ANTE PROTOCOL V0.5 LITEPAPER

hello@ante.finance

Abstract. Ante is a smart contract testing protocol that constructs a new primitive allowing anyone to express confidence or doubt about the success or failure of live, on-chain smart contract tests. We explain the design of Ante Protocol v0.5 and show how to use Ante to signal pseudonymous confidence and provide additional live protection to protocols.

Contents

1. Introduction

1.1. **Motivation.** Interest in decentralized finance (DeFi) has increased over 100x since 2020, with over $100B total value locked (TVL) in DeFi smart contracts. The increased interest involves two key factors: first, the emergence and popularization of permissionless smart contract systems offering economically productive on-chain uses of digital assets, such as lending (Aave [1], Compound [6]) and automated liquidity provision (Uniswap [2], Curve [4]); second, the advent of liquidity mining, a user incentive mechanism where projects award early users with tokens for usage.

A problem gating further growth for DeFi is that smart contracts are by nature subject to potential loss of user funds. This can occur via a bug in or hack of smart contract code or an economic exploit of a poorly designed protocol mechanism. Further, many project teams are pseudonymous, opening the possibility that the team behind a protocol may be dishonest and use a hidden exploit or back door to steal user funds. For many users, these uncertainties add unnecessary risk, and users have unexpectedly lost funds in many projects in 2020, such as bZX ($8m+), Yam, dForce ($25m), Eminence ($15m), Harvest ($24m), Pickle ($19m), and more [7].

Date: June 2021.

DeFi lacks clarity around its security and risks, and Ante Protocol was designed to provide a market-based solution to clarify user trust in DeFi protocols.

1.2. **Summary of the Ante Protocol.** Ante (Autonomous Native Testing Environment) Protocol is a smart contract testing protocol on Ethereum. Ante allows anyone to create and deploy Ante Tests, which are live, on-chain tests that confirm a target protocol's invariants are valid. DeFi protocol developers and project supporters (such as users or investors) can signal confidence in the protocol's robustness by staking on Ante Pools corresponding to these tests. Security experts can monetize their expertise by identifying flawed Ante Tests or broken invariants by challenging the corresponding Ante Pools and verifying the Ante Test's status. If a failure is confirmed by the Ante Test, the funds staking an Ante Pool are paid pro rata to pool challengers, with a bonus for the challenger who initiated the test. The remainder of the time, challengers' balances experience an ongoing decay that is claimable by stakers pro rata.

Because staked funds are at risk if tests fail, the total amount of staking capital and its ratio to challenging capital reflects a protocol's perceived stability. This gives users an additional signal about which DeFi protocols are legitimate by allowing them to compare the amount of capital staked behind similar protocols. In addition, users can verify which protocols have responsibly written and deployed live tests, which provide additional protection for secure smart contract systems beyond ordinary static, off-chain testing. In this way, Ante makes it easier for DeFi protocol developers to bootstrap legitimacy and trust in their protocols.

2. System Design

The Ante Protocol allows protocols on Ethereum to signal confidence in their code by putting "skin in the game" behind programmatically checked properties of their contract state, referred to as **on-chain invariants**. We briefly describe the mechanism below and devote the rest of this section to a detailed description.

To use Ante for a given protocol, developers codify an invariant in a smart contract following the `AnteTest` interface which contains a function checking on-chain whether the invariant currently holds. They then generate an Ante Pool instance for that Ante Test following the `IAnteTest` interface using the `AntePoolFactory`. The Ante Pool allows anyone to deposit capital, either as a **staker** (to express the belief that the Ante Test will always pass) or as a **challenger** (to express the belief that the Ante Test may fail).

At any time, any challenger can **verify** on-chain if the Ante Test still passes. If such a check fails, then stakers' funds are frozen and distributed to challengers and the test verifier. Meanwhile, challengers pay an ongoing **decay reward** to stakers via the Ante Pool.

2.1. **On-chain invariants and Ante Tests.** Ante Tests test on-chain invariants, which are boolean statements which can make reference to *any* data in the current Ethereum state. Ante Tests must implement this statement in the `checkTestPasses()` function in their interface. Though Ante may most frequently be applied to smart contracts from a single protocol, Ante Tests are also free to make use of data from multiple other smart contracts.

Simple examples of possible invariants for Ante Tests include:

- To test that WBTC has not been compromised, an invariant may check
$$\texttt{WBTCToken.totalSupply()} \leq 21 \cdot 10^6 \cdot 10^8$$
to ensure total supply of WBTC is at most 21 million.
- To test that the `Eth2DepositContract` address does not have ≈99.99% of its current ETH withdrawn (likely by a compromise), an invariant may check
$$\texttt{depositContract.balance} \geq 500 \cdot 10^{18}.$$

- To test that the circulating supply of wETH matches the balance of ETH in its contract, an invariant may check

$$\texttt{WETH9Addr.balance} = \texttt{WETH9Token.totalSupply()}.$$

Examples which might be more relevant for decentralized finance protocols include:

- For overcollateralized lending protocols, an invariant may verify correct overcollateralization at a per-position or protocol level by checking

$$\texttt{value\_of\_debt} \leq \texttt{value\_of\_collateral} \cdot \texttt{safety\_factor}.$$

- For constant function market makers (see [3]) which allow trades which increase the value of a function $f(X, Y) = K$ of the reserve amounts X and Y of two tokens, an invariant may verify the relation relating reserves to the constant K in fact holds. For example, for Uniswap v2 [2], one could test that in `UniswapV2Pair.sol` the following equality holds:

$$\texttt{reserve0} \cdot \texttt{reserve1} \geq \texttt{kLast}.$$

  NB: If the 5bp Uniswap v2 protocol fee is off, `kLast` is fixed at 0, and this test will always pass. It will be a more non-trivial test for pairs with the protocol fee turned on.
- For protocols with treasury controlled by a multi-sig wallet, an invariant may verify

$$\texttt{balance\_of\_multisig} \geq 0.01 \cdot \texttt{initial\_balance}$$

  to ensure that the multi-sig members of a potentially anonymous team have not moved more than 99% of the multi-sig funds. While such a move could in principle be legitimate, it may also be caused by a compromise of the multi-sig or malicious behavior on the part of its members. Such an Ante Test would thus provide partial protection against a so-called "rug pull" by the protocol team.

2.2. **Staking, challenging, and verifying via Ante Pools.** Once an Ante Test has been deployed, developers can integrate it with an Ante Pool, a smart contract which allows users to stake, challenge, or verify the test. We describe each operation in turn.

- **Staking:** Stakers deposit ETH into the Ante Pool to stake an Ante Test and express the view that the test will continue to pass. Stakers earn block-by-block yield by proportionally splitting decay rewards from challengers, but lose their deposit if the Ante Test fails verification.
- **Challenging:** Challengers deposit ETH into the Ante Pool to challenge an Ante Test and express the view that the test will fail at some point. If on-chain verification of the Ante Test fails at any point, the challengers will receive their deposits back as well as split the stakers deposits less a 5% bounty for the verifier. However, challengers' deposit decays by 100 gwei/ETH each block (approximately 20% annualized) to pay for stakers' decay reward.
- **Verifying:** Any challenger who has deposited for at least 12 blocks can verify that the Ante Test passes for the current on-chain state by calling the `checkTestPasses()` function. If the function returns true, i.e. the test passes, nothing happens. If the function returns false, i.e. the test fails, the verifier is paid a bounty of 5% of stakers' deposits, and the rest of the stakers' deposits are rewarded pro rata to all challengers.

Stakers and challengers may withdraw their deposits in an Ante Pool if they no longer wish to stake or challenge. To allow sufficient time for verifiers to respond on-chain to knowledge of an exploit, stakers must wait for 6000 blocks (approximately 1 day) to withdraw funds.

2.3. **Implementation details.** We now describe and explain some technical details affecting the precise implementation of Ante.

- **Waiting times for deposits and withdrawals:** To prevent challengers from depositing funds to frontrun a verification which causes an Ante Test to fail, challengers are only eligible to receive payouts after their deposits have been in an Ante Pool for at least 12 blocks. This interval was chosen as a threshold for approximate finality on Ethereum.

  To prevent stakers from withdrawing funds in response to an exploit which is publicly known off-chain but has not yet caused the relevant Ante Test to fail on chain, stakers must wait for 6000 blocks (approximately 1 day) to withdraw funds. This interval was chosen to allow verifiers sufficient time to verify an Ante Test before stakers withdraw funds.
- **Reversion of Ante Tests:** Reversions of the `checkTestPasses()` function are treated as failures of the corresponding Ante Test. It is therefore the responsibility of stakers to ensure that any test they stake on can be executed correctly.
- **Upgradeable contracts:** Ante Tests are immutable. Thus if a contract referenced by an Ante Test is upgraded in an incompatible way, the `checkTestPasses()` function will revert and the Ante Test will always fail. We recommend that projects using Ante either upgrade in a compatible way or coordinate off-chain for all stakers in the un-upgraded Ante Test to migrate to a new Ante Test.

3. Applications

3.1. **Signaling legitimacy.** New DeFi projects wishing to express confidence in their code may use Ante as a new hard-to-fake signal of legitimacy. For example, if the developers or investors behind a newly released overcollateralized lending protocol are willing to stake a substantial amount of funds on an Ante Test checking that the protocol remains solvent, users may have more confidence that the code and mechanism design enforcing solvency will work in practice. On the other hand, if security experts are dubious about the design and implementation of the protocol, they may challenge the Ante Test, thereby justly reducing user confidence.

Such a decentralized measure of project legitimacy is especially valuable for pseudonymous teams who may lack other means of demonstrating good-faith, but we envision it to complement tools such as audits, bug bounties, and traditional insurance coverage for protocols of all kinds.

3.2. **On-chain trust quantification.** Users can compute a **decentralized trust score** between 0 and 100 for each Ante Test based on its Ante Pool participation using the formula

$$\texttt{decentralized\_trust\_score} = \frac{\texttt{staker\_deposits}}{\texttt{staker\_deposits} + \texttt{challenger\_deposits}} \cdot 100.$$

This score provides a rough indication of the aggregated confidence that stakers and challengers on the Ante Pool have that the Ante Test will continue to pass. Because this score is based on on-chain deposits alone, it becomes increasingly difficult to influence as the amount of funds in an Ante Pool increases, and users may compute and verify the score using on-chain data alone.

4. Possible extensions to the Ante Protocol

We now describe possible improvements or upgrades to the Ante Protocol which we hope to see from the community in future versions.

4.1. **Multiple collateral asset support.** Ante Pools may be extended to accept deposits of multiple collateral types instead of only ETH. This will require using oracles such as Chainlink [5] to compare the value of different assets and may also require a protocol governance procedure to adjudicate which types of collateral are permitted.

4.2. **Ante Metapools.** Ante Pools may be extended to Ante Metapools, which allow stakers to stake the same capital on several Ante Pools simultaneously. Stakers in an Ante Metapool would earn more decay rewards by using the same deposit to express confidence in the security of several Ante Tests, but they would would lose their deposit if any of those tests failed. Protocol governance procedures may be relevant for deciding which Ante Tests to include in such Ante Metapools.

4.3. **Emergency freezes of protocols based on Ante Tests.** Ante Tests can be generalized to include a `checkTestAndFreeze()` function in addition to `checkTestPasses()`. Protocols may whitelist Ante Tests to allow `checkTestAndFreeze()` to freeze all functionality aside from emergency withdrawals on their protocol in the event of an invariant failure. By thus providing a live on-chain analogue of a traditional assert statement, Ante Tests could enable protocols to give users greater confidence that unexpected violations of the claimed invariants will lead to orderly emergency recovery and not a catastrophic loss of funds.

Disclaimer

This paper is for general information purposes only. It does not constitute investment advice or a recommendation or solicitation to buy or sell any investment and should not be used in the evaluation of the merits of making any investment decision. It should not be relied upon for accounting, legal or tax advice or investment recommendations. This paper reflects current opinions of the authors and is not made on behalf of Unitary or its affiliates and does not necessarily reflect the opinions of Unitary Inc, its affiliates or individuals associated with Unitary Inc. The opinions reflected herein are subject to change without being updated.

References

[1] Aave protocol whitepaper v2.0, 2020. `https://github.com/aave/protocol-v2/raw/master/aave-v2-whitepaper.pdf`.

[2] Hayden Adams, Noah Zinsmeister, and Dan Robinson. Uniswap v2 core, 2020. `uniswap.org/whitepaper.pdf`.

[3] Guillermo Angeris and Tarun Chitra. Improved price oracles. *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, Oct 2020.

[4] Michael Egorov. Stableswap: efficient mechanism for stablecoin liquidity, 2019. `https://curve.fi/files/stableswap-paper.pdf`.

[5] Steve Ellis, Air Juels, and Sergey Nazarov. ChainLink, 2017. `https://research.chain.link/whitepaper-v1.pdf`.

[6] Robert Leschner and Geoffrey Hayes. Compound: The money market protocol, 2019. `https://compound.finance/documents/Compound.Whitepaper.pdf`.

[7] Sam M. Werner, Daniel Perez, Lewis Gudgeon, Ariah Klages-Mundt, Dominik Harz, and William J. Knottenbelt. Sok: Decentralized finance (defi), 2021. `https://arxiv.org/pdf/2101.08778.pdf`.