PoR^2 – Proofs of Retrievability and Redundancy

Ludovic Barman

EPFL

August 27, 2015

BBC | Sign in | News | Sport | Weather | Shop | Earth | More

NEWS

Home | Video | World | UK | Business | Tech | Science | Magazine | Entertainment & Arts | Health | In

Technology

Google loses data as lightning strikes

19 August 2015 | Technology

Top St

Stock ma fears

4 hours

Deal agre tensions

6 hours

Carbon c money'

18 minut

Featur

"Google loses data as lightling strikes", BBC.com, 19.08.2015

- It is not a critical event, but some data were lost
- Which (naturally...) led the journalists to write advice such as :

  Still, no one should ever put their faith in the cloud — always backup your data locally - Geek.com, 21.08.2015

- Indeed, but what if we could ?
- Nothing is perfectly safe - but it would be nice to know the risks !

Table of Contents

Table of Contents

Introduction

Motivation

1. Cloud services are more widespread than ever
2. It offers numerous benefits, in particular for the mobile user
3. It simplifies data storage, synchronization and backup
4. One downside : Data availability is less obvious for users
5. Current cloud providers accept little liability for data loss

Introduction

Motivation

1. Cloud providers often replicate the data to reduce risks of loss
2. This replication is often done obliviously to the users
3. Some cloud providers allows users to choose the degree of replication
4. But users cannot check the veracity of these claims

Hence, users cannot assess nor control the risks they incurs, and need to trust the cloud providers.

Introduction

Motivation

1. Protocols (PDP/POR) exists to check the retrievability of a file
2. Recently, PDP have been extended for checking multiples replicas
3. In all those protocols, the users needs to replicate locally, and send the replicas

Numerical example: A user wants to store a 1GB file, and wants a degree of replication of 3 to reduce risks. He needs to sends around 3GB of data.

5. Worst, the cloud providers cannot check the replicas
6. Users can cheat by claiming encrypted files are replicas

Introduction

Goals

1. We design proofs that enable users to check
   1. the retrievability of their stored files
   2. the correctness of the replicas of the data

2. In addition, the cloud provider creates the replicas, which
   1. significantly lessen the burden on users
   2. prevents the users from cheating [1], hence allowing new business models for the cloud providers

[1] by claiming some encrypted data are replicas

Introduction

Team

Dr. Ghassan Karame

Prof. Dr. Frederik Armknecht

Dr. Jens-Matthias Bohli

Ludovic Barman

Introduction

Contribution

1. We propose a new formal model PoR^2 which extends the POR/PDP model and addresses new threats not covered so far
2. We propose new novel solution for data replication and retrievability, Mirror, and show that it is secure in the PoR^2 model
3. We implement and evaluate the performance of Mirror in a realistic cloud setting.

Introduction

Contribution

My main contributions :

1. We explore the solution space for constructing MIRROR, and propose three novel solutions
2. We analyze the performance and security of our proposals, and extract the relevant lessons
3. By leveraging our observations, we design a fourth solution, MIRROR, which establishes a strong balance between security and performance
4. We implement a prototype of all four devised solutions, and compare the performance in a realistic cloud setting

Table of Contents

Related work

Definitions

1. Two (similar) kind of protocols allows to check the integrity of a remote file :

Definitions of PDP and PoR

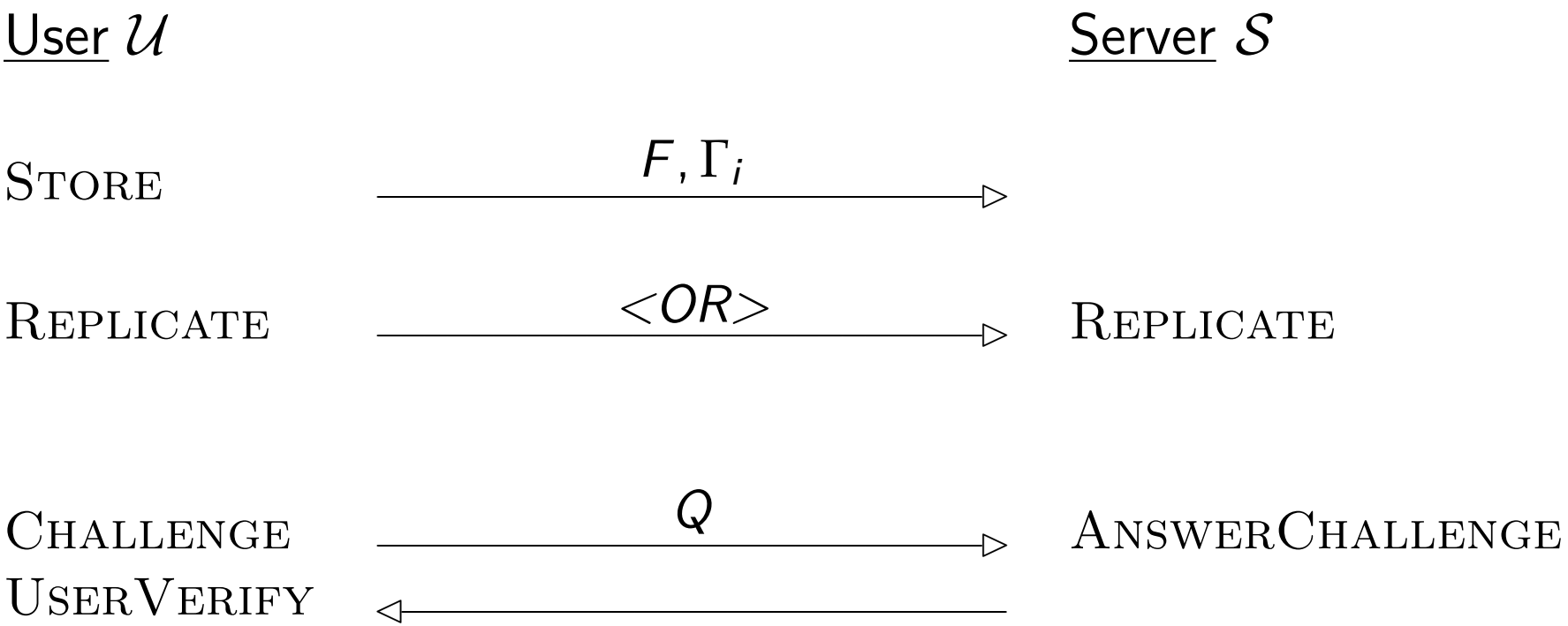
PDP. Provable Data Possession. Aims to validate the integrity of outsourced data efficiently and securely.
PoR. Proof of Retrievability. Aims to provide evidence that a verifier can retrieve and reconstruct the entire outsourced data.

2. Related work mainly try to improve performance, to give the ability to update an uploaded file (dynamic schemes), or to process file with multiple replicas
3. We focus on efficient multi-replica schemes

Related Work

Common architecture

Usually, (multi-replica) PDP and POR schemes share this common architecture :



Related work

Definitions (2)

Formal difference between PDP and PoR : PoR provide *extractability*.

Extractability

Given a sufficient number of runs of the VERIFY where the user accepts, there exists an algorithm EXTRACTOR, run by the user, that takes as input the secret parameters of the user, and has non-black box access to the machine of the server, that recovers the file with high probability.

Personal analysis : It's a desirable trait (which often hinders performance)

Related Work

Common architecture (2)

Usually, (multi-replica) PDP and POR schemes share these common traits :

1. The file F is usually considered to be encrypted by the user beforehand (for privacy purposes).

2. We consider that the user applies erasure-coding, i.e. if a given proportion of the file is recoverable, then the whole file is.

Table of Contents

Model

Introduction

We *quickly* summarize the most important aspects from the PoR^2 model:

1. The user $\mathcal{U}$ and the server $\mathcal{S}$ agree on a SLA on storing a file F. The server needs to:
   1. Store the file F in its entirety
   2. Store R replicas $F^{(1)}...F^{(R)}$ in their entirety
2. The protocol is the same as the one presented in common architecture.
3. Security goals :
   1. Extractability
   2. Storage allocation – *enough storage is allocated by* $\mathcal{S}$
   3. Correct replication – *replicas do not encode extra information*

Model

Adversary

- We consider the *rational attacker model* as done in related work
- Such adversary only deviates from the protocol if he gains something by doing so

For instance :

- the cloud provider tries to cut corners and save costs, by storing less than the agreed storage
- the user tries to save costs by uploading other information instead of the replicas

Table of Contents

Proposed solutions

Overview

Those are the common trait of our proposals :

1. We want the replication to happen in the cloud
2. We do not want the server to replicate on-the-fly
3. Hence, the replication is designed as a puzzle
4. This puzzle need to :
    1. require a significant time to compute for the server
    2. require significantly less time to compute for the user
    3. have solutions at least as large as the required storage for a replica
    4. be efficiently combined in the scheme

Our schemes are inspired from the private version of SW-POR.

$\sum$ Additive Scheme

Proposed solutions

Additive scheme - Protocol (1)

Client

STORE (File F, k_{PRF}):

Using the PRF :
... Pick $\alpha_1 \cdots \alpha_s$ from $\mathbb{Z}_{\phi(N)}$
... Pick $\text{secret}_1 \cdots \text{secret}_n$ from $\mathbb{Z}_N$

For $i \in [1, n]$:
... $\sigma_i \leftarrow \text{secret}_i + \sum_{j=1}^{s} \alpha_j \cdot m_{i,j}$
$\tau_U \leftarrow (k_{\text{PRF}}, n)$
Keep $\tau_U, \{\sigma_i\}$, may delete $\{m_{i,j}\}$ $\xrightarrow{\{m_{i,j}\}, \{\sigma_i\}}$ Server

Proposed solutions

Additive scheme - Protocol (2)

Server

REPLICATE (r) :
For all $i \in [1, n]$, $j \in [1, s]$:
... $m_{i,j}^{(r)} \leftarrow m_{i,j} + p_r^{E_r^{(i-1)s+j}}$

Client

CHALLENGE:
Choose l indices $i_1 \cdots i_l$ in $[1, n]$
Pick $v_{i_1} \cdots v_{i_l}$ at random in $\mathbb{Z}_p$
$Q = \{(i_k, v_{i_k})\} \; \forall \; k \in [1, l]$ $\xrightarrow{\qquad Q \qquad}$

Proposed solutions

Additive scheme - Protocol (3)

Server

$\sigma \leftarrow \sum_{(i,v_i)\in Q} \sigma_i \cdot v_i$

$\forall j \in [1, s]$,

... $\mu_j \leftarrow \sum_{(i,v_i)\in Q} v_i \cdot m_{i,j}^{(r)}$

$$\xrightarrow{\sigma, \{\mu_j\}}$$

Client

$\text{LHS} = \sum_{j=1}^{s} \alpha_j \cdot \mu_j + \sum_{(i,v_i)\in Q} v_i \cdot \text{secret}_i$

$\text{RHS} = \sigma + \sum_{j=1}^{s} \alpha_j \cdot P_j$, where

$P_j := \sum_{(i,v_i)\in Q} v_i \cdot p_r^{E_r{}^{(i-1)\cdot s+j} \bmod \phi(N)} \bmod (N)$

Accepts if LHS = RHS

Proposed solutions

Benchmarking

1. Prototypes made in Scala (run on JVM), run in realistic cloud setting

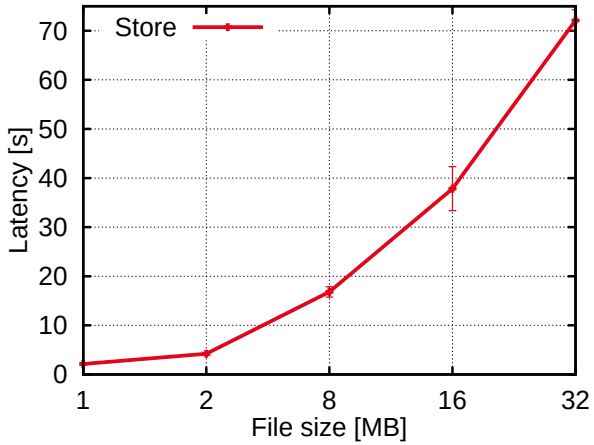
Parameter	**Value**
Machine Type	2 x Intel Xeon E5-2640
Number of cores per machine	24 cores
RAM per machine	32 GB
Network speed	100 Mbps
Mean packet latency	10 ms
Variance in packet latency	4 ms

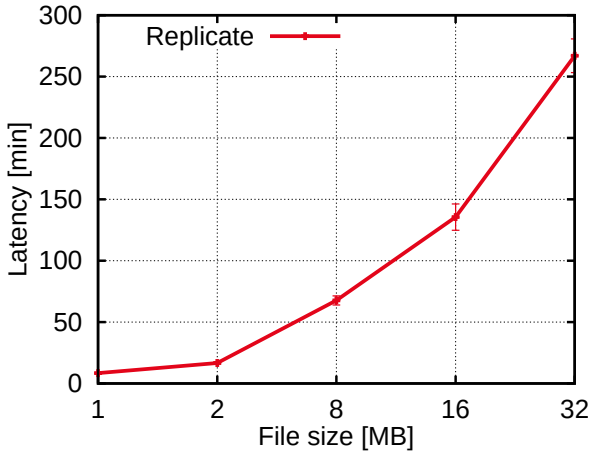
Table: Implementation Settings

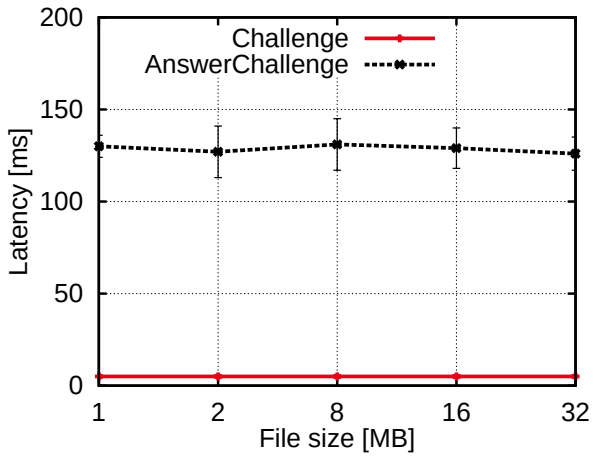
1. Each data point is averaged over 10 independant samples
2. 95% confidence intervals are shown where applicable

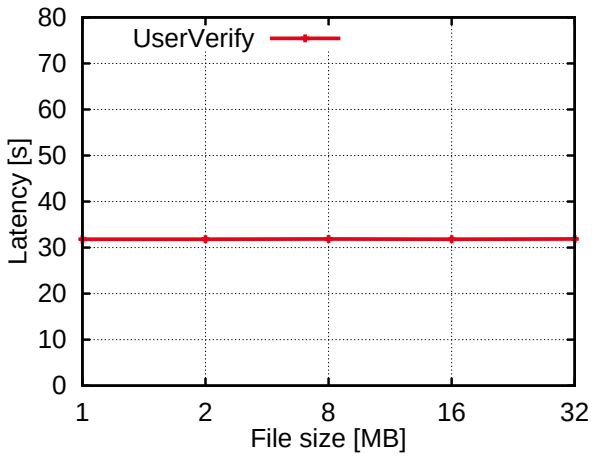
Proposed solutions

Additive Scheme - Performance









Proposed solutions

Additive scheme - Security Analysis (1)

- **Puzzle hardness**
  1. Puzzle of the form $p^E \mod N$, $E \gg N$, which is inherently a sequential process that is hard to parallelize
  2. Hence, multi-threaded server have no significant advantage over mono-threaded ones
  3. Assuming a server with bounded number of threads, we can detect for l big enough (puzzles starts to queue up)
- **Extractability**
  1. Each successful run, the user learns s equations $\mu_j \leftarrow \sum_{(i,v_i)\in Q} v_i \cdot m_{i,j}^{(r)}$
  2. With enough equations, we apply standard Gaussian elimination
  3. With x successful run, $x \geq l \cdot r$, we can recover r replicas

Proposed solutions

Additive scheme - Security Analysis (2)

- **Storage allocation**
  1. Suppose a malicious server that stores correctly a percentage ρ of a file
  2. Number of puzzles to be recomputed per challenge $l \cdot s \cdot (1 - \rho)$
  3. Let the puzzle time be Δ_{Puzzle}, the extra time needed to answer is

$$l \cdot s \cdot (1 - \rho) \cdot \Delta_{\text{Puzzle}}$$

  4. This needs to be tuned given the expected response time of an honest server
- **Correct replication**
  1. Correct by design

$\prod$ Multiplicative Scheme

Proposed solutions

Multiplicative scheme

We replace

1. every addition with a multiplication
2. every multiplication with a exponentiation

... which preserves the homomorphic properties of the scheme.

As a result :

1. everything is slightly slower
2. but we can use a speed-up in UserVerify

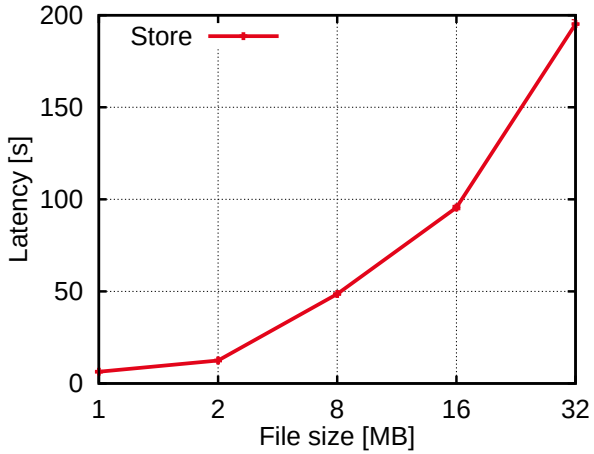
Proposed solutions

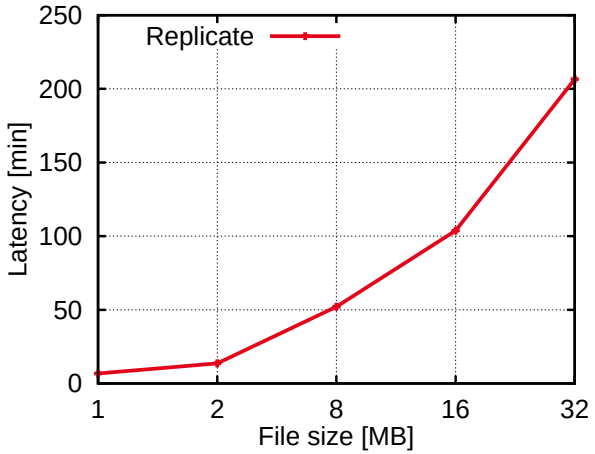
Multiplicative scheme - Security Analysis

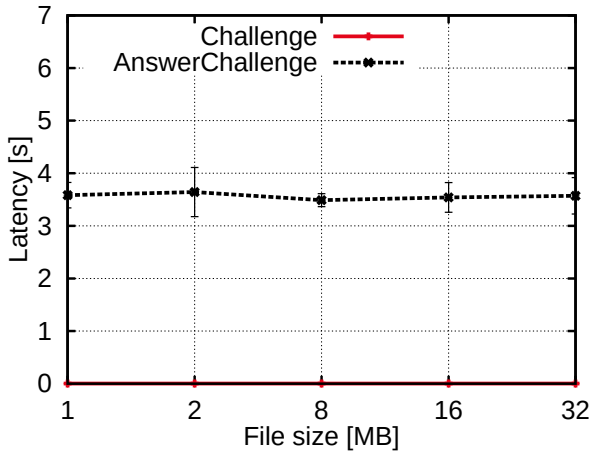
- **Puzzle hardness**
- **Extractability**
  1. Each successful run, the user learns s equations $\mu_j \leftarrow \prod_{(i,v_i)\in Q} {m_{i,j}^{(r)}}^{v_i}$
  2. This time, the solutions are not unique (2 or 4 possibilities per $m_{i,j}^{(r)}$)
  3. It will be fixed in MIRROR by forcing each sector to include a recognizable pattern
- **Storage allocation**
- **Correct replication**

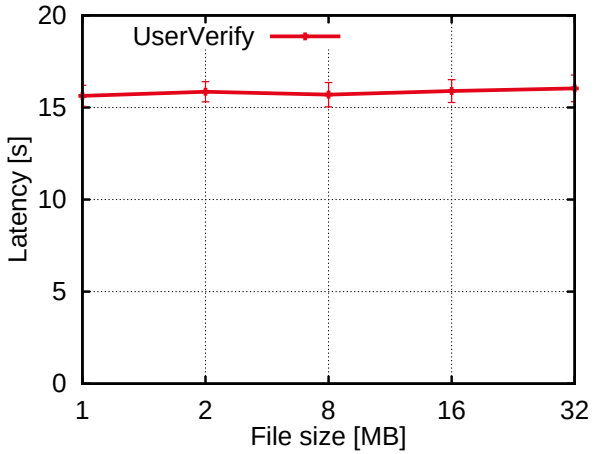
Proposed solutions

Multiplicative scheme - Performance









Scheme with precomputation

Proposed solutions

Scheme with precomputation

We now try to play with the replication time.

- We rearrange the previous $n \cdot s$ puzzles in :
  1. 1 precomputation (hopefully tunable as we want)
  2. $n \cdot s$ fast computations based on the precomputation

1. We fill a lookup-table with powers of p^{E^x}, $0 < x \leq \Lambda$, with $\Lambda = n \cdot s \cdot \lambda$
2. In the multiplicative scheme, we replace the puzzle

$$p^{E^{(i-1)\cdot s+j}}$$

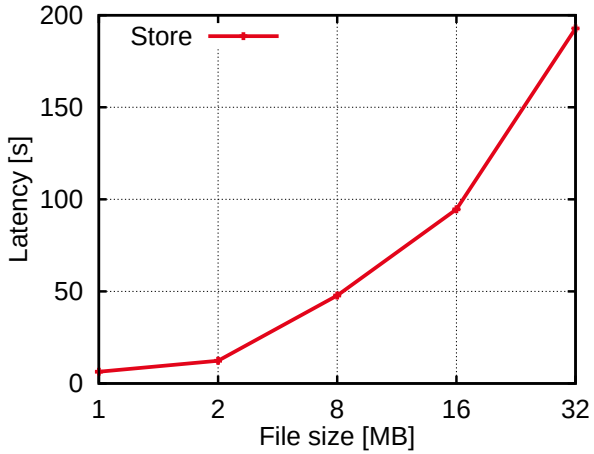
with

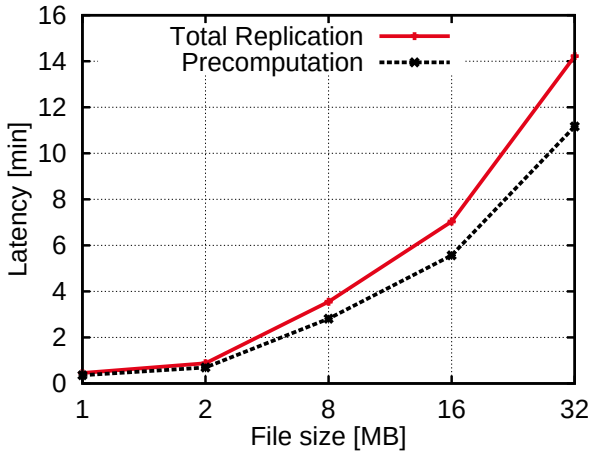
$$\prod_{i_k \in\ \Omega(i,j)} p^{E^{i_k}}$$

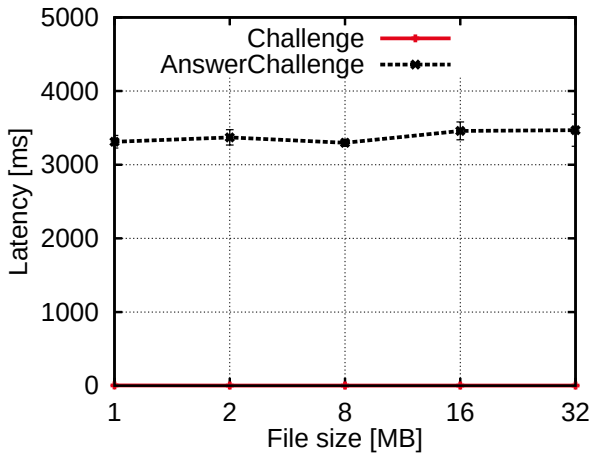
where Ω is a PRF that yield w values in $[1, \Lambda]$

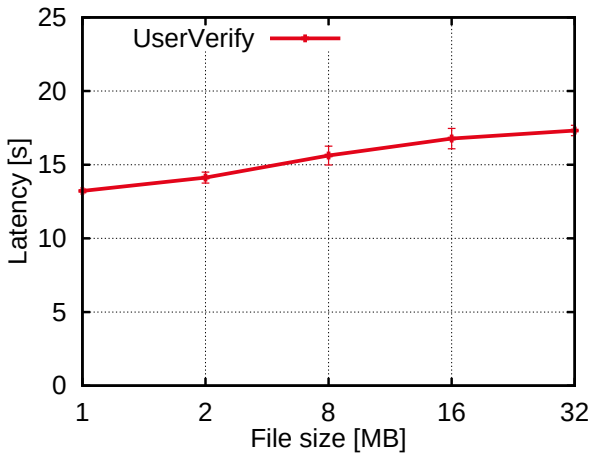
Proposed solutions

Scheme with precomputation - Performance









Proposed solutions

Scheme with precomputation - Security

1. Unfortunately, the malicious cloud provider has an efficient way of cheating
2. He computes a partial lookup table $p^{E^{k \cdot x}}, k \in \mathbb{N}, 0 < x \leq \Lambda$
3. He can freely choose k as a tradeoff between storage and time to answer a challenge
4. With k sufficiently high, the lookup table is smaller than a replica
5. To recompute a missing value p^{E^x} requires at most k exponentiations power E — hard to detect

Mirror

Mirror

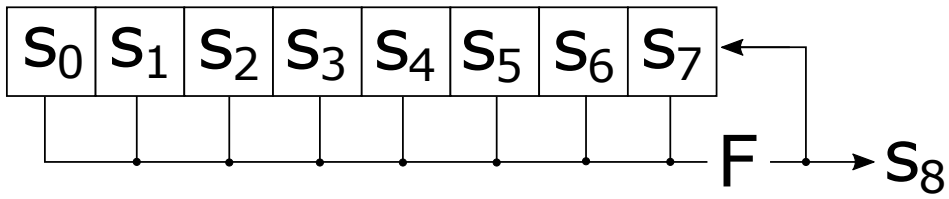
Introduction

1. We use the same construction as before (multiplicative scheme)
2. We change the way of producing the puzzles
3. We use a combination of LFSR and Time-lock puzzles

Mirror

LFSR

- Stands for Linear Feedback Shift Register
- Produces a random-looking sequence from an initial state S and a feedback function F



- Under certain conditions, a LFSR can produce the same output sequence than a shorter one
- We give the user the short LFSR, and the server the long one
- We increase the asymmetry by giving the elements mapped to $x \rightarrow g^x$, and making the server compute exponentiations

Mirror

LFSR

Client

LFSR of length $\lambda = 3$

$$F : s_4 = a_1 \cdot s_1 + a_2 \cdot s_2 + a_3 \cdot s_3$$

Server

LFSR of length $\lambda_2 = 5$

$$F' : s_6 = a'_1 \cdot s_1 + a'_2 \cdot s_2 + a'_3 \cdot s_3 + a'_4 \cdot s_4 + a'_5 \cdot s_5$$

The client generates an element g, and sends $\{t_i\} \leftarrow \{g^{s_i}\}$ instead of the initial state $\{s_i\}$ to the server.

$$\widetilde{F} : t_4 = g^{a_1 \cdot s_1 + a_2 \cdot s_2 + a_3 \cdot s_3}$$

$$\widetilde{F'} : t_6 = {t_1}^{a'_1} \cdot {t_2}^{a'_2} \cdot {t_3}^{a'_3} \cdot {t_4}^{a'_4} \cdot {t_5}^{a'_5}$$

- Both (L)FSR produce the same sequence, but one takes more resources
- The asymmetry is further increase when computing the product of output element

Proposed solutions

LFSR

- The output sequence of the server LFSR is be the puzzle / blinding factor
- We use *two* LFSR pairs
- The two server LFSR output sequence are called g_i and h_i
- The new sector is computed as follow

$$m_{i,j}^{(r)} \leftarrow m_{i,j} \cdot g_{(i-1)s+j} \cdot h_{ns-(i-1)s-j}$$

- One sequence is going forward, the other backwards

Mirror

Protocol (1)

Client

STORE (File F, k_{PRF}):

Using the PRF :
... Pick $\epsilon_1 \cdots \epsilon_s$ from $\mathbb{Z}_{\phi(N)}$

For $i \in [1, n]$:
... $\sigma_i \leftarrow \prod_{j=1}^{s} {m_{i,j}}^{\epsilon_j}$
$\tau_U \leftarrow (k_{\mathsf{PRF}}, n)$
Keep $\tau_U, \{\sigma_i\}$, may delete $\{m_{i,j}\}$ $\xrightarrow{\{m_{i,j}\}, \{\sigma_i\}}$ Server

Mirror

Protocol (2)

Server

REPLICATE (r) :

For all $i \in [1, n]$, $j \in [1, s]$: $m_{i,j}^{(r)} \leftarrow m_{i,j} \cdot g_{(i-1)s+j}^{(r)}$

For all $i \in [n, 1]$, $j \in [s, 1]$: $m_{i,j}^{(r)} \leftarrow m_{i,j}^{(r)} \cdot h_{(i-1)s+j}^{(r)}$

Client

CHALLENGE:

Choose l indices $i_1 \cdots i_l$ in $[1, n]$

Pick $v_{i_1} \cdots v_{i_l}$ at random in $\mathbb{Z}_p$

$Q = \{(i_k, v_{i_k})\} \; \forall \; k \in [1, l]$ $\longrightarrow Q$

Mirror

Protocol (3)

Server

$\sigma \leftarrow \prod_{(i,v_i)\in Q} \left(\sigma_i \cdot \prod_{j=1}^{s} \prod_{r\in[1,R]} m_{i,j}^{(r)}\right)^{v_i}$

$\forall j \in [1, s]$,

... $\mu_j \leftarrow \prod_{(i,v_i)\in Q} \; m_{i,j}^{(r)\,v_i}$

$\sigma, \{\mu_j\}$ ⟶

Client

$\text{LHS} \leftarrow \sigma \cdot \prod_{(i,v_i)\in Q} \left(\prod_{j=1}^{s} \prod_{r\in k} g_{(i-1)s+j}^{(r)} \cdot h_{(i-1)s+j}^{(r)}\right)^{-v_i}$

$\text{RHS} \leftarrow \prod_{j=1} \mu_j{}^{\epsilon_j + \|k\|}$

Accepts if LHS = RHS

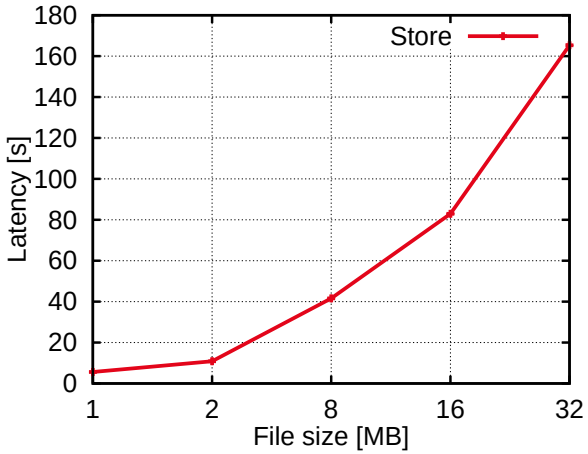
Proposed solutions

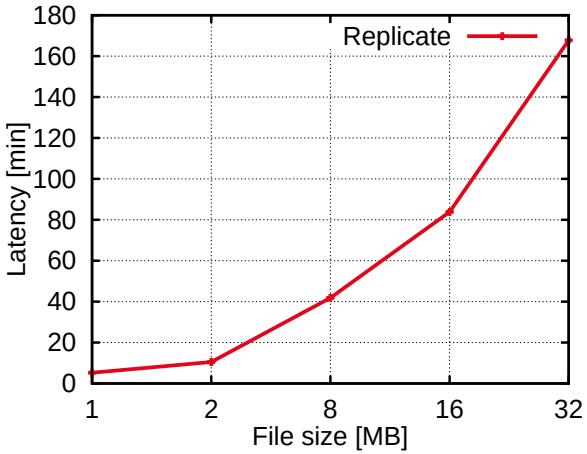
Mirror - Security Analysis

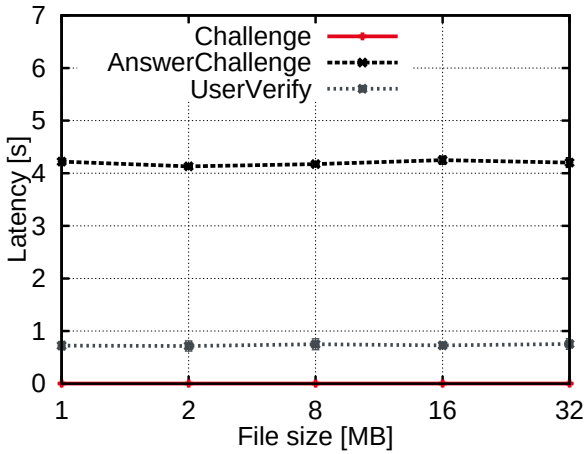
- **Puzzle hardness**
  1. The demonstration is outside the scope of this thesis
  2. There is a formal proof which gives a lower bound on the puzzle computation time
  3. This lower bound depends on the biggest LSFR coefficient, WLOG the first one α_1
  4. Even with this pessimistic lower bound, this time is sufficiently high to be detectable by the user
- **Extractability** - as in the multiplicative scheme
- **Storage allocation** - as in the additive scheme
- **Correct replication** - correct by design

Proposed solutions

Mirror - Performance (1)







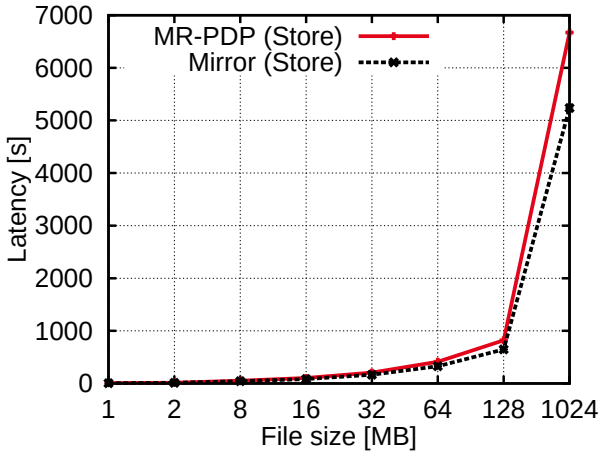
Proposed solutions

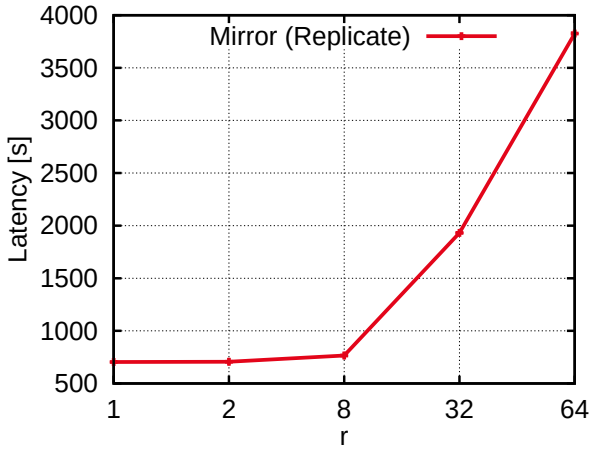
Mirror - Performance (2)

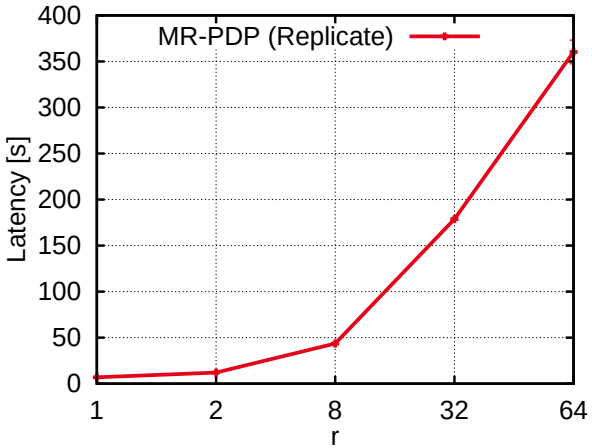
- The performance are now reasonable, especially in terms of AnswerChallenge and UserVerify
- We will compare the scheme against the scheme from Curtmola et al, MR-PDP

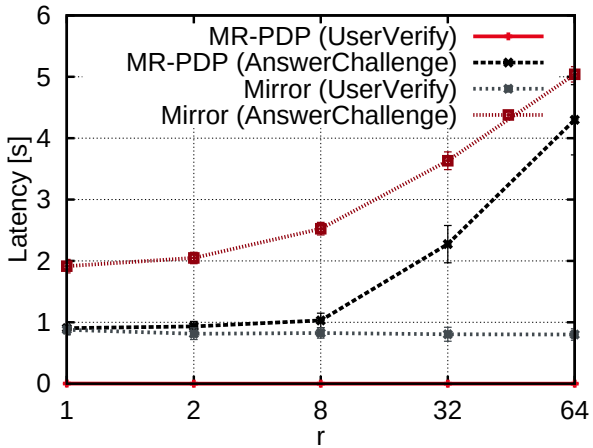
Proposed solutions

Mirror - Performance (3)



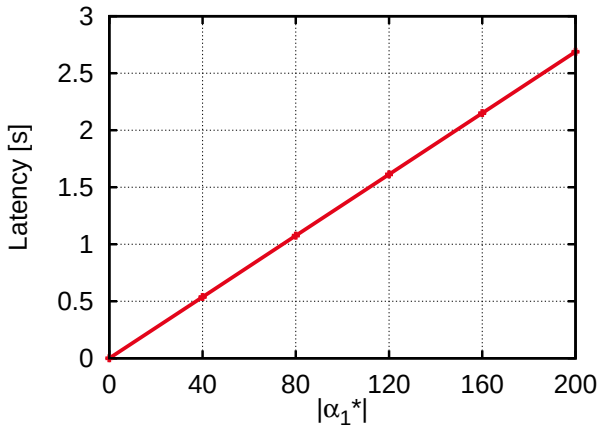


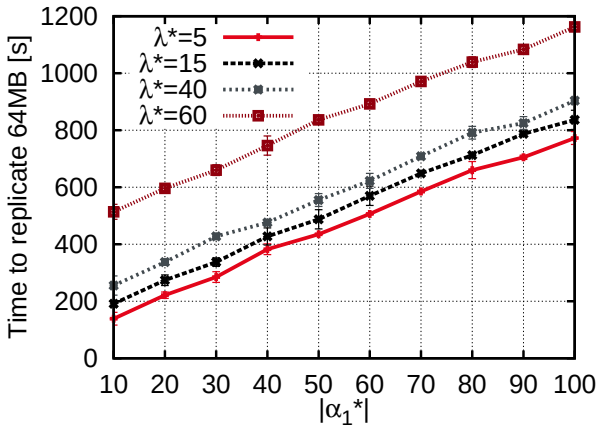




Proposed solutions

Mirror - Choice of parameters





- The bitsize of α_1 (and the size of the server LFSR) has an impact on the replication time
- The bitsize of α_1 has a big impact on the response time of a misbehaving cloud provider (that only stores 90% of the file)

Table of Contents

Conclusion

In this work,

1. we proposed four novel solutions that allow users to check the retrievability *and* redundancy of their files
2. we analyzed, implemented, and benchmarked them in a realistic settings, before discussing their strengths and weaknesses
3. the selected solution, MIRROR, incurs tolerable overhead and secure in the chosen model
4. All those solutions make the cloud replicate the data, lessening the burden on users, and preventing abuse from the users

References I

Shacham H., Waters B.

Compact proofs of retrievability.

Advances in Cryptology-ASIACRYPT 2008, 2008 - Springer.

Curtmola, R., Khan, O., Burns, R., Ateniese

MR-PDP: Multiple-replica provable data possession.

Distributed Computing Systems, 2008. ICDCS'08.