Contest Proposal: Groth16 zkSNARK Proof Verification Use Cases

Submission period: May 22, 2021 00:01 UTC - June 30, 2021 at 23:59 UTC

Voting period: 20 days

Background and Description

=nil; Foundation as an initial member of Free TON community developed an upgraded version of TON Virtual Machine, which includes cryptographic primitives required for usage zero-knowledge proof verification within the virtualized applications. =nil; Foundation also prepared C++ (https://github.com/nilfoundation/cpp-ton) and Rust-y (https://github.com/nilfoundation/rust-ton) ZK proof verification instruction-enhanced TON protocol implementations.

A test protocol instance was launched using the C++ ZK proof verification instruction-enhanced implementation. Network configuration used for the contest is available at: https://github.com/NilFoundation/ton-proof-verification-contest/blob/master/config/testnet.config.json**.**

ZKP test network visualization is available at https://live.freeton.nil.foundation and at https://nil.ton.live.

Before the Free TON community will be able to patch a mainnet node-clients this ZKP clients should be tested for security and stability.

This document proposes the first in a series of “ZKP contests” aiming motivation of Free TON developer community to try prepared tools and to crowdsource simple ZKP use cases for testing purposes.

Instructions for participants

Participants are expected to create any trivial sample case which uses Groth16 proofs.

Contest repository (aka place to start) is available at: https://github.com/nilfoundation/ton-proof-verification-contest

Advanced proof generation and circuit definition documentation is available at: https://crypto3.nil.foundation/projects/crypto3/d8/da7/blueprint_usage_manual.html.

General requirements

Solutions provided are expected:

- To be a correctly functioning FreeTON LSCS deployed on a test network (https://live.freeton.nil.foundation)
- Not to be a TONCash-alike or any anonymous transactions/token proposal. There is a separate contest for that.

- To involve VERGRTH16 TVM instruction usage.
- To contain circuit definitions done (preferably) with =nil; Crypto3 Blueprint library (https://github.com/NilFoundation/crypto3-blueprint) or as a formal statement.
- To contain proving/verifying key and the statement being proved (primary and auxiliary inputs).

Evaluation criteria and winning conditions

- Apart from uploading a submission, a code should be submitted in accordance with https://github.com/freeton-org/readme.
- A participant should do a presentation of her solution at a convenient time agreed with DevEx members. A solution should include tests with clear instructions.
- If a test does not cover some scenarios, then jurors can develop their own tests, but it should reduce such a submission score.
- The solution should have an open source license.
- The solution has to comply with formal requirements introduced by the instructions for jurors.
- Each submission should be rated by jurors based on its:
    - Innovativeness
    - Complexity
    - Suitability for real use

Instructions for jurors

Jurors are supposed to verify the correctness of the proof submitted by the participant to the test cluster (in case it is not possible to trust the usual verification because that is exactly what is being tested). To achieve that it is required:

1. To reproduce the proof generation the same way as the participant did it. Participants are expected to provide a proving/verifying keypair and circuit definition for that.

    Circuit definitions are accepted in:

    1. =nil; Crypto3 Blueprint library format. Usually this is a set of C++ sources defined as the manual states: https://crypto3.nil.foundation/projects/crypto3/d8/da7/blueprint_usage_manual.html. Jurors are expected to simply compile sources provided by participants. This is an option preferred.
    2. Formal statement. This will require for the juror to implement the circuit definition himself with some tool available for that (=nil; Crypto3 Blueprint library, Bellman library, libsnark, ZooKrates DSL, etc). This is not an option preferred.
2. Generate the proof using participant's proving key and make sure it is the same as the participant submitted to the test cluster. The process is the same as participants are supposed to perform: https://github.com/NilFoundation/ton-proof-verification-contest
3. Verify the proof out of the TVM using participant's verification key and some native verification mechanism. The easiest way to do that is to use =nil; Crypto3 ZK library

(https://github.com/NilFoundation/crypto3-zk) according to the instructions available in here: https://github.com/NilFoundation/ton-proof-verification-contest

After each of these steps is completed successfully, the juror can consider the particular participant's proof generation and verification formally correct.

Voting

- Jurors whose team(s) intend to participate in this contest by providing submissions lose their right to vote in this contest.
- A jury from other sub-governance groups could be added to this contest to provide additional technical expertise.
- Each juror will vote by rating each submission on a scale of 1 to 10.
- Jurors should provide feedback on each submission.
- The jury will reject duplicate, subpar, incomplete, or inappropriate submissions.

Reward

Only submissions with an average score equal to or more than 4.0 can get a reward.

1st prize ... 80,000 TONs
2nd prize ... 60,000 TONs
3rd prize ... 40,000 TONs
4th place … 35,000 TONs
5th place … 30,000 TONs
6th place … 25,000 TONs
7th place … 20,000 TONs
8th place … 15,000 TONs
9th place … 10,000 TONs
10th place … 5,000 TONs

Note: If the number of winning submissions is less than the number of rewards available, any remaining rewards are not subject to distribution and are considered void.

Jury rewards

An amount equal to 15% of all total tokens actually awarded will be distributed equally between all jurors who vote and provide feedback. Both voting and feedback are mandatory in order to collect the reward.

Governance rewards

An amount equal to 2 % of the prize fund will be allocated to members who participated in organizing the contest, to be distributed equally among them:

- @nemothenoone
- @prigolovko
- @Futurizt

- @anovi

Procedural remarks

- Participants must upload their work correctly so it can be viewed and accessible in the formats described. If work is inaccessible or does not fit the criteria described, the submission may be rejected by jurors.
- Participants must submit their work before the closing of the filing of applications. If not submitted on time, the submission will not count.
- Participants are free to ask/resolve any questions and propose any suggestions at the AMA event planned to be held no later than May 20, 2021. Exact timing will be agreed in DevEx Telegram chat-group.
- Initial contest manual and documentation version is supposed to be done no later than May 10, 2021. It is supposed to enhance, fix and enrich those manuals and documentation during the contest according to participants' needs and confusions.