specifications

All properties are expressed in a behavioral interface specification language that CertiK has developed for Solidity, which allows us to specify the behavior of each function in terms of the contract state and its parameters and return values, as well as contract properties that are maintained by every observable state transition. Observable state transitions occur when the contract's external interface is invoked and the invocation does not revert, and when the contract's Ether balance is changed by the EVM due to another contract's "self-destruct" invocation. The specification language has the usual Boolean connectives, as well as the operator `\old` (used to denote the state of a variable before a state transition), and several types of specification clause:

Apart from the Boolean connectives and the modal operators "always" (written `[]`) and "eventually" (written `<>`), we use the following predicates to reason about the validity of atomic propositions. They are evaluated on the contract's state whenever a discrete time step occurs:

- `requires [cond]` - the condition `cond`, which refers to a function's parameters, return values, and contract state variables, must hold when a function is invoked in order for it to exhibit a specified behavior.
- `ensures [cond]` - the condition `cond`, which refers to a function's parameters, return values, and both `\old` and current contract state variables, is guaranteed to hold when a function returns if the corresponding requires condition held

when it was invoked.

- `invariant [cond]` - the condition `cond`, which refers only to contract state variables, is guaranteed to hold at every observable contract state.
- `constraint [cond]` - the condition `cond`, which refers to both `\old` and current contract state variables, is guaranteed to hold at every observable contract state except for the initial state after construction (because there is no previous state); constraints are used to restrict how contract state can change over time.

Description of the Analyzed Ownable Properties

Properties related to function `transferOwnership`

ownable-transferownership-correct

Invocations of `transferOwnership(newOwner)` must transfer the ownership to the `newOwner`.

Specification:

```
ensures this.owner() == newOwner;
```

Properties related to function `owner`

ownable-owner-succeed-normal

Function `owner` must always succeed if it does not run out of gas.

Specification:

```
reverts_only_when false;
```

Properties related to function `renounceOwnership`

ownable-renounce-ownership-is-permanent

The contract must prohibit regaining of ownership once it has been renounced.

Specification:

```
constraint \old(owner()) == address(0) ==> owner() == address(0);
```

ownable-renounceownership-correct

Invocations of `renounceOwnership()` must set ownership to address(0).

Specification:

```
ensures this.owner() == address(0);
```

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.

Elevating Your Web3 Journey

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is the largest 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.