Lesson 14

Update on rollups and EIP 4844

See[Update] (https://twitter.com/epolynya/status/1504389895925600261)
Costs currently
ORs > ZKRs > validiums
Post EIP 4844,
Validiums > ZKRs > ORs
He suggests that off chain data makes less sense after EIP 4844

EIP 4844 - SHARD BLOB TRANSACTIONS

Today, rollups use calldata. In the future, rollups will have no choice but to use sharded data (also called "blobs") because sharded data will be much cheaper. Hence, rollups cannot avoid making a large upgrade to how they process data at least once along the way. But what we can do is ensure that rollups need to only upgrade once.
This immediately implies that there are exactly two possibilities for a stopgap:
(i) reducing the gas costs of existing calldata, and
(ii) bringing forward the format that will be used for sharded data, but not yet actually sharding it. Previous EIPs were all a solution of category (i); this EIP is a solution of category (ii).

(Answering homework 8)

MEV on L2s (Starknet)

From article

StarkNet's approach to L1<>L2 interoperability is through messages. There's two kind of messages:

A: L1 → L2 MESSAGES

1. L1 contract calls `send_message()` on the StarkNet L1 contract, passing target and calldata
2. L2 sequencer consumes the message and invokes the function on the target L2 contract (implemented with the `l1_handler` modifier)

B: L2 → L1 MESSAGES

1. L2 contract calls system function `send_message_to_l1()`, registering the message on the sequencer's state
2. L2 sequencer settles state on L1, storing the message on the StarkNet L1 contract waiting to be consumed
3. The L1 target contract can now call `consumeMessageFromL2()` of the StarkNet L1 contract

- L2 sequencers can mev any regular L2 and `l1_handler` transaction

- Nothing currently forces L2 sequencers to invoke `l1_handler` functions let alone in a particular order. This makes `A2` a focal point for mevving
- Conversely, this renders `A1` a no-MEV zone since txs don't get orderer until the L2 sequencer picks them up on the other side
- No "immediate" MEV can be extracted from `B2` (L2 → L1 state settlement) since it has no immediate side effects on any layer
- Since the L2 → L1 lifecycle is closed by an async L1 transaction, it would make sense to have some sort of keeper/worker/scheduler network automatically executing them for a profit, creating a mevvable market

In summary

- `A1`: can be mevved by the L1 block proposer
- `A2`: can be mevved_by the L2 sequencer
- `B1`: can be mevved by the L2 sequencer
- `B2`: can be mevved by the L1 block proposer
- `B3`: can be mevved by the L1 block proposer

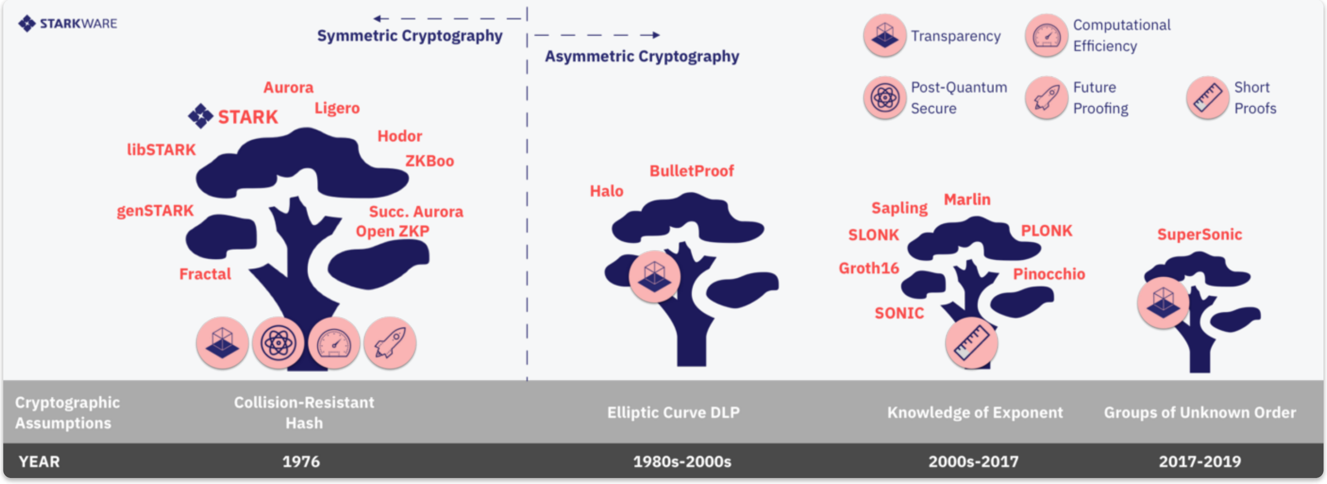
Multiple execution layers make MEV extraction a more complex landscape. Considering how complex single-layer or cross-chain existing MEV strategies are, adding multiple ordering points and asynchrony opens the door for a combinatorial amount of new strategies, markets, and actors.

Some work on frontrunning mitigation strategies have been presented already like this proof of concept of a commit-reveal + timelock system by Yael Doweck and team from StarkWare.

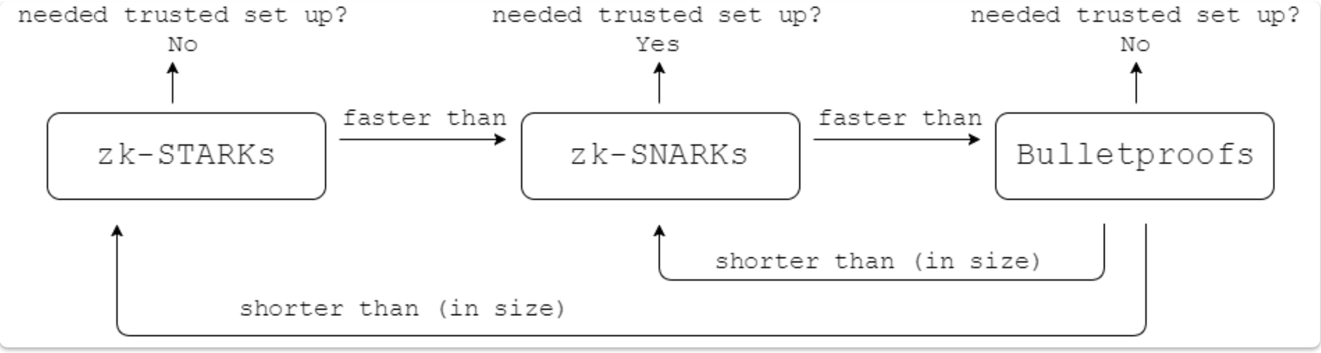
Evolution of ZKPs - Cambrian Explosion

Cambrian Explosion : https://medium.com/starkware/the-cambrian-explosion-of-crypto-proofs-7ac080ac9aed

Overview of proving systems



STARKS / SNARKS / BULLETPROOFS



See my overview video at Ethereum community conference

Trusted Setups

Recent Article from Vitalik

POWERS OF TAU

A powers-of-tau setup is made up of two series of elliptic curve points that look as follows:

$[G1, G1 * s, G1 * s2 ... G1 * s^{n_1 - 1}]$

$[G2, G2 * s, G2 * s2 ... G2 * s^{n_2 - 1}]$

G1 and G2 are the standardized generator points of the two elliptic curve groups; in BLS12-381, G1 points are (in compressed form) 48 bytes long and G2 points are 96 bytes long.
n1 and n2 are the lengths of the G1 and G2 sides of the setup.
Some protocols require n2=2, others require n1 and n2 to both be large, and some are in the middle (eg. Ethereum's data availability sampling in its current form requires n1=4096 and n2=16). s is the secret that is used to generate the points, and needs to be forgotten.

To make a KZG commitment to a polynomial $P(x) = \sum_i c_i x^i$, we simply take a linear combination $\sum_i c_i S^i$ where $S^i = G1 * s^i$ (the elliptic curve points in the trusted setup). The G2 points in the setup are used to verify evaluations of polynomials that we make commitments to.

KZG Commitment Schemes

THE INITIAL ZCASH SETUP

article

Libraries for zkSNARK circuit construction

Libraries, see

- libsnark (C++)
    - Tutorial
    - Tutorial
- bellman (rust)

    - demo circuit
- jsnark (Java, bindings to libsnark)
- Circom - Rust
  Docs
- snarky (Ocaml, from authors of Mina)
- zokrates (toolbox for zkSNARKs on Ethereum)
    - ZoKrates Remix plugin tutorial
    - Zero Knowledge Proof Application Demo, with libsnarks, truffle and docker
- ethsnarks by HarryR (alternative toolkit for viable zk-SNARKS on Ethereum, Web, Mobile and Desktop)
- gnark - library for zero-knowledge proof protocols written in Go
- circom and snarkjs tutorial
    - Roll-up tutorial using Circom and SnarkJS by Ying Tong
- blst
  - written in C and assembly , bindings for Rust / Go
  -ZEXE Snark Gadgets - Rust
- Dalek - for Bulletproofs - Rust

ZKP use cases

Private / Confidential cryptocurrencies

ZCASH, MONERO - SEE PREVIOUS NOTES

[MIMBLEWIMBLE / GRIN] (HTTPS://GRIN.MW/)

Docs
- Similar process to ZCash
- Every transaction has to prove two basic things:
- Zero sum - The sum of outputs minus inputs should always equal zero, proving that a transaction did not create new coins, without revealing the actual amounts.
- Possession of private keys - ownership of outputs is guaranteed by the possession of ECC private keys. However, the proof that an entity owns those private keys is not achieved by directly signing the transaction, as with most other cryptocurrencies.

BEAM

Comparison between Beam / ZCash / Monero
Grin and BEAM are two open-source projects that are implementing the Mimblewimble blockchain scheme. Both projects are building from scratch. Grin uses Rust while BEAM uses C++
See comparison

Blockchain Scalability

See lesson 9 notes

Distributed Proof Systems

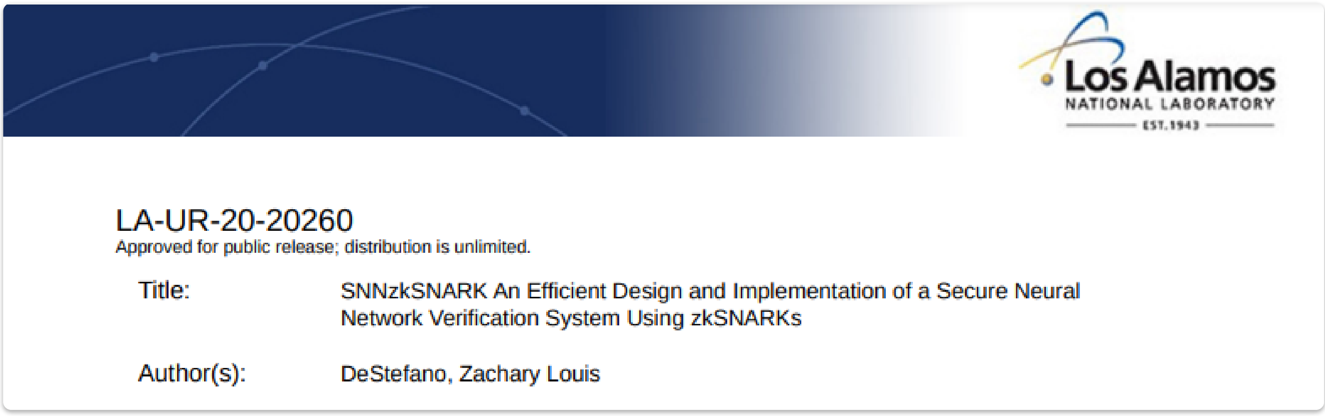
DIZK : https://eprint.iacr.org/2018/691.pdf

Proof of computation with ML

Discussion from Ethereum team
Outsourcing machine learning
Federated ML Overview

Private training for ML
- Using homomorphic encryption of numpy

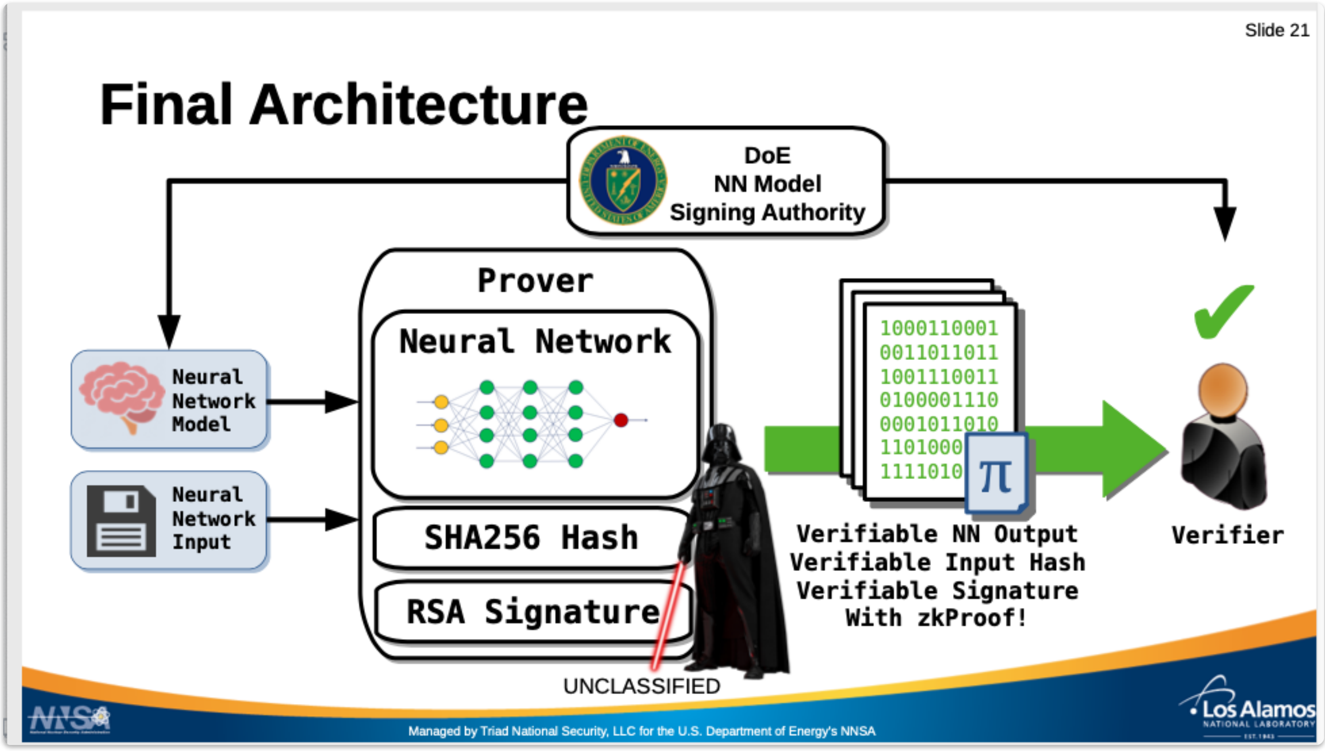
Nuclear Treaty Verification



LA-UR-20-20260
Approved for public release; distribution is unlimited.

Title: SNNzkSNARK An Efficient Design and Implementation of a Secure Neural Network Verification System Using zkSNARKs

Author(s): DeStefano, Zachary Louis



Slide 21

Final Architecture

DoE
NN Model
Signing Authority

Neural Network Model

Neural Network Input

Prover

Neural Network

SHA256 Hash

RSA Signature

1000110001
0011011011
1001110011
0100001110
0001011010
1101000
1111010

π

Verifiable NN Output
Verifiable Input Hash
Verifiable Signature
With zkProof!

Verifier

UNCLASSIFIED

Managed by Triad National Security, LLC for the U.S. Department of Energy's NNSA

Filecoin

PROOF OF REPLICATION

In a Proof of Replication, a storage miner proves that they are storing a physically unique copy, or *replica*, of the data. Proof of Replication happens just once, at the time the data is first stored by the miner.

Whereas Proof of Replication is run once to prove that a miner stored a physically unique copy of the data at the time the sector was sealed, Proof of Spacetime (PoSt) is run repeatedly to prove that they are continuing to dedicate storage space to that same data over time.

Both the Proof of Replication and Proof of Spacetime processes in Filecoin use zk-SNARKs for compression.

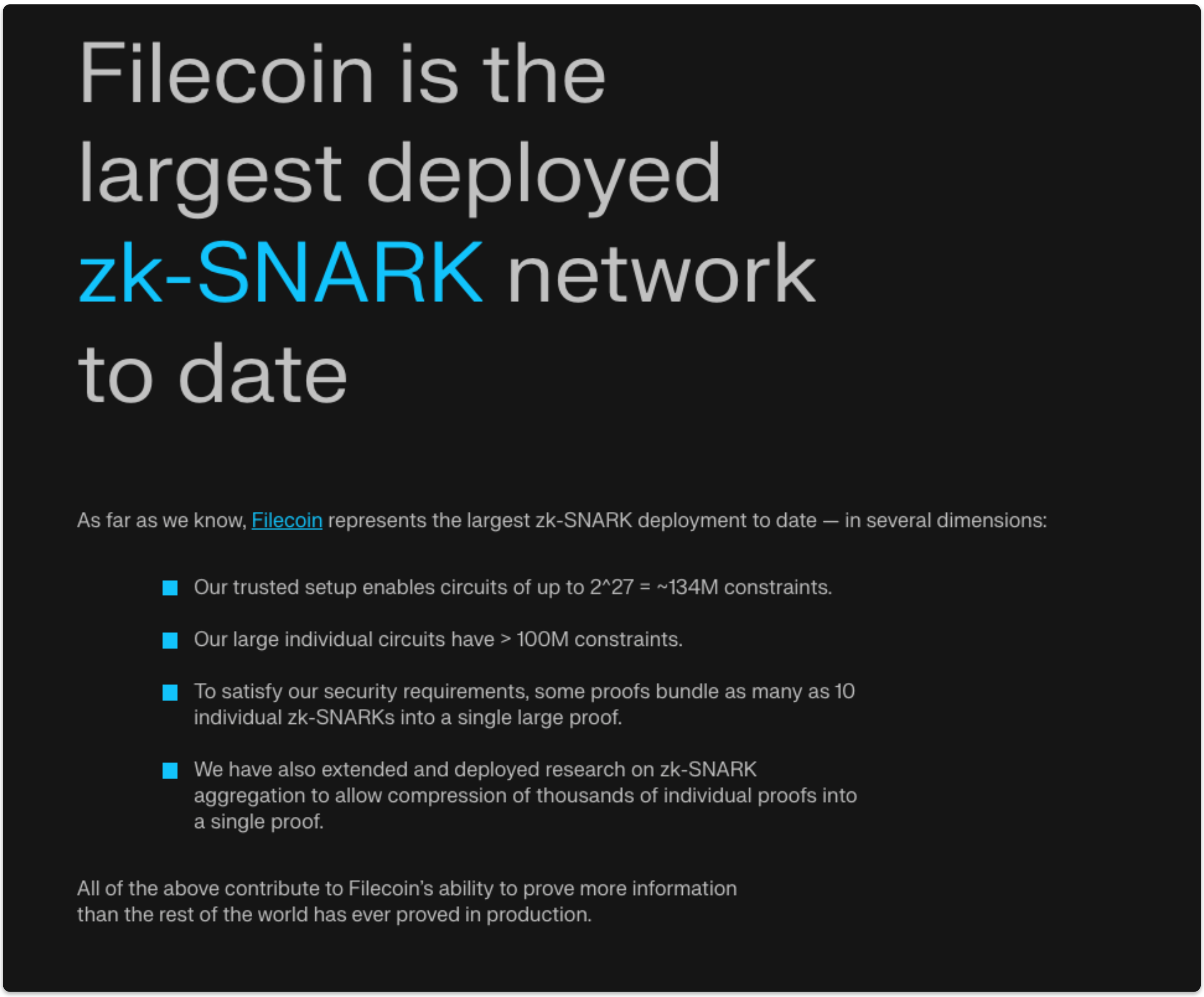
The process of creating Filecoin's zk-SNARKs is computationally expensive (slow), but the resulting end product is small and the verification process is very fast. Compared to the original proofs, zk-SNARKs are tiny, making them efficient to store in a blockchain. For example, a proof that would have taken up hundreds of kilobytes on the Filecoin chain can be compressed to just 192 bytes using a zk-SNARK.

See zkSNARKS for the world

For storage to be verified on Filecoin, two proofs are involved: *Proof of Replication (PoRep)* and *Proof of Spacetime (PoSt).* In PoRep, a storage provider proves that they are storing a unique copy of a piece of data or information. PoRep happens just once, when the initial storage deal between client and provider happens and the data is first stored by the miner. Each PoRep that goes on-chain includes 10 individual SNARKs, which together prove that the process was done correctly through probabilistic challenges.

PoSt, on the other hand, serves to prove that the storage provider *continues* to store the original data over time without manipulation or corruption. When a storage provider first agrees to store data for a client, they must put down collateral in the form of FIL. If at any point during the agreement, the provider fails to prove PoSt, they are penalized and can lose all or some of their posted FIL collateral.

The result of an on-chain interaction in which the *prover* and *verifier* agree that data has been stored and maintained in an appropriate manner is a *proof.* As mentioned above, without a solution to make these proofs small and efficient, they would take up a tremendous amount of the network's bandwidth and deliver high operational costs to both storage providers and miners. By using zk-SNARKs to generate the proofs, however, the resulting proofs are small and the verification process is extremely fast (and thus, cheap). For example, proofs that typically would require hundreds of kilobytes to verify can instead be compressed to just 192 bytes with zk-SNARKs. As mentioned above, each PoRep includes 10 SNARKs, meaning 1920 bytes in each (10*192 bytes).



Filecoin is the largest deployed zk-SNARK network to date

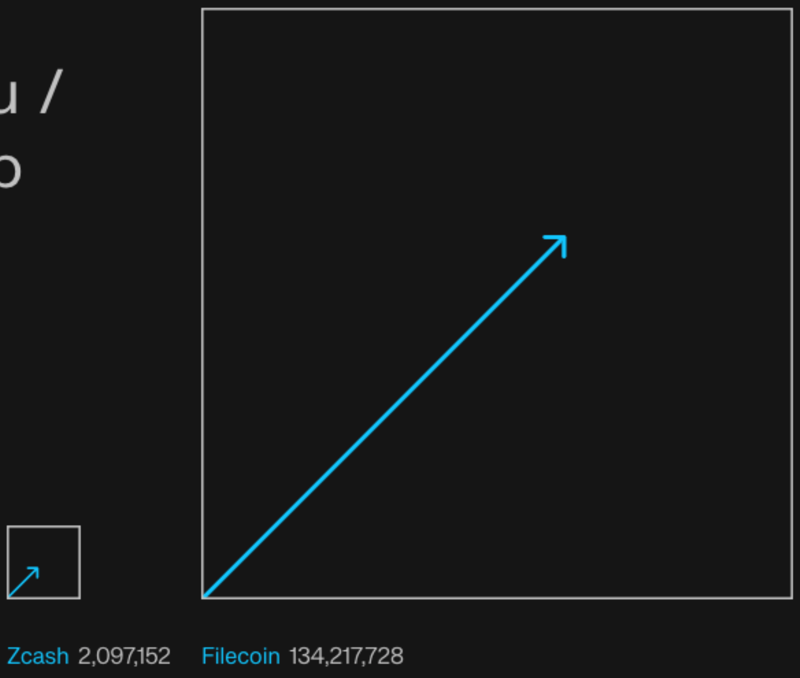
As far as we know, Filecoin represents the largest zk-SNARK deployment to date — in several dimensions:

- Our trusted setup enables circuits of up to 2^27 = ~134M constraints.
- Our large individual circuits have > 100M constraints.
- To satisfy our security requirements, some proofs bundle as many as 10 individual zk-SNARKs into a single large proof.
- We have also extended and deployed research on zk-SNARK aggregation to allow compression of thousands of individual proofs into a single proof.

All of the above contribute to Filecoin's ability to prove more information than the rest of the world has ever proved in production.

The baseline

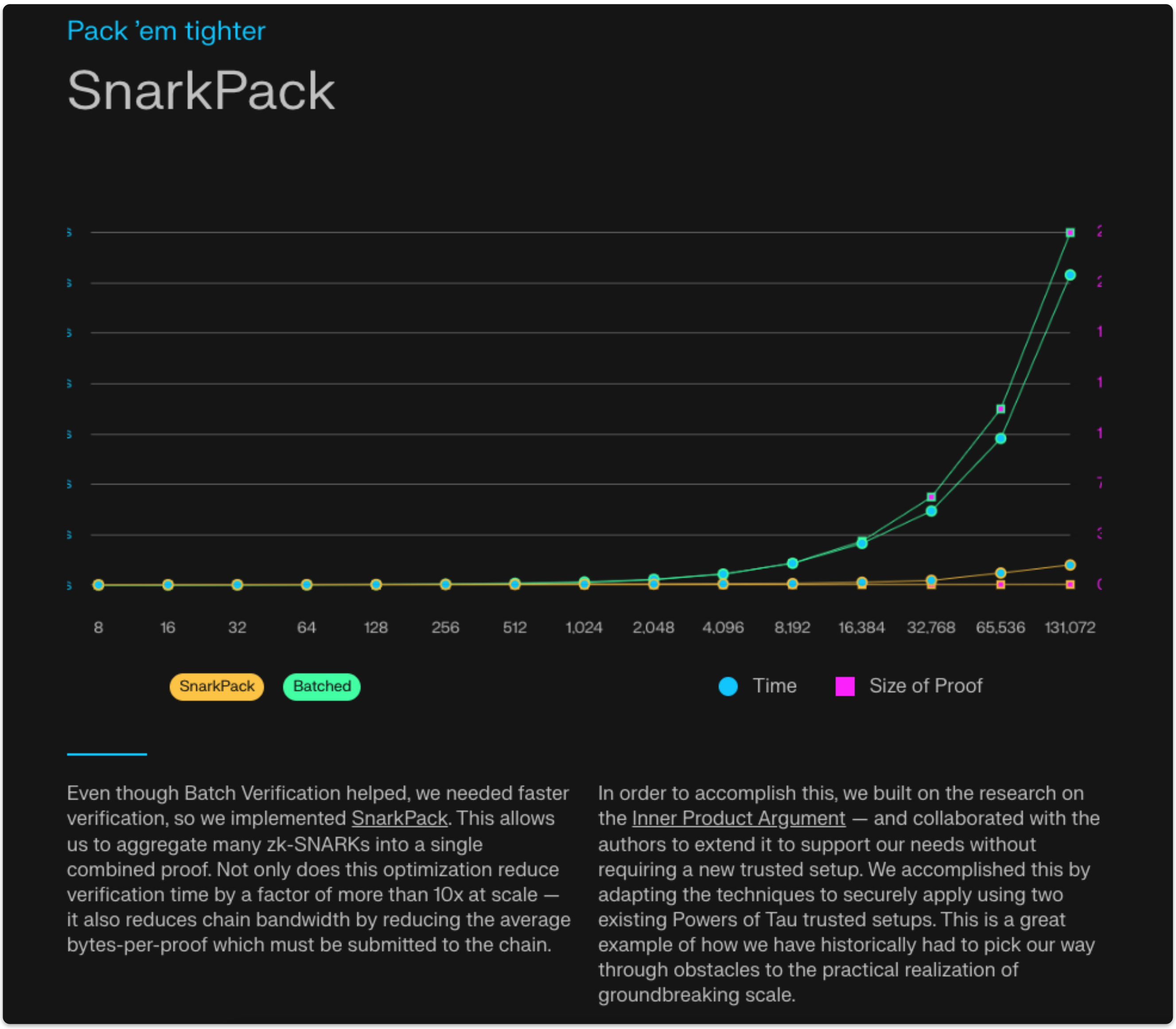
Powers of Tau / Trusted Setup



Maximum Constraints

To support the amount of constraints needed for Filecoin, we ran a new Powers of Tau Ceremony, increasing the supported number by a factor of 64, over the Ceremony Zcash had run. This allows us to generate proofs of over 100 million constraints, limited only by the size of the parameters which must be distributed.

To support the Phase 2 (circuit-specific) trusted setup for our large circuits, we implemented techniques to dramatically decrease RAM usage, allow for parallelism, and reduce I/O overhead — in order to allow many parties using practical hardware to participate during the 7 weeks the ceremony took place.



Pack 'em tighter

SnarkPack

Even though Batch Verification helped, we needed faster verification, so we implemented SnarkPack. This allows us to aggregate many zk-SNARKs into a single combined proof. Not only does this optimization reduce verification time by a factor of more than 10x at scale — it also reduces chain bandwidth by reducing the average bytes-per-proof which must be submitted to the chain.

In order to accomplish this, we built on the research on the Inner Product Argument — and collaborated with the authors to extend it to support our needs without requiring a new trusted setup. We accomplished this by adapting the techniques to securely apply using two existing Powers of Tau trusted setups. This is a great example of how we have historically had to pick our way through obstacles to the practical realization of groundbreaking scale.

Tornado Cash

To process a deposit, Tornado.Cash generates a random area of bytes, computes it through the Pederson Hash (as it is friendlier with zk-SNARK), then send the token & the hash to the smart contract. The contract will then insert it into the Merkle tree.

To process a withdrawal, the same area of bytes is split into two separate parts: the **secret** on one side & the **nullifier** on the other side.
The nullifier is hashed.
This nullifier is a public input that is sent on-chain to get checked with the smart contrat & the Merkle tree data. It avoids double spending for instance.

Thanks to zk-SNARK, it is possible to prove the hash of the initial commitment and of the nullifier without revealing any information.
Even if the nullifier is public, privacy is sustained as there is no way to link the hashed nullifier to the initial commitment. Besides, even if the information that the transaction is present in the Merkle root, the information about the exact Merkle path, thus the location of the transaction, is still kept private.

Deposits are simple on a technological point of view, but expensive in terms of gas as they need to compute the hash & update the Merkle tree. At the opposite end, the withdrawal process is complex, but cheaper as gas is only needed for the nullifier hash and the zero-knowledge proof.

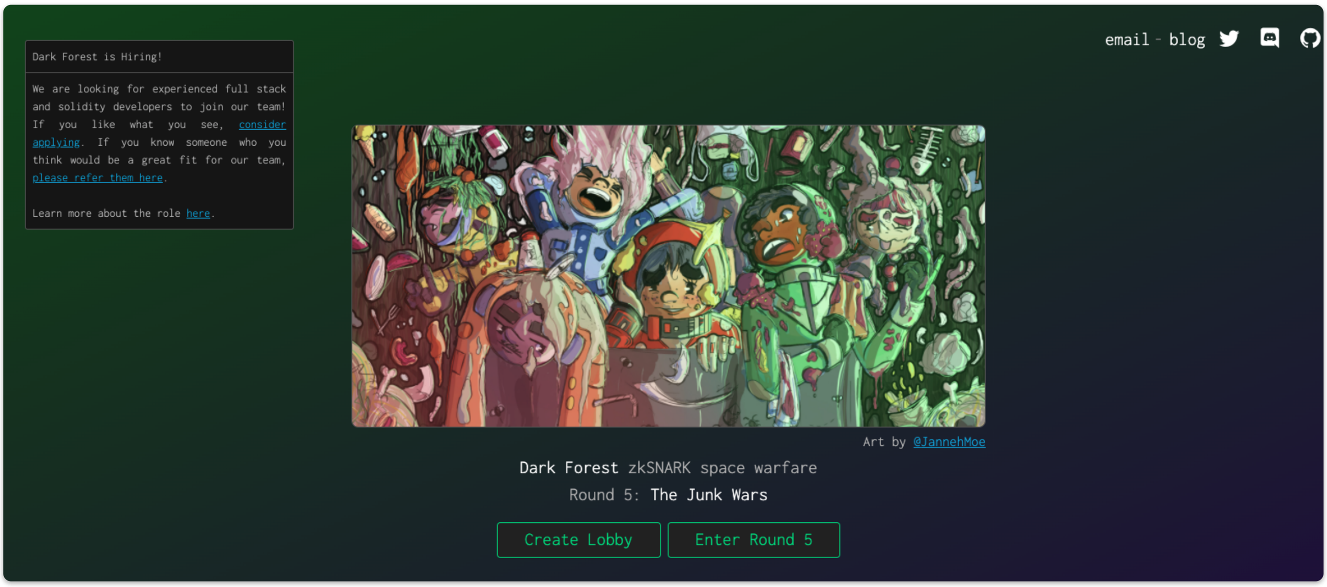
TORNADO CASH VERIFIER CONTRACT

Contract

Zero Knowledge Lottery based on Tornado Cash

See [article] (https://killari.medium.com/zero-knowledge-lottery-437e456dc3f2)

Dark Forest



Dark Forest is Hiring!

We are looking for experienced full stack and solidity developers to join our team! If you like what you see, consider applying. If you know someone who you think would be a great fit for our team, please refer them here.

Learn more about the role here.

Art by @JannehMoe

Dark Forest zkSNARK space warfare

Round 5: The Junk Wars

How does Dark Forest use SNARKs?

A central mechanic in Dark Forest is that the cryptographic “fog of war.” The fog of war ensures that you don’t automatically know where all players, planets, and other points of interests are in the universe; you have to spend computational resources to discover them. This mechanic is secured by zkSNARKs.

In a universe with a fog of war, the locations of all players are private and hidden from each other. This means that players don’t upload the coordinates of their planets to the Ethereum blockchain, which can be publicly inspected. Instead, each player uploads the hash of their location to the blockchain. This ensures that players stay “committed” to a specific location, but also that the location can’t be determined from inspection of the Ethereum data layer.

Without zkSNARKs, there’s an obvious attack vector - if a player uploads a random string of bytes that doesn’t correspond to a real and valid location, and the integrity of the game is broken. To prevent this, Dark Forest requires players to submit zkSNARKs with every move to ensure that players are indeed submitting hashes corresponding to valid coordinates that they have knowledge of.

When players make moves, they’re also required to submit ZK proofs that their moves are “valid” - you can’t move too far or too fast. Without zkSNARKs, a malicious player could make illegal “teleport” moves by claiming that the hash they are moving from is next to the hash they’re moving to, even if the two locations are actually on opposite sides of the universe. Once again, requiring ZK proofs keeps players honest. To use a chess analogy, the required ZK proofs basically tell the contract, “I’m moving my knight; I’m not going to tell you where I moved my knight from, or where I moved it to, but this proof proves that it did in fact move in a legal L-shape.”

Other existing / suggested applications

- Verifying sufficient credit score - using range proofs
- replacing username / passwords M-Pin
- Supply Chain Transparency Origin Trail
- Software Verification Paper
- Digital ForensicsPaper
- Identity Sovrin
- Verifying Qualification TiiQu
- KYC ING
- Tax Qedit
- Legal evidence integrity Stratuum

ZKP Standards

There is an initiative to agree on standards within ZKP technology
ZKP Standards
"ZKProof is an open-industry academic initiative that seeks to mainstream zero-knowledge proof (ZKP) cryptography through an inclusive, community-driven standardization process that focuses on interoperability and security."