Synthetic PoRep security audit report

Authors	Kubuxu Max Luca Irene
Created by	Max
Created time	@May 18, 2023 5:55 PM
Date	@May 24, 2023
Project	
Stage	

🧑‍🚀 This report covers the internal security audit performed on Synthetic PoRep

Audit	Kuba, Luca, Irene
Advisors	Jake, Nemo
Report	Max
Date	@May 8, 2023 to @May 31, 2023
Codebase	https://github.com/filecoin-project/rust-fil-proofs/pull/1701
FIP	#59
Spec	https://hackmd.io/@jake/synth-porep-spec

📔 Introduction

Synthetic PoRep achieves reduction in used up space by reducing the set of challenges that might be chosen during the interactive Commit step from all possible challenges to some predetermined number that is feasible to precompute.

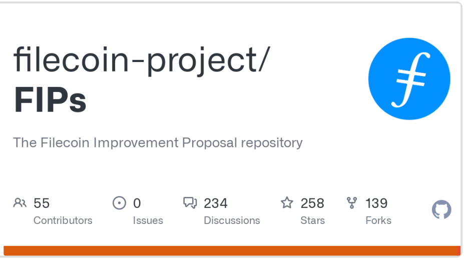
- A Storage Provider can complete the challenge generation and vanilla proof computation before performing PreCommit on-chain thus removing layers data before the sector is pre-committed on-chain;
- The GPU cost for SNARK generation during Commit is not significantly increased.

For more info see

FIPs/fip-0059.md at master · filecoin-project/FIPs

The Filecoin Improvement Proposal repository. Contribute to filecoin-project/FIPs development by creating an account on GitHub.

https://github.com/filecoin-project/FIPs/blob/master/FIPS/fip-0059.md



filecoin-project/
FIPs

The Filecoin Improvement Proposal repository

55 Contributors · 0 Issues · 234 Discussions · 258 Stars · 139 Forks

🧑‍💻 Internal security audit

👊 Main findings

Challenge generation attack

🧑‍🚀 Type: security
Status: resolved ✅
Severity: high

As designed and implemented, the synthetic challenge generation used only ReplicaID for generating challenges.
The synthetic challenge generation is a partial Fiat-Shamir transform of the interactive Replication protocol. The Fiat-Shamir transform requires that all inputs of the protocol, up until the point of the usage, have to be hashed. As presented in the *Fully Interactive* diagram, these inputs are ReplicaID, CommD (which is included in ReplicaID) and CommR.

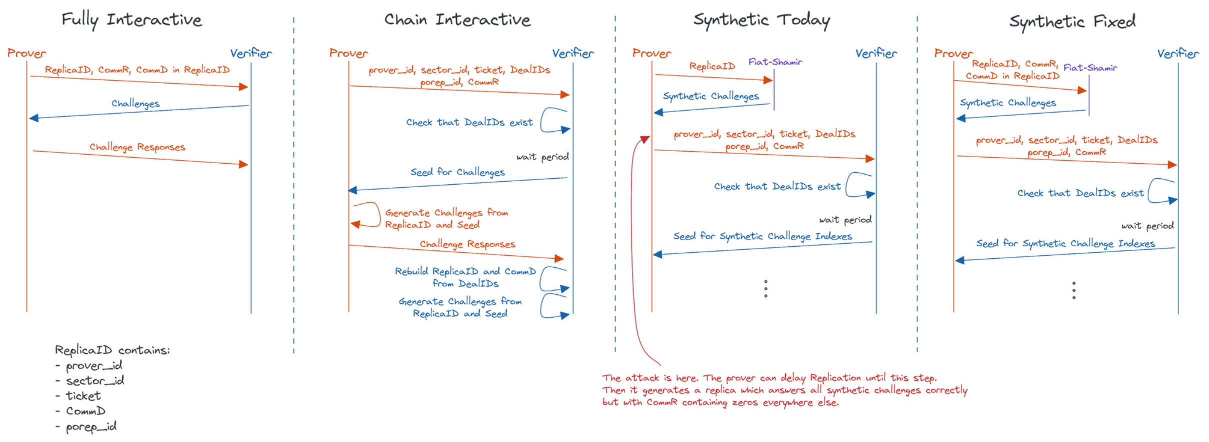
The CommR input was committed during the initial design of Synthetic PoRep,

Without CommR, we found an attack. A malicious SP uses the replica_id to get the synthetic challenge set before sending CommR on chain. Given the set, the SP crafts a replica SDR labels and R' where the selected nodes (ie, the nodes in the `N_synth` vanilla proofs) are valid, but every where else there are zeros. SP computes commR' and submits it to chain. Now the PoRep proofs (and also every following windowPoSt proof) will be accepted for R' (which is basically almost zeros everywhere) as it would for a valid (non-zeros) replica R.

Solution

Add CommR as input of `gen_synth_challenges` and change the chacha key to key = `Hash(Replica_id || CommR)`.

This is the correct solution because if the SP tries to change the SDR labels to put zeros on the nodes that not challenges, the crafted R' will not match with commR and the porep proof will fail.


Fully Interactive
Prover
Verifier
ReplicaID, CommR, CommD in ReplicaID
Challenges
Challenge Responses
ReplicaID contains:
- prover_id
- sector_id
- ticket
- CommD
- porep_id
Chain Interactive
Prover
Verifier
prover_id, sector_id, ticket, DealIDs porep_id, CommR
Check that DealIDs exist
wait period
Seed for Challenges
Generate Challenges from ReplicaID and Seed
Challenge Responses
Rebuild ReplicaID and CommD from DealIDs
Generate Challenges from ReplicaID and Seed
Synthetic Today
Prover
Verifier
ReplicaID
Fiat-Shamir
Synthetic Challenges
prover_id, sector_id, ticket, DealIDs porep_id, CommR
Check that DealIDs exist
wait period
Seed for Synthetic Challenge Indexes
The attack is here. The prover can delay Replication until this step. Then it generates a replica which answers all synthetic challenges correctly but with CommR containing zeros everywhere else.
Synthetic Fixed
Prover
Verifier
ReplicaID, CommR, CommD in ReplicaID
Fiat-Shamir
Synthetic Challenges
prover_id, sector_id, ticket, DealIDs porep_id, CommR
Check that DealIDs exist
wait period
Seed for Synthetic Challenge Indexes

Number of challenges

🧑‍🚀 Type: security
Status: resolved ✅
Severity: medium

The FIP sets the number of synthetic challenges at 2^18, but the code used the previous parameter value equal to 2^14.

Solution

Changed the codebase to 2^18 challenges.

ChaCha20 vs ChaCha12

👨‍🚀 Type: performance
Status: resolved ✅
Severity: low

What is the performance of the verifier's challenge generation using ChaCha20? If the CPU time spent there is even on the radar there, we should consider switching to ChaCha12 which is a safe choice given the progress of ChaCha's cryptanalysis.

- Jake's benchmarks (May-15) for 32GIB sector size, 2^18 synth challenges, 180 PoRep challenges, and ChaCha20 (using the DSTs specified above and inclusion of CommR in synth challenge generation ChaCha20 key):
    - Synth challenge generation: 0.07 sec
    - PoRep challenge selection & generation: 0.000079 sec = 790k gas
    - Regular PoRep challenge generation (180 challenges): 0.00006 sec
    - ChaCha12:
        - Synth challenge generation: 0.0466 sec
        - PoRep challenge selection & generation: 0.000077 sec

Solution

The difference between ChaCha20 and ChaCha12 is small enough in the on-chain phase not to matter. A much more significant difference is observable in the generation of all challenges. Still, that improvement does not translate to on-chain challenge generation, probably due to the effects of branch prediction and caches. For reference, this is a paper justifying this approach.

✅ Check list

- [x] ~~Synthetic challenge generation (`gen_synth_challenges`)~~
    - [x] ~~no grindability issues (check inputs of hash, `gen_porep_challenges`)~~

        - [x] ~~Check that all the properties which generate a sector encoding are included in the Fiat-Shamir of the synthetic challenge generation.~~
    - [x] ~~hash output parsing~~
- [x] ~~PoRep challenge sampling from synthetic set~~
- [x] ~~Make sure we don't add anything grindable into the interactive challenge selection~~

Coverage

- We primarily reviewed deltas/changes, including changes to the verification and internal proving code paths.

What wasn't audited

- We didn't review changes to the rust-fil-proofs external APIs required for adoption of these changes.

Level of confidence

- High

🛣️ Next steps

Why not an external audit?

We decided not to run an external audit for the following reasons

- It is a very scoped change which treats the rest of the system as a black box. On the other hand it requires detailed understanding why the rest of the system can be treated as a black box.
- It is orders of magnitude less complex than other changes that were externally audited.
- Time and resource contraints. There are other areas of the filecoin codebase where an external audit would have a bigger impact.
- Sometimes external audits missed some issues (for example in SnapDeals).