Proof of Storage Time: Efficiently Checking Continuous Data Availability

Giuseppe Ateniese
Stevens Institute of Technology

Mohammad Etemad
Stevens Institute of Technology

Long Chen
New Jersey Institute of Technology

Qiang Tang
New Jersey Institute of Technology
Outsourced Storage is a common practice

Backup

Data sharing

Saving Cost
Example Case

A hospital stores medical imaging data on the cloud

Surgeons will consult these data during an emergency surgery

A brief downtime will cause a serious medical accident!

Continuous data availability is crucial
Mission and Business Critical Applications

Brief downtime may lead to serious negative consequences

- Lost of productivity
- Financial pain
- Damages to the business’ reputation
Threats to Continuous Availability

- Equipment failures
- Power outrages
- Malicious attackers
Cost of Continuous Availability

More replications

More hardware and software components

More complex administration

Continuous availability means a high price!
Verify Continuous Availability

A dishonest server would provide an inferior service

The client who paid a high price must verify the continuous availability
Decentralized Storage Market

Data Owner

Blockchain

File
Decentralized Storage Market

- Data Owner
- File
- Smart Contract
- Server

- $\pi$ must be succinct
- Verification must be cheap
Proof of Storage-time

- Definition
- Construction
- Instantiation
Proof of Storage-time

- Definition
- Construction
- Instantiation
PoSt Framework
Security Definition

Proof of Retrievability

- Goal: Verify data availability
- Security: Extractability

Proof of Storage-time

- Goal: Verify continuous availability
- Security: Continuous extractability
Continuous Extractability

Adversary → Extractor

Stop at time point $t$
Record the current state in the memory

Extractor → File
Proof of Storage-time

Definition

Construction

Instantiation
Naïve Attempts

Proof of Retrievability
- A challenge and response protocol
- Only certify availability at the time a valid proof is processed

Frequent PoR
- Inefficient communication and verification
- The client needs to be always online
Unsuccessful Attempts

Send PoR challenges in advance
  ◦ The prover may compute all PORs rapidly and discard the data

Send PoR challenges in the end
  ◦ The prover could keep data offline and retrieve them at the last moment
Filecoin’s proposal

1. Send PoR challenge $c_0$
2. Compute the PoR proof $p_0$
3. Let $c_1 = \text{Hash}(p_0)$
4. Compute the PoR proof $p_1$
5. ......
6. Send back all $c_i$ and $p_i$

Problem:
- No concrete delay guarantee
- Verification is inefficient
Verifiable Delay Function

\[ F(x) = y \ \pi \]

- To compute \( y \) for honest guys takes time almost \( T \)
- Malicious guy, even with parallel ability, can not get the result within time \( T \)
- Anyone can *efficiently* verify the correctness of the evaluation with a proof \( \pi \)
Warm-up Construction

1. Given a PoR challenge $c_0$
2. Generate the PoR $p_0$
3. Compute $(c_1, \pi_1) = VDF(p_0)$
4. Generates the PoR $p_1$
5. Etc.
6. Output all $c_i, p_i, \pi_i$
Problem

- The proof size is too large
- Verification is inefficient
\[ F(x) = y \]

- To compute \( y \) for honest guys takes time almost \( T \)
- Malicious guy can not get \( y \) within time \( T \) even with parallel computing

\[ F(x, \text{trapdoor}) = y \]

- Anyone with trapdoor can compute \( y \) within time significantly smaller than \( T \)
Main Construction

Verification:

\[ c_0, p_0, c_1, p_1, ... = c_0, p_0, c_1, p_1, ... \]

Aggregation

\[ \text{Hash}(c_0, p_0, c_1, p_1, ...) = \text{Hash}(c_0, p_0, c_1, p_1, ...) \]

Public Validation

\[ \text{tag} = \text{Hash}(\text{Hash}(c_0, p_0, c_1, p_1, ...)) \]
Decentralized Storage Market

Data Owner

File

Smart Contract

Server

Verify $tag = Hash(h)$
Efficiency of Each Procedure

Verification
- Extremely efficient

Proof
- Inherent cost

Storing
- Main left concern
Storing Procedure Optimization

- Adopting Hash based PoRs
- Precomputation
- Accelerate the PoR by Parallel Computation
- Others...
Storing Benchmark

![Graph showing the relationship between setup time and data available time for various data sizes (64 MB, 128 MB, 192 MB, 256 MB). The graph includes a line for each data size, with setup time on the y-axis and data available time on the x-axis. The data points illustrate the increase in setup time as the data available time increases for each data size.]
PoSt can verify continuous data availability

PoSt can be used to realize the decentralized storage market

Future work
- Optimization the storing procedure
- Make it stateless
- Achieve public verifiable
- More applications
Thank you for attention

Giuseppe Ateniese  
Stevens Institute of Technology

Mohammad Etemad  
Stevens Institute of Technology

Long Chen  
New Jersey Institute of Technology

Qiang Tang  
New Jersey Institute of Technology