Zero-Knowledge Proof of Compression (zkPoC): A Cryptographic Primitive for Trustless Data Storage and Verification

Author: Kaleb Barnhart
Contact: X @xkal3b · Telegram @kb441

0. Abstract

We introduce the concept of Zero-Knowledge Proof of Compression (zkPoC), a cryptographic primitive designed to reduce storage overhead in blockchain and distributed systems. zkPoC enables a prover to demonstrate that a compressed file or dataset correctly decompresses to a known original, without requiring the verifier to store or reprocess the original data. This approach allows blockchains to anchor compressed representations plus succinct proofs instead of large payloads, thereby saving costs while preserving integrity. We show how zkPoC can be used for data availability layers, rollups, NFT storage, archival systems, and cross-chain state transfers, and we outline its integration into smart contracts for verifiable permanence.

1. Introduction

Blockchains face a fundamental tradeoff between decentralization and scalability. One of the greatest challenges is state bloat: as chains record more data, long-term costs of storage rise disproportionately. Ethereum, for example, requires rollups to post raw or lightly compressed transaction data on-chain as calldata, which is expensive and inefficient. Arweave, by contrast, provides permanent storage, but requires replicating large payloads that strain capacity. Existing methods such as Merkle proofs and hash commitments prove inclusion and integrity, but they do not address correctness of compression. In other words, they cannot guarantee that a compressed file truly expands back into the claimed original dataset. We propose zkPoC as a remedy: a system in which a prover stores compressed data plus a zero-knowledge proof ensuring the data decompresses faithfully to the original. This enables verifiable storage at scale and could radically reduce costs across blockchain platforms.

2. Related Work

2.1 Verifiable Computation Zero-knowledge proofs (ZKPs) are powerful cryptographic tools for verifiable computation. zkSNARKs and zkSTARKs provide succinct proofs that arbitrary computations were performed correctly, but their application has largely been in rollups and privacy systems rather than compression verification. zkPoC formalizes this application as a dedicated primitive. 2.2 Proof-of-Storage Filecoin introduced Proof-of-Replication and Proof-of-Spacetime to ensure that miners store client data. However, these proofs say nothing about compression correctness; they only establish that storage occurs. 2.3 Data Availability Layers Solutions such as Celestia, EigenDA, and Avail reduce blockchain load by dispersing shards of data, but they too assume validity of provided data chunks without verifying compression consistency. zkPoC fills this gap by explicitly connecting compression to verifiable computation.

3. System Model

3.1 Participants The zkPoC model includes three main actors: Prover, Verifier, and Clients. The Prover compresses an original file F to obtain C(F), then produces a zero-knowledge proof demonstrating that decompression yields F. The Verifier (e.g., a smart contract) validates this proof against the hash of F, denoted H(F). Clients can later reconstruct F from the compressed file and compare their own decompression result to H(F). 3.2 Workflow 1. Prover computes compressed form C(F). 2. Prover generates zk proof P such that D(C(F)) = F and H(F) matches a public commitment. 3. Verifier validates proof and stores only C(F), P, and H(F) on-chain. 4. Clients optionally reconstruct F. 3.3 Notation - F: Original file - C: Compression function - D: Decompression function - H: Cryptographic hash function - zkPoC statement: Prove knowledge of F such that H(F) = H* and D(C(F)) = F.

4. Protocol Design

4.1 Deterministic Compression Compression algorithms must be deterministic to ensure identical outputs for identical inputs. Common choices include DEFLATE, Brotli with fixed parameters, or Zstd with seeded dictionaries derived from file hashes. 4.2 zk Circuit Construction The zk circuit encodes the decompression process. This includes bit-level checks for Huffman decoding or dictionary lookups for LZ77. The output of decompression is hashed and compared against the claimed H(F). 4.3 Recursive Aggregation For scalability, large files can be split into fixed-size chunks. Each chunk is individually verified in a zk circuit, and recursive aggregation is applied to compress multiple proofs into one. This keeps verification efficient for smart contracts. 4.4 Commitments On-chain commitments include: - H(F): The canonical hash of the original. - C(F): The compressed representation, possibly chunked. - Proof: The succinct zk proof, aggregated if necessary. This architecture ensures verifiability without overwhelming blockchain storage.

5. Use Cases

5.1 Rollup Data Availability Ethereum rollups could post compressed transaction data with zkPoC proofs instead of full calldata, dramatically reducing gas fees while preserving verifiability. 5.2 NFT and Media Authenticity NFTs referencing large media files could store compressed versions plus zkPoC proofs on-chain, guaranteeing authenticity of artwork, video, or 3D models without bloating storage. 5.3 Decentralized Archival Libraries, governments, and scientific repositories could use zkPoC to anchor data permanently. Only proofs and compressed data remain on-chain, yet integrity is guaranteed. 5.4 Cross-Chain Bridges Cross-chain systems could exchange state snapshots as compressed blobs with zkPoC proofs, reducing bandwidth while ensuring correctness of transferred data.

6. Benefits

- Cost Efficiency: Blockchains avoid storing full payloads. - Verifiability: Zero-knowledge ensures compressed data faithfully represents the original. - Scalability: Chunking plus recursive proofs allow terabyte-scale datasets to be supported. - Privacy: Sensitive data need not be revealed; only commitments are public. - Interoperability: zkPoC can integrate with L1s, L2s, DA layers, and archival systems.

7. Challenges and Limitations

- Circuit Complexity: Encoding decompression in zk circuits is computationally heavy; optimization is required for practical deployment. - Proof Generation Costs: Provers may incur high costs for generating proofs over large datasets, though future zk hardware acceleration may mitigate this. - Codec Standardization: Without canonical compression specifications, interoperability may fail. zkPoC requires fixed parameters for codecs to ensure determinism. - Adoption Barriers: Integrating zkPoC into existing rollup or DA systems requires significant engineering, and teams may resist unless clear cost savings are demonstrated.

8. Conclusion

zkPoC is a novel primitive enabling compressed storage with verifiability guarantees. It directly addresses blockchain state bloat while maintaining trustlessness. By enforcing deterministic compression and leveraging zero-knowledge proofs, zkPoC allows blockchains to commit only compressed data plus succinct proofs, opening a path for more scalable and cost-efficient decentralized storage.

9. References

[1] Ben-Sasson, E., et al. "Scalable, transparent, and post-quantum secure computational integrity." (STARKs). [2] Bowe, S. "Halo Infinite Recursive Proof Composition." (ECC). [3] Filecoin Protocol: Proof-of-Replication and Proof-of-Spacetime. [4] Ethereum Rollup Data Availability discussions (EIP-4844, Proto-Danksharding).