Applicative Crypto: Proof of Retrievability (PoR) Based Application

{nadav.shaked, hagar.aloni, yuval.terry}@post.runi.ac.il

March 13, 2025

Abstract

This project implements a **Proof of Retrievability (PoR) system** combined with a **file management system** to ensure data integrity, secure storage, and efficient file retrieval. The system incorporates **Reed-Solomon error correction** for data redundancy, **PoR authentication** for verification, and **Solana blockchain integration** for escrow-based storage subscriptions. Through this approach, we provide a secure and efficient method for data validation with minimal interaction between the buyer and the storage provider, ensuring trust and reliability in decentralized storage systems.

1 Introduction

1.1 Background and Motivation

Ensuring secure, verifiable, and retrievable file storage is a fundamental challenge in decentralized storage systems. This project implements a **Proof of Retrievability (PoR) system** alongside a **file management system** to provide both integrity verification and efficient data handling. By integrating **Reed-Solomon encoding**, **cryptographic authentication**, and **blockchain-based storage management**, we enable users to store files securely while ensuring their retrievability.

1.2 Problem Statement

A secure and efficient way to validate stored files is needed with minimal interaction between the buyer and the seller (storage server). The seller should require minimal knowledge about the stored files while ensuring data integrity. Additionally, the buyer must be able to restore their file using **error correction codes** if the seller corrupts it. This system addresses these challenges by implementing cryptographic verification techniques and blockchain-based economic incentives.

2 System Architecture

2.1 Components Overview

- **PoR Application:** Encodes & decode files using **Reed-Solomon error correction** and **PoR authentication**, and managing storage subscriptions on Solana.
- **Storage Server:** A Flask-based service for secure file storage, retrieval, and validation.
- **Storage Server Portal:** GUI for buyers to manage their stored files easily.
- **Solana API Gateway:** Manages interactions with the **Solana blockchain** for subscription-based storage.
- **Solana Blockchain:** Handles storage payments via escrow accounts.
- **Solana Explorer:** GUI that provides transaction and account status information.

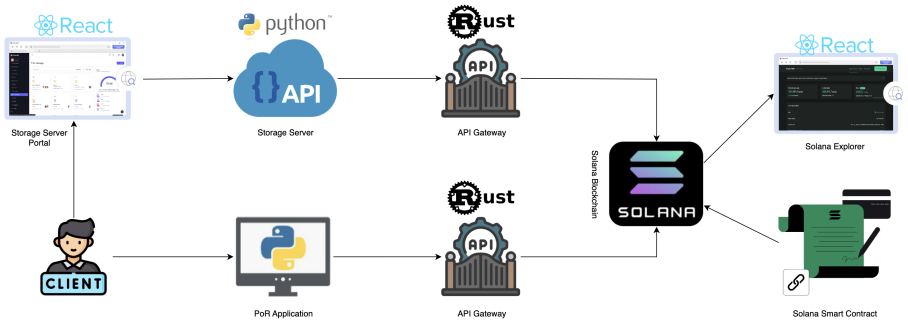

Figure 1: System Architecture

3 Implementation

3.1 Technologies Used

- **Cryptographic Library:** `bls12_381` for elliptic curve operations, including **pairing-based cryptography**.
- **Programming Languages:** Rust and Python.
- **Solana Blockchain:** Manages storage transactions.
- **Anchor Framework:** Facilitates Solana smart contract development.

3.2 Methods

- **Proof of Retrievability (PoR):** Uses Reed-Solomon encoding and authentication blocks for integrity verification.
- **Elliptic Curve Cryptography (ECC), Hash-to-Curve, and Pairing-Based Cryptography:** These cryptographic methods are used for secure encoding, verification, and integrity validation.
- **Cryptographic Random Value Generation:** Utilizes Python's `secret` library for generating cryptographic random values.
- **Solana Smart Contract Random Query Generation:** Uses Solana's **slot hash** to generate randomized values for validation queries.
- **Blockchain Integration:** Manages storage subscriptions through Solana smart contracts.

4 Challenges and Solutions

4.1 Challenges

- Missing locally decodable erasure code implementations.
- Finding a method to efficiently pass elliptic curve cryptographic data between Python and Rust.
- Implementing Python-like operations in Rust efficiently, like modulus and cryptography randomness.
- Running `bls12_381` operations on Solana smart contracts exceeded compute unit limits.
- Communication between multiple languages and Solana (Python, Rust) - compress and uncompressed elliptic curve points and transfer the points between the services.

4.2 Solutions

- Used Reed-Solomon encoding instead of locally decodable erasure codes.
- Used Rust's `BigInt` library and slot hash for random values.
- Mocked validation logic due to Solana's compute unit constraints.
- Created a centralized API server for Solana interaction with Docker-based scalability.

5 Results and Performance

- Secure storage management through blockchain-based escrow.
- Reliable file integrity checks using PoR authentication.
- Error detection and recovery mechanisms using Reed-Solomon encoding.

6 Future Enhancements

- Optimized Smart Contracts: Reduce costs on Solana transactions.
- Additional Cryptographic Enhancements: Improve efficiency of pairing-based operations.

7 Conclusion

This project successfully demonstrates **Proof of Retrievability (PoR)** techniques to enhance data integrity and retrievability. By integrating **Reed-Solomon encoding** and **Solana blockchain-based subscriptions**, it provides a secure and efficient storage solution.

8 References

- PoR Formula and Implementation
- PoR Concept Explanation