Binance :: Proofs of Liabilities

Security Report by Kostas Kryptos Chalkias (Mysten Labs)
- - cross-checked by Panos Chatzigiannis (Visa Research) and Yan Ji (Cornell)

Binance algorithm (plain Merkle tree approach)

https://www.binance.com/en/proof-of-reserves
https://www.binance.com/en/support/faq/how-to-verify-merkle-tree-819fadc1c16b499d85bc3a3d0fab76bc

7 Findings in total:

- 4 potentially exploitable vulnerabilities (where Binance could understate liabilities)
- 1 privacy issue (Binance reveals number of users)
- 1 trust-model issue (Users should blindly trust the auditor for the sum Vs other better Liability proof solutions)
- 1 important performance optimization (which results in a much cheaper and faster proof verification on the user side, while the current model is impractical as users need to download and compute the whole Merkle tree, consisting of millions of leaves).

Vulnerabilities

1. Truncated Merkle leaves to 8 bytes (16 hex characters)

Impact: Binance could understate liabilities (balances) for ALL of its users.

TL;DR: Binance could (offline) find collisions for two different balances for the same user (and for each user). Then it can report a lower balance to the auditor (i.e., zero), and the correct balance to the user. In theory, Binance could report zero total liabilities balance and nobody will notice. Different salt per user will allow this attack to require 2^32 hashes per user (super easy with today's CPU/GPU/ASICS power). Note that the auditor cannot know or learn if Binance performed an offline collision attack.

Solution: Do not truncate Merkle leaves, let it be 256bits (32 bytes).

2. Potentially reusable AccountIDs

Impact: Binance could send the same leaf to two or more different users that happen to have the same balance(s) and understate (some) liabilities.

TL;DR: Binance could group users with the same balances by temporarily using the same userID between them. Then show one ID to the auditor and send the same leaf to all of the users of each group. Binance can thus report a particular balance only once and share the same proof with all of the users having that same balance, thus hiding duplicated balances from the sum of liabilities.

Solution: Ensure AccountIDs are unique, i.e., compute it by user's email address. I.e., Hash(secret_salt, email) = AccountID (or Account Code as referred to Binance's document)

3. Not pinned Merkle Root hash

Impact: Binance could send different roots to different users, because it only publishes this in their website (not in a public bulletin board). Then it can understate liabilities.

TL;DR: The Merkle root is currently served from Binance's website, but there is no guarantee Binance does not show different Merkle root per IP that enters the website. Thus in theory, Binance could just post fake Merkle root and Merkle trees per user.

Solution: Sign the root by an auditor's signature or better publish the root in the blockchain (which works as a public bulletin board record that cannot be altered).

4. Binance knows who checked the proof

Impact: Binance could track who is downloading Merkle trees and checked proofs. It can then remove users from future Solvency proofs who never check (with high confidence).

TL;DR: Now everything is served by Binance's website, which implies that Binance can fully track which users check the proofs. It can then do analytics and remove balances of users who never check proofs in all future liabilities. The probability that these users will never check is very high and thus Binance can understate future liabilities by predicting which accounts it can omit.

Solution: The service that offers the proof elements and checks proofs should be audited by another external auditor who will test that Binance is not tracking which users download or check proofs. Alternatively, Binance can provide a decryption key to every user and the user can download and check the proofs in another website (i.e. from the auditor).

Privacy issues

1. Binance reveals number of users

Impact: Externals can know exactly how many users Binance has every time it publishes a Solvency proof. Observers can track progress in customer-base sizes, before any public financial announcement from the company and use this information for trading.

TL;DR: Binance uses the plain Mekrle tree approach and actually publishes the whole list of leaves. So one can count the exact number of users Binance has.

Solution: Add padding nodes; however there are better solutions, like using DAPOL+ sparse Merkle trees that require logN padding nodes.

Trust issues

1. Users need to blindly trust the auditor

Impact: Users cannot know if the sum of liabilities is correct, they blindly trust the auditor

TL;DR: Binance uses the plain Mekrle tree approach which only hashes to the next layer. The only entity that checks all balances and does the sum is the auditor. We have examples in traditional companies (Enron) where there was collusion or (accidental) malfunction of the auditor checks. Ideally users should ensure that they have the power to check liabilities even if the auditor is rogue

Solution: Use DAPOL+ which applies a summation tree with Bulletproofs (proposed by FB research and this year candidate as best security research paper in the world, finished in top10 in CSAW 2022 competition). Generally DAPOL+ is compatible with full auditor visibility but also protects users against privacy and trust issues and is considered the state of the art and the only protocol with security proofs regarding security, transparency and correctness. Published in CCS 2021 (one of the top cryptography and security conferences). Alternatively, a generic SNARK circuit to prove correct summation and correct accumulator computation (tree or any other commitment scheme) would also work.

Performance issues

1. Users need to download the whole list of Merkle tree leaves

Impact: The list is big and might disincentivize users with low internet data plans to download and check.

TL;DR: At the moment, each user has to download a huge list of Merkle leaves which is not practical, especially on mobile internet. It might also take 20 mins in low-end devices, just to generate the tree locally, which is time-consuming and might disincentivize verifiers.

Solution: Users do not need to download everything, just an inclusion proof as described in DAPOL+

Research Papers to consider:

- **Broken Implementations for Proof of Reserves in major crypto exchanges** - CoDecFin FC 2022 (*as you will notice NOBODY is doing it right*: all Deloitte, Kraken, Armanino, the old Coinfloor and BHEX etc audits have exploitable bugs or processes) https://eprint.iacr.org/2022/043

- **GPOL scheme for Proof of Liabilities (this is DAPOL+)** - CCS 2021 (the most recent protocol for proving liabilities - considered for standardization + top10 finalist in CSAW 2022, one of the most prestigious applied research competitions re most innovative / impactful papers of the year) *Tech: sparse Merkle trees + Bulletproofs range proofs + random sampling* https://eprint.iacr.org/2021/1350

- **SoK for Crypto Audits** - ACNS 2021 (the most comprehensive survey on what algorithms exist - probably read this first as you'll find concrete asset proofs solutions even for zCash, Monero etc) https://eprint.iacr.org/2021/239

- **Distributed Audit Proofs of Liabilities** - ZKPROOF 2020 (FB's Libra effort to standardize proofs of liabilities) *Tech: Bulletproofs range proofs + Merkle trees* https://eprint.iacr.org/2020/468.pdf

- **Provisions** - CCS 2015 (a MUST read: the 1st privacy preserving Solvency solution - DAPOL+ is an evolved version of Provision) *Tech: concrete ZKP solution + proofs of no collusion* for proofs of assets https://eprint.iacr.org/2015/1008

- **Having a safe CEX: proof of solvency and beyond** - Vitalik's recent blogpost https://vitalik.ca/general/2022/11/19/proof_of_solvency.html