Proofs of Data Possession and Retrievability Based on MRD Codes*

Shuai Han[1], **Shengli Liu**[1], **Kefei Chen**[2], **Dawu Gu**[1]
[1]Department of Computer Science and Engineering
Shanghai Jiao Tong University, Shanghai 200240, China
[2]School of Science, Hangzhou Normal University, Hangzhou, China
shuaihan.sjtu@gmail.com, {slliu, kfchen, dwgu}@sjtu.edu.cn

Abstract

Proofs of Data Possession (PoDP) scheme is essential to data outsourcing. It provides an efficient audit to convince a client that his/her file is available at the storage server, ready for retrieval when needed. An updated version of PoDP is Proofs of Retrievability (PoR), which proves the client's file can be recovered by interactions with the storage server. We propose a PoDP/PoR scheme based on Maximum Rank Distance (MRD) codes. The client file is encoded block-wise to generate homomorphic tags with help of an MRD code. In an audit, the storage provider is able to aggregate the blocks and tags into one block and one tag, due to the homomorphic property of tags. The algebraic structure of MRD codewords enables the aggregation to be operated over a binary field, which simplifies the computation of storage provider to be the most efficient XOR operation. We also prove two security notions, unforgeability served for PoDP and soundness served for PoR with properties of MRD codes. Meanwhile, the storage provider can also audit itself to locate and correct errors in the data storage to improve the reliability of the system, thanks to the MRD code again.

Keywords: Data integrity, dependable storage, error localization, cloud computing.

1 Introduction

Data outsourcing to cloud storage reduces the storage cost and makes the data maintenance easier at both personal and business level. When clients move their data to a service provider for storage, they can enjoy the convenience of outsourcing storage with a relative low fee. However, they may also worry about the security of their data, as their data is out of their hands, and can be manipulated by the untrustworthy storage provider. The storage provider may maliciously delete some rarely accessed data to save space, or lose some data due to system failure. Hence, one of the main issues bothering the clients is whether their data is still available for retrieval if needed. The fact that even Amazon's S3, one of the best data outsourcing service provider, suffered significant downtime in 2008, makes this issue more critical.

A naive way to solve this issue is that a client retrieves all his/her data to check its authenticity and integrity from time to time. This costs lots of communication bandwidth, and deviates from the original appealing features of cloud storage.

A better approach is that a client performs an audit of the storage provider. A successful audit verifies that the provider stores the data, without the retrieval and transfer of all the data from the provider to the client. The ability of a storage system to generate proofs of possession of client's data, without having to retrieve the whole file, is called *Proofs of Data Possession*(PoDP). PoDP only concerns the provider's proof of data possession. An improved version of PoDP is known as *Proof of Retrievability* (PoR), which enables the provider to convince the client that the original data could be recovered through enough interactions between the client and provider.

*Corresponding author: Shengli Liu. Funded by NSFC Nos.61170229, 61133014, 61373153, Innovation Project (No.12ZZ021) of Shanghai Municipal Education Commission, and Specialized Research Fund (No.20110073110016) for the Doctoral Program of Higher Education, Major State Basic Research Development Program (973 Plan)(No.2013CB338004).

More precisely, a PoDP/PoR scheme is implemented through an audit in which a client interacts with the storage provider. In each interaction, the client issues a query, and the provider feeds back a response. The client verifies the consistency of the response. After several interactions, the client decides whether the audit is successful or not according to some audit strategy. A successful audit indicates the client's confidence of the data possession of the provider. Proof of retrievability means that if a storage provider succeeds in an audit, then there exists an extractor, which can extract all the data from interactions with the provider.

In the above introduction, we assume that clients implement the audit, and this is called PoDP/PoR with private verification. Audit can also be performed by any trustworthy third party, and in this case it is called PoDP/PoR with public verification. Public verification often involves complicated computation like modular exponentiations or bilinear pairings, which makes the corresponding PoDP/PoR inefficient. In this paper, we only study PoDP/PoR with private verification.

To evaluate a PoDP/PoR scheme, both "system" and "crypto" criteria should be considered. As suggested in [14], the "system" criteria includes *computational complexity* of the storage provider and clients, *communication complexity* between the provider and clients, *statelessness* and *unbounded use* of the interactions in the audit. The *computational complexity* and *communication complexity* determine the efficiency of the PoDP/PoR scheme. The *unbounded use* means that the number of possible interactions should not be bounded to some fixed threshold. The *statelessness* requires there should not be state to maintain and update between interactions in an audit. The "crypto" criteria requires that a successful audit guarantees the provider server is actually storing the file (as for PoDP), with overwhelming probability, and the existence of an extractor to recover data with interactions between the client and the provider (as for PoR).

In a PoDP/PoR system, the storage provider is in charge of responding to all the clients for auditing. It is very important for the storage provider to implement light-weight computation, otherwise the provider will be the bottleneck of the system. However, almost all the available PoDP/PoR schemes associate the storage provider with heavy computation, like modular exponentiations or modular multiplications.

1.1 Related Works

Deswarte et al. [5] (2003) and Filho et al. [8] (2006) designed schemes to prove the data availability based on RSA. The schemes require huge amount of modular exponentiations. Similarly, Schwarz et al. [13] (2006) also proposed to employ algebraic signatures for integrity check of storage. The highly computational complexity makes the aforementioned schemes impractical.

It was Juels and Kaliski [10] (2007) who first formulated the concept of Proof of Retrievability and defined the corresponding security model. Dodis, Vadhan and Wichs [7] (2009) presented Proof of Retrievability Codes (PoR codes) for PoR schemes, and they also divide PoR schemes to "bounded vs. unbounded" according to the number of possible queries in an audit, and "knowledge soundness vs. information soundness" according to whether the extractor is probabilistic polynomial time (PPT) or not. Naor and Rothblum [11] (2005) designed "sublinear authenticator", which can be considered as "information-theoretic unbounded-use of PoR schemes" according to Dodis, Vadhan and Wichs [7].

More constructions of PoR schemes are given by Ateniese et al. [6] (2007), Shacham and Waters [14] (2008), or Schwarz and Miller [13] (2006). When authentication is achieved with a digital signature scheme, one gets a PoR scheme with public verification.

Among all the PoDP/PoR schemes, the most charming ones are those with homomorphic authenticators. These schemes were identified as *homomorphic linear authenticator schemes* in [7]. Ateniese et al. [6] (2007) proposed constructions of PoDP scheme (called the ABCHKPS scheme) with homomorphic verifiable authenticator based on the RSA assumption. The PoDP security model was further extended to be against arbitrary adaptive PPT adversaries by Shacham and Waters [14](2008), who presented PoR schemes (called the SW scheme) both for private verification and public verification, and their security was given in the full security model. The file of a client is divided into blocks, and an authenticator (tag) is computed for each block. In the audit, the client samples blocks/authenticators (tags). The authenticator (tag) helps the owner of the file to check the integrity of the block with an authentication key. Because of the homomorphic property, multiple file blocks resp. authenticators can be combined into a single aggregated block/authenticator pair. And the verification is simplified to integrity check of the aggregated block. Therefore, both the computational complexity and communication complexity of

an audit are greatly reduced, due to the homomorphic property of authenticators.

Wang et al. proposed [16] (2012) to integrate PoDP with dependable storage. They used Reed-Solomon(RS) erasure-correcting codes to encode the original file of the client before storage. The audit of PoDP will locate erasures which are later corrected with RS decoding. Their scheme is bounded and only the client is able to implement the erasure-correcting. At the same time, erasure correction needs interactions between the client and each individual servers to locate erasures, recover the whole encoded file, and distribute the file among all the distributed servers. This imposes a great burden to clients' local computation, storage, and management, and again deviates from the original goal (free of data management) of cloud storage.

Next, we briefly review two seminal schemes. The first one was proposed by Ateniese et al. [6] (2007) and referred to the ABCHKPS scheme. The second one was proposed by Shacham et al. [14] (2008) and referred to the SW scheme.

1.2 The ABCHKPS Scheme

The ABCHKPS scheme proposed by Ateniese et al. [6] (2007) is based on the RSA assumption in the Random Oracle security model. It is roughly recalled as follows.

Homomorphic Tag Generation. Each file M is divided into blocks of 1024-bit, i.e., $M = (M_1, M_2, \cdots, M_w)$. For each block M_i, an RSA-like signature is computed as a tag T_i.

Interactions in Audit. In an audit, the client will interact with the storage provider in the form of challenge/response. When a challenge binary vector $(v_1, v_2, \cdots, v_w)$ of weight c is issued, the provider computes an aggregated tag, whose computation is dominated by $g^{\sum_{i=1}^{w} v_i M_i} \mod N$. Here N is a 1024 modulus, and g is a random element from the Quadratic Residue subgroup of $\mathbb{Z}_N^*$. Verifying the consistence of the provider's response is just the RSA signature verification.

The scheme presented a security analysis about a malicious storage provider. If the provider deletes t blocks of file M, then the scheme cannot detects the provider's misbehavior with a probability $\zeta = \binom{w-t}{c} / \binom{w}{c}$.

Remember c is the weight of the challenger vector, i.e., the number of sampled blocks, and w the total number of blocks in file M. If $c = 460$ and $t = 0.01w$, i.e., 1% fraction of M is deleted, ζ can be as small as 1% according to [6].

The probability ζ suggests a necessary condition on auditing strategy. Only if the storage provider correctly answers the challenge vector with probability larger than ζ, does the audit claim success.

Now we give a simple analysis of the scheme.

- To answer a challenge in the audit, the computation of the storage provider is dominated by $g^{\sum_{i=1}^{w} v_i M_i} \mod N$, which needs about c modulo exponentiations with a 1024-bit modulus. The reason is that the exponent $\sum_{i=1}^{w} v_i M_i$ is about c bits longer than a 1024-bit block. Without the secret factorization of N, the provider can not reduce the exponent $\sum_{i=1}^{w} v_i M_i$ with $\phi(N)$ before the computation of modular exponentiation.

- If the storage provider deletes only a few blocks, the stored file is not the original one any more, but it is very hard to detect the provider's misbehavior. Let $t = 1, c = 460$, M be a file of 1GB, Then $w = 2^{23}$, but $\zeta = (w - c)/w \approx 100\%$.

Therefore, the ABCHKPS scheme only detects the service provider's misbehavior when a large portion of the file is missing. In the mean time, the computational overhead of the service provider for audit is heavy.

1.3 The SW Scheme

The scheme presented by Shacham and Waters [14] is called the SW scheme. A new security notion, namely ϵ-soundness, was proposed in the SW scheme. If the service provider correctly answers a client's challenges with probability at least ϵ, it is possible for the client to recover the file from enough interactions with the provider. ϵ-soundness suggests a sufficient condition on the auditing strategy. As long as the

storage provider correctly answers the challenge vector with probability at least ϵ, the audit can claim success.

The idea of the SW scheme is as follows.

Erasure Encoding. The original file M' is encoded into M with an erasure code, such that the retrieval of ρ fraction of M is able to recover the original M'.

Homomorphic Tag Generation. The encoded file M is divided into w blocks of the same length (the last block may be padded to achieve the length): $M = (M_1, M_2, \cdots, M_w)$. A tag σ_i is generated from each block M_i with help of an authentication key K. The final file M^* consists of the blocks and their tags.

Interactions in Audit. In an audit, the client will interact with the storage provider in the form of challenge/response. Let $(v_1, v_2, \cdots, v_w)$ be a challenge vector, where $v_i \in B \subseteq \mathbb{F}_p$. The provider can generate an aggregated pair (μ, σ) of message and tag, where $\mu = \sum_{i=1}^{w} v_i M_i$ and $\sigma = \sum_{i=1}^{w} v_i \sigma_i$. Given the prover's response (μ', σ'), the client is able to use the authentication key K to verify whether (μ', σ') is consistent or not. A consistent pair (μ', σ'), i.e., the pair that passes the verification, must guarantee $\mu' = \sum_{i=1}^{w} v_i M_i$ with overwhelming probability.

To prove that the original file M' can be retrieved from a storage provider who feeds back consistent responses upon queries with some probability ϵ, an extractor is constructed to interact with the storage provider to recover M'. If the storage provider's reply (μ, σ) passes the verification, we call that (μ, σ) is consistent to the challenge vector $(v_1, v_2, \cdots, v_w)$. Firstly, it is necessary to prove that a consistent reply (μ, σ) is in fact a linear combination of the blocks, i.e., $\mu = \sum_{i=1}^{w} v_i M_i$, with overwhelming probability. Secondly, if ϵ is big enough, the consistent responses will bring more and more information about M' in the form of linear combinations $\mu = \sum_{i=1}^{w} v_i M_i$. When enough consistent replies are collected, a ρ fraction of the encoded file M can be recovered by solving a system of equations. Finally, with decoding algorithm of the erasure code, ρ fraction of the encoded file M uniquely determines the original file M'.

Although the SW scheme enjoys short responses due to the homomorphic verifiable tags, there are some other problems with the SW scheme.

- The Erasure Encoding imposes a heavy computational burden to the clients, and makes the SW scheme unfriendly to dynamic update of the clients' files.

  The employment of erasure correction code in [14] [7] is directly served to the security proof: it is much easy for the extractor to recover ρ fraction of a file than the whole one. However, this imposes a great computational burden for the client to encode the original file M' into M. If M' has w' blocks, the error-erasure encoding will extend the w' blocks to w blocks, then $\rho = \frac{w'}{w}$. The expansion rate is defined as $\theta = \frac{w}{w'}$. For a security level of 80 bits, all the operations should be over finite field $\mathbb{F}_p$ with p at least 80 bits. However, such an encoding needs at least $O(w'^2)$ multiplications over the prime field $\mathbb{F}_p$. For example, if a 1GB file M' is divided into 2^{20} blocks of length 1KB, then the encoding needs at least $(1024 \times 8/80) \times 2^{40} > 2^{46}$ modular multiplications over $\mathbb{F}_p$. Such a computational complexity served to store a file is unbearable to clients.

  On the other hand, the original w' blocks are encoded to w blocks. When one of the w' blocks is modified (deletion, insertion, update), at least another $w - w'$ blocks changes correspondingly.

- The challenge vector $(v_1, v_2, \cdots, v_w) \in (\mathbb{F}_p)^w$ cannot be reduced to a binary vector to save communication and computational overhead. The challenge vector implies the aggregated response should be $(\sum_{i=1}^{w} v_i M_i, \sum_{i=1}^{w} v_i \sigma_i)$. To reduce the communication band and the computational overhead of the storage provider, the SW scheme suggests that only l ($l \leq w$) random locations in the vector are chosen to take values from a small set $B \subseteq \mathbb{F}_p$. If $B = \{1\}$, the computation of $(\sum_{i=1}^{w} v_i M_i, \sum_{i=1}^{w} v_i \sigma_i)$ is reduced to addition over the prime field $\mathbb{F}_p$. Unfortunately, $B = \{1\}$ gives an attack to the security of the SW scheme, as shown in [14]. Therefore, it is inevitable for the storage provider to use multiplications over $\mathbb{F}_p$ to prepare the aggregated response.

- Theoretically, the storage provider is able to correct errors in M if an Error-Correction Code instead of an Erasure Correction Code is involved. However, the decoding algorithm is much slower than the encoding algorithm, and computational complexity of decoding is significantly larger than $O(w'^2)$, where w' is the number of blocks of the original file.

1.4 Our Contributions

In this paper, we proposed a PoDP/PoR scheme based on MRD codes. The main idea is as follows.

Divide the original file M into blocks $M = (M_1, M_2, \cdots, M_w)$, each block being a matrix of $t \times k$ over a small field $\mathbb{F}_q$. Block M_i is encoded into a MRD codeword C_i, which is a matrix of $t \times n$ over $\mathbb{F}_q$, for $i = 1, 2, \cdots, w$. Then each codeword C_i is used to generate a tag $\boldsymbol{\sigma}_i = C_i \cdot \mathsf{k} + \mathsf{k}_i$, where k, k_i, $i = 1, 2, \cdots, w$ are authentication keys, and $\boldsymbol{\sigma}_i, \mathsf{k}, \mathsf{k}_i$ are vectors over $\mathbb{F}_q$. The final file to be stored in the server is $M^* = (C_1||\boldsymbol{\sigma}_1, C_2||\boldsymbol{\sigma}_2, \cdots, C_w||\boldsymbol{\sigma}_w)$.

A challenge vector $(v_1, v_2, \cdots, v_w)$ is just a random sequence of length w over the field $\mathbb{F}_q$. The corresponding valid response should be a linear combination of the blocks $(\sum_{i=1}^{w} v_i \circ C_i, \sum_{i=1}^{w} v_i \circ \boldsymbol{\sigma}_i)$, where $\circ$ is scalar multiplication over $\mathbb{F}_q$.

The validity of the response can be verified with $\sum_{i=1}^{w} v_i \circ \boldsymbol{\sigma}_i = (\sum_{i=1}^{w} v_i \circ C_i) \cdot \mathsf{k} + \sum_{i=1}^{w} v_i \circ \mathsf{k}_i$, using the authentication key k, k_i, $i = 1, 2, \cdots, w$. Pseudo-random functions are used to generate k_i to reduce the local key storage.

The linearity of the MRD code results in homomorphic authenticators (tag), just like the scheme by Shacham and Waters (the SW scheme). However, our proposal has different features from the SW scheme.

- **Efficient computation by the service provider.** The MRD encoding plants algebraic structure in blocks so that the generation of tags and aggregation of messages and tags can be computed over smaller field $\mathbb{F}_q$. As to the instantiation of $\mathbb{F}_2$, the challenge vector $(v_1, v_2, \cdots, v_w)$ can be taken as a random binary sequence of length w, and the computation of the storage server for the response is simply XOR. This is the most efficient operation for the storage server.

- **Mild amount of pre-computation by the client.** The MRD encoding is applied block-wise. If we take a file as a matrix. Then each row, as a block, is encoded into an MRD codeword, in contrast to each column being encoded into an Erasure Correction Codeword in the SW scheme. The file encoding involves w MRD encoding, so the encoding computational complexity is of $O(w)$ in terms of MRD encoding (compared with the $O(w^2)$ complexity of the SW scheme). When parameters of the MRD code are fixed, the MRD encoding consumes a constant amount of computation.

- **Our proposal considers both PoDP and PoR with two security notions, which justify both the sufficiency and necessity of the audit strategy.** We define two security notions served for both the sufficiency and necessity of the audit strategy, namely **unforgeability** and **ϵ-soundness**. **Unforgeability** considers the cheating probability of storage provider in one interaction of challenge-response when some blocks are missing. It serves for the proof of data possession. **ϵ-soundness** considers the proof of retrievability, which suggests that if the provider answer challenges with consistent responses with probability at least ϵ, it is possible for an extractor to retrieve the original file. The security proof strongly relies on the property of MRD codewords. The algebraic property of MRD codewords makes an adversary impossible to forge a response, which passes client's verification but is not computed as it is supposed to be. This helps to prove the **unforgeability** and **ϵ-soundness** of the scheme and justify the rationality of the audit strategy. To prove the **ϵ-soundness** of the scheme, we do not rely on an extractor's ability to recover some ρ fraction of the encoded file M^*, as the SW scheme did. We rely on the randomness of each entry v_i in the challenge vector $(v_1, v_2, \cdots, v_w)$ and the algebraic structure of MRD codewords instead.

- **Efficient error location and correction for dependable storage.** The linearity of the MRD codes also enable the storage provider to implement an efficient self-audit. If there is a failure in a self-audit, it is possible for the provider to locate the error block and fix it with MRD decoding. To locate an error, binary search applies with complexity of $O(\log w)$. Compared with other error-correction codes, an MRD code is more powerful since it is capable of correcting row erasures, column erasures and rank errors.

1.5 Performance Comparison

Below gives a comparison among the ABCHKPS[6], SW[14], and our schemes. For a 1GB file, the block size is 1024-bit(as suggested in the ABCHKPS scheme [6]), 1KB(the SW scheme [14]), 4KB(as suggested

in Section 5, our proposal). The block number is $w = 2^{23}$, 2^{20}, 2^{18} respectively.

Unforgeability defines the successful probability ζ of a service provider in an interaction when t blocks of the file are deleted. It is closely related to the sampling strategy. The SW scheme has the same sampling strategy as the ABCHKPS scheme, hence they share the same unforgeability. When sampling c blocks among the totally w blocks for an interaction, ζ can approximately be estimated by the probability that no deleted block is sampled, i.e., $\zeta \approx \binom{w-t}{c} / \binom{w}{c} \geq (1 - t/(w - c + 1))^c$. Let the number of sampled blocks for each interaction be $c = 460$, as suggested in [6]. We have a different sampling strategy, namely, each block is sampled with probability 1/2. Therefore $\zeta \approx 1/2^t$. It is easy to see that the unforgeability ζ of our scheme is better, especially when the fraction of deleted blocks is small. To obtain the same unforgeability as our scheme, both the ABCHKPS scheme and the SW scheme have to sample about half of the blocks, i.e., $c \approx \frac{w}{2}$.

Suppose the multiplication in $\mathbb{F}_p$ has computational complexity $O((\log p)^2)$ and exponentiation in $\mathbb{F}_p$ has computational complexity $O((\log p)^3)$ in terms of bit-XOR. The SW scheme takes a 80-bit p for $\mathbb{F}_p$. Our proposal takes a binary field as suggested in Section 5.

With the unforgeability of our scheme as a benchmark, we show the computation complexity of the three schemes in Table 1. The table includes precomputation, computation of the client in one interaction, and computation of the server in one interaction.

Computation \ Scheme	ABCHKPS	SW	Ours
Precomput. (bit-XOR)	$O(2^{53})$	$O(2^{59})$	$O(2^{49})$
Client Comput. (bit-XOR)	$O(2^{31})$	$O(2^{31})$	$O(2^{31})$
Server Comput. (bit-XOR)	$O(2^{52})$	$O(2^{38})$	$O(2^{33})$

Table 1: Computational Complexity of the ABCHKPS, SW and Our Schemes

It should be noted the difference between the ABCHKPS scheme, our scheme and the SW scheme. In the ABCHKPS scheme and our scheme, when one block is missing in the storage server, theoretically the storage provider does not possess the client's file any more. When some block is corrupted, this specific block might be able to be recovered with MRD decoding in our scheme, but not in the ABCHKPS scheme. As for the SW scheme, the fact that some blocks are missing does not mean that the original file can not be recovered, due to the way error-correction code is used.

2 PoDP/PoR Scheme: Definition and Security Model

2.1 Notations and Assumptions

We use the following notations throughout the paper.

- $\mathbb{F}_q$: a finite field with q elements, i.e., $GF(q)$.
- λ: security parameter, which is an integer tending to infinity. The current security standard requires that $\lambda \geq 80$.
- k: a column vector of dimension t over $\mathbb{F}_q$, or an element in $\mathbb{F}_{q^t}$.
- k^T: transpose of the vector k.
- $(C_1, \cdots, C_n)^{\mathsf{T}}$: it represents $\begin{pmatrix} C_1 \\ \vdots \\ C_n \end{pmatrix}$ instead of $\begin{pmatrix} C_1^T \\ \vdots \\ C_n^T \end{pmatrix}$, where $C_1, \cdots, C_n$ are matrices.

- $\circ$: scalar multiplication over $\mathbb{F}_q$. Let $\mathsf{k} = \left(k^{(1)}, k^{(2)}, \cdots, k^{(n)}\right)^T \in (\mathbb{F}_q)^n$, and $v \in \mathbb{F}_q$. Then $v \circ \mathsf{k} = \left(v \cdot k^{(1)}, v \cdot k^{(2)}, \cdots, v \cdot k^{(n)}\right)^T$, where $\cdot$ is the multiplication over $\mathbb{F}_q$.
- $\vec{\mathsf{a}}$: a row vector of column vectors, i.e., $\vec{\mathsf{a}} = (\mathsf{a}_1, \mathsf{a}_2, \ldots, \mathsf{a}_n)$, where a_i is a column vector over $\mathbb{F}_q$. It can also be regarded as a matrix over $\mathbb{F}_q$. Then $v \circ \vec{\mathsf{a}} = (v \circ \mathsf{a}_1, v \circ \mathsf{a}_2, \cdots, v \circ \mathsf{a}_n)$.
- $A||B$: the concatenation of A and B.
- $|S|$: the cardinality of set S.
- $s \leftarrow S$: sample element s uniformly at random from set S.
- $|s|$: the bit length of string s.
- $a \leftarrow \mathsf{Alg}(x)$: run the algorithm $\mathsf{Alg}(\cdot)$ with input x and obtain a as an output, which is distributed according to the internal randomness of the algorithm.
- A function $f(\lambda)$ is *negligible* if for every $c > 0$ there exists a λ_c such that $f(\lambda) < 1/\lambda^c$ for all $\lambda > \lambda_c$.
- PRF: pseudo-random function.
- PPT: probabilistic polynomial time.

To reduce communication band and local storage, we will use pseudo-random functions to generate the key sequence k_i, $i = 1, 2, \cdots, w$. Below define pseudo-random functions.

Let l_1 and l_2 be two positive integers, which are polynomially bounded functions in security parameter λ. Let $\mathsf{PRF} = \{F_s\}_{s \in S}$ be a family of functions indexed by s, and $F_s : \{0,1\}^{l_1} \rightarrow \{0,1\}^{l_2}$. Function $F_s(\cdot)$ can also be represented by $F(s, \cdot)$. Let Γ_{l_1,l_2} be the set of all functions from $\{0,1\}^{l_1}$ to $\{0,1\}^{l_2}$.

Definition 1 *[12] Given an adversary $\mathcal{A}$ which has oracle access to a function in Γ_{l_1,l_2}, suppose that $\mathcal{A}$ always outputs a bit. The advantage of $\mathcal{A}$ over PRF is defined as*

$$\mathsf{Adv}^{\mathcal{A}}_{\mathsf{PRF}}(1^\lambda) = \left| Pr\left[\mathcal{A}^{F_s}(1^\lambda) = 1 \mid s \leftarrow S\right] - Pr\left[\mathcal{A}^{f}(1^\lambda) = 1 \mid f \leftarrow \Gamma_{l_1,l_2}\right] \right|.$$

Define the advantage of PRF as $\mathsf{Adv}_{\mathsf{PRF}}(\lambda) = \max_{PPT \mathcal{A}} \mathsf{Adv}^{\mathcal{A}}_{\mathsf{PRF}}(1^\lambda)$.
PRF is called pseudo-random if $\mathsf{Adv}_{\mathsf{PRF}}(\lambda)$ is negligible.

Obviously, the output of a pseudo-random function is a pseudo-random sequence, which is indistinguishable from a real random sequence to any PPT adversary.

2.2 PoDP/PoR Scheme

A PoDP/PoR scheme consists of seven PPT algorithms KeyGen, Encode, Decode, Challenge, Response, Verification and an audit algorithm Audit, where the algorithms Challenge, Response and Verification constitute an interaction protocol. A PoDP/PoR system associates a data storage service provider($\mathcal{P}$) with clients($\mathcal{V}$). The algorithms in a PoDP/PoR scheme works as follows.

Key generation: $(sk, pp) \leftarrow \mathsf{KeyGen}(1^\lambda)$**.** Each client $\mathcal{V}$ calls $(sk, pp) \leftarrow \mathsf{KeyGen}(1^\lambda)$ to get his own private key sk and a public parameter pp. $\mathcal{V}$ keeps the private key sk secret and stores (sk, pp) locally.

Storage encoding: $M^* \leftarrow \mathsf{Encode}(sk, pp, M)$**.** When a client $\mathcal{V}$ prepares a file M for outsourced storage, he/she will call $M^* \leftarrow \mathsf{Encode}(sk, pp, M)$ to encode M to M^*. The client $\mathcal{V}$ submits M^* to the storage provider.

Storage recovery: $\{M, \perp\} \leftarrow \mathsf{Decode}(sk, pp, M^*)$**.** When a client $\mathcal{V}$ gets back the out-sourced file M^*, he/she will call $\{M, \perp\} \leftarrow \mathsf{Decode}(sk, pp, M^*)$ to recover the original file M or $\perp$ indicating that M^* is too corrupted to recover M.

Storage. The storage provider $\mathcal{P}$ stores M^* submitted by the client $\mathcal{V}$.

Audit: $\beta \leftarrow$ **Audit** $(u,$ **Interaction**$(\mathcal{P}(M^*, pp) \leftrightharpoons \mathcal{V}(sk, pp)))$**.** A client $\mathcal{V}$ can run the interactive protocol $\mathsf{Interaction}(\mathcal{P}(M^*, pp) \leftrightharpoons \mathcal{V}(sk, pp))$ with the storage provider $\mathcal{P}$ for u times, which is polynomial in λ. The storage provider $\mathcal{P}$ possesses M^* and the client $\mathcal{V}$ keeps secret key sk. The protocol consists of the following three algorithms.

$\mathsf{Interaction}\ (\mathcal{P}(M^*, pp) \leftrightharpoons \mathcal{V}(sk, pp))$

$ch \leftarrow$ **Challenge**(1^λ)**:** This is a PPT algorithm run by the verifier $\mathcal{V}$. On input the security parameter λ, this algorithm outputs a challenge ch.

$re \leftarrow$ **Response**(pp, M^*, ch)**:** This is a deterministic algorithm run by the prover $\mathcal{P}$. On input the public parameter pp, the encoded file M^* and the challenge ch, the algorithm returns a corresponding response re.

$b \leftarrow$ **Verification**(pp, sk, ch, re)**:** This is a deterministic algorithm run by the verifier $\mathcal{V}$. On input the public parameter pp, the secret key sk, the challenge/response pair (ch, re), the algorithm returns a bit $b \in \{0, 1\}$, indicating acceptance with 1 or rejection with 0.

If $b = 0$, the interaction protocol outputs $\perp$; otherwise the protocol outputs (ch, re), and we call (ch, re) a consistent pair and the execution of the protocol successful.

After u executions of the interaction protocol in the audit, the client will output a bit β according to some audit strategy. If $\beta = 1$, the audit succeeds and the client will be convinced that the storage provider is still storing the file M^*. If $\beta = 0$, the audit fails, and the client will consider that his/her file M^* has been corrupted or lost.

Correctness of a PoDP/PoR scheme requires that for all (sk, pp, M^*) such that $(sk, pp) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and $M^* \leftarrow \mathsf{Encode}(sk, pp, M)$, the following two statements hold with probability 1.

- The original file M is always recovered from the correctly encoded file M^*, i.e., $M = \mathsf{Decode}(sk, pp, M^*)$ if $M^* \leftarrow \mathsf{Encode}(sk, pp, M)$.
- For honest $\mathcal{P}, \mathcal{V}$, algorithm $\mathsf{Verification}(pp, sk, ch, re)$ always outputs 1 in the interaction protocol, hence the interaction protocol outputs a challenge/response, i.e, $(ch, re) \leftarrow \mathsf{Interaction}(\mathcal{P}(M^*, pp) \leftrightharpoons \mathcal{V}(sk, pp))$.

Security of a PoDP/PoR scheme is considered in the following aspects.

Unforgeability for PoDP. We consider the probability ζ of a successful interaction in which a storage provider uses a deleted version of M^* to reply a client's challenges. This probability ζ will play an important role in the auditing strategy: An audit is claimed to be successful only if the number of the successful interactions among the total u interactions is larger than $\zeta \cdot u$. We will define and evaluate this probability in an unforgeability game. This probability is related to the deleted amount of M^*.

Soundness for PoR. Each successful execution of $\mathsf{Interaction}$ protocol brings some confidence to the client that his/her file is still stored by the storage provider. Successful executions of $\mathsf{Interaction}$ protocol provide consistent challenge/response (ch_i, re_i) pairs and each pair may provide some information about the original M. The soundness of the PoR scheme suggests that the original file M can be extracted from those consistent challenge/response (ch_i, re_i) pairs collected from successful interactions as long as the number (polynomial in λ) of executions of $\mathsf{Interaction}$ protocol is big enough.

2.3 Unforgeability of PoDP Scheme

A malicious storage provider may delete those data that is rarely accessed by clients to save storage space, or suffer from data loss due to outside attacks. However, the storage provider may still try to convince the client its storage of the file M^*, even though a file M^* is partially missing. Unforgeability of a PoDP scheme requires that the probability of a successful interaction should be low if some part of M^* is deleted. Unforgeability explains how much confidence a successful $\mathsf{Interaction}$ protocol gives a client.

It suggests how to set an audit strategy such that the malicious server cannot save on space by deleting a portion of M^* and still pass an audit. This probability is related to the proportion of the deleted to the whole file.

Given a PoDP scheme ($\mathsf{KeyGen}, \mathsf{Encode}, \mathsf{Decode}, \mathsf{Challenge}, \mathsf{Response}, \mathsf{Verification}, \mathsf{Audit}$), we define the following unforgeability game $\mathsf{Unforge}_{\mathcal{A}}^{PoDP}(\lambda)$ between an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and a challenger $\mathcal{V}$.

$\mathsf{Unforge}_{\mathcal{A}}^{PoDP}(\lambda)$

1. The challenger $\mathcal{V}$ obtains a secret key and a public parameter by $(sk, pp) \leftarrow \mathsf{KeyGen}(1^\lambda)$, and sends pp to the adversary $\mathcal{A}_1$.
2. $\mathcal{A}_1$ chooses a message M from the message space $\mathcal{M}$, and sends M to $\mathcal{V}$.
3. $\mathcal{V}$ encodes M to M^* with $M^* \leftarrow \mathsf{Encode}(sk, pp, M)$, and sends M^* to $\mathcal{A}_1$.
4. $\mathcal{A}_1$ deletes δ fraction of M^* to get $\tilde{M}^*$. The adversary $\mathcal{A}_2$ is given $(pp, \tilde{M}^*)$.
5. Test stage. $\mathcal{V}$ launches an $\mathsf{Interaction}$ protocol by sending $\mathcal{A}_2$ a challenge ch, with $ch \leftarrow \mathsf{Challenge}(1^\lambda)$. $\mathcal{A}_2$ computes a response $re' \leftarrow \mathcal{A}_2(pp, \tilde{M}^*, ch)$.
   $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ wins if $\mathsf{Verification}(pp, sk, ch, re') = 1$, i.e., this $\mathsf{Interaction}$ protocol is successful.

Definition 2 *A PoDP scheme is (δ, ζ)-unforgeable, if any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ wins in the above game $\mathsf{Unforge}_{\mathcal{A}}^{PoDP}(\lambda)$ with probability at most ζ. In formula,*

$$Pr\left[\mathsf{Verification}(pp, sk, ch, re') = 1 \;\middle|\; \begin{array}{l}(sk, pp) \leftarrow \mathsf{KeyGen}(1^\lambda), M \leftarrow \mathcal{A}_1(pp), \\ M^* \leftarrow \mathsf{Encode}(sk, pp, M), \tilde{M}^* \leftarrow \mathcal{A}_1(pp, M, M^*), \\ ch \leftarrow \mathsf{Challenge}(1^\lambda), re' \leftarrow \mathcal{A}_2(pp, \tilde{M}^*, ch)\end{array}\right] \leq \zeta,$$

where $\tilde{M}^*$ is a δ fraction-deleted version of M^*, and $\mathcal{A}_1, \mathcal{A}_2$ share neither coins nor state.

2.4 Soundness of PoR Scheme

Even a truthful storage provider may malfunction in an audit, due to temporary system failure, and that does not imply that the provider does not possess the audited data. To be more robust, it is desirable that the PoR system supports an ϵ-admissible storage prover, where the storage prover convincingly answers an ϵ fraction of the challenges in an audit. We want to prove that it is still possible to distill the original file M, as long as ϵ is larger than some threshold and the number of executions of $\mathsf{Interaction}$ protocol in the audit is big enough. This can be characterized by the *soundness of Proof of Retrievability (PoR)*.

A (cheating) prover $\tilde{\mathcal{P}}_\epsilon$ is *ϵ-admissible*, if it convincingly answers a challenge with probability at least ϵ, i.e., $\Pr\left[\mathsf{Verification}(pp, sk, ch, re') = 1 \;\middle|\; ch \leftarrow \mathsf{Challenge}(1^\lambda), re' \leftarrow \tilde{\mathcal{P}}_\epsilon(pp, ch)\right] \geq \epsilon$. The PoR system is ϵ-sound if there exists a PPT algorithm $\mathsf{Extractor}$, which recovers M on input the public parameter pp, the secret key sk and the output of the interactive protocol $\mathsf{Interaction}(\tilde{\mathcal{P}}_\epsilon(pp, \cdot) \leftrightharpoons \mathcal{V}(sk, pp))$, i.e., $\mathsf{Extractor}\big(pp, sk, \mathsf{Interaction}(\tilde{\mathcal{P}}_\epsilon(pp, \cdot) \leftrightharpoons \mathcal{V}(sk, pp))\big) = M$, with overwhelming probability.

The formal definition of soundness of PoR scheme has been presented in [7, 14]. Here we give a refined description.

Given a PoR scheme ($\mathsf{KeyGen}, \mathsf{Encode}, \mathsf{Decode}, \mathsf{Challenge}, \mathsf{Response}, \mathsf{Verification}, \mathsf{Audit}$), we define the following soundness game $\mathsf{Sound}_{\mathcal{A}}^{PoR}(\lambda)$ between an adversary $\mathcal{A}$ and a challenger $\mathcal{V}$. $\mathcal{A}$ is going to create an adversarial prover $\tilde{\mathcal{P}}_\epsilon$, and a PPT $\mathsf{Extractor}$ aims to extract the original file from interactions with $\tilde{\mathcal{P}}_\epsilon$.

$\mathsf{Sound}_{\mathcal{A}}^{PoR}(\lambda)$

1. The challenger $\mathcal{V}$ obtains a secret key and a public parameter by $(sk, pp) \leftarrow \mathsf{KeyGen}(1^\lambda)$, and sends pp to the adversary $\mathcal{A}$.
2. $\mathcal{A}$ chooses a message M from the message space $\mathcal{M}$, and sends M to $\mathcal{V}$.
3. $\mathcal{V}$ encodes M to M^* with $M^* \leftarrow \mathsf{Encode}(sk, pp, M)$, and sends M^* to $\mathcal{A}$.

4. Test stage. The adversary $\mathcal{A}$ gets protocol access to $\mathcal{V}(sk, pp)$ and can interact with $\mathcal{V}$ for a polynomial times. Then the adversary $\mathcal{A}$ outputs an ϵ-admissible prover $\tilde{\mathcal{P}}_\epsilon$. This ϵ-admissible prover $\tilde{\mathcal{P}}_\epsilon$ functions as an oracle, which will output a consistent re for a proper query (challenge) ch with probability at least ϵ.

5. A PPT Extractor is given the public parameter pp, the secret key sk and oracle access to the ϵ-admissible prover $\tilde{\mathcal{P}}_\epsilon$. After querying $\tilde{\mathcal{P}}_\epsilon$ for a polynomial times, the Extractor outputs a file M'.

Definition 3 *[7] A PoR scheme is ϵ-sound, if there exists a PPT* Extractor*, such that for any PPT adversary $\mathcal{A}$ which outputs an ϵ-admissible $\tilde{\mathcal{P}}_\epsilon$ through interacting with $\mathcal{V}$ in the above game $\mathsf{Sound}_{\mathcal{A}}^{PoR}(\lambda)$,* Extractor *is able to recover the original file M by querying $\tilde{\mathcal{P}}_\epsilon$ except with negligible probability. In formula,*

$$Pr\left[M' \neq M \;\middle|\; \begin{array}{l}(sk, pp) \leftarrow \mathsf{KeyGen}(1^\lambda), M \leftarrow \mathcal{A}(pp), M^* \leftarrow \mathsf{Encode}(sk, pp, M), \\ \tilde{\mathcal{P}}_\epsilon \leftarrow \mathcal{A}\left(pp, M, M^*, access\ to\ \mathcal{V}(sk, pp)\right), M' \leftarrow \mathsf{Extractor}(pp, sk, \tilde{\mathcal{P}}_\epsilon)\end{array}\right]$$

is negligible.

3 Maximum Rank Distance Codes and Gabidulin Codes

Rank distance was first considered for error-correcting codes by Delsarte [4], and lots of work has been devoted to rank distance properties, code construction, and efficient decoding. In [4, 3, 2], a Singleton bound on the minimum rank distance of codes was established, and a class of codes achieving the bound with equality was constructed, e.g. MRD codes. Gabidulin codes is one of MRD codes, which can be considered as the rank metric analogues of Reed-Solomon codes. In [15], MRD codes was used in authentication.

3.1 Rank Distance Codes

Let $\mathbb{F}_q$ be a finite field, and $\mathbb{F}_{q^t}$ be an extended field of degree t. Let $(\mathbb{F}_{q^t})^n$ be the n-dimensional vector space over $\mathbb{F}_{q^t}$. Given a vector $\vec{\mathsf{a}} = (\mathsf{a}_1, \mathsf{a}_2, \ldots, \mathsf{a}_n) \in (\mathbb{F}_{q^t})^n$, each entry $\mathsf{a}_i \in \mathbb{F}_{q^t}$ can be expressed as a t-dimensional column vector over $\mathbb{F}_q$, i.e., $\mathsf{a}_i = \left(a_i^{(1)}, a_i^{(2)}, \cdots, a_i^{(t)}\right)^T$. Consequently, vector $\vec{\mathsf{a}}$ can be expressed as a $t \times n$ matrix over $\mathbb{F}_q$ by expanding all the entries of $\vec{\mathsf{a}}$.

Definition 4 *[1] A rank distance code $\mathcal{C}_R$ over finite field $\mathbb{F}_{q^t}$ is a subspace of $(\mathbb{F}_{q^t})^n$, satisfying the following properties.*

(a) *For each codeword $\vec{\mathsf{c}} \in \mathcal{C}_R$, the rank weight (over $\mathbb{F}_q$) of $\vec{\mathsf{c}}$, denoted as $rank(\vec{\mathsf{c}}|\mathbb{F}_q)$, is defined as the rank of the corresponding $t \times n$ matrix over $\mathbb{F}_q$ when $\vec{\mathsf{c}}$ is expressed as a matrix over $\mathbb{F}_q$.*

(b) *For any two codewords $\vec{\mathsf{a}}, \vec{\mathsf{b}} \in \mathcal{C}_R$, the rank distance (over $\mathbb{F}_q$) of $\vec{\mathsf{a}}$ and $\vec{\mathsf{b}}$ is defined as $d_R(\vec{\mathsf{a}}, \vec{\mathsf{b}}) = rank(\vec{\mathsf{a}} - \vec{\mathsf{b}}|\mathbb{F}_q)$.*

(c) *The minimum rank distance of the code $\mathcal{C}_R$, denoted as $d_R(\mathcal{C}_R)$, is the minimum rank distance of all possible pairs of distinct codewords in $\mathcal{C}_R$, i.e., $d_R(\mathcal{C}_R) = \min\limits_{\substack{\vec{\mathsf{a}}, \vec{\mathsf{b}} \in \mathcal{C}_R \\ \vec{\mathsf{a}} \neq \vec{\mathsf{b}}}} d_R(\vec{\mathsf{a}}, \vec{\mathsf{b}})$.*

We call $\mathcal{C}_R$ a $[q, c, n, t, d]$ rank distance code, if each codeword consists of n elements in $\mathbb{F}_{q^t}$, $c = |\mathcal{C}_R|$ is the number of codewords, $d = d_R(\mathcal{C}_R)$ is the minimum rank distance.

3.2 Maximum Rank Distance Codes and Gabidulin Code

For a $[q, c, n, t, d]$ rank distance code $\mathcal{C}_R$ with $t \geq n$, it was shown that $d_R \leq d_H$ in [3], where d_H denotes the minimum Hamming distance of $\mathcal{C}_R$. With the Singleton Bound for block codes, we have the minimum rank distance of $\mathcal{C}_R$ satisfies $d = d_R \leq n - k + 1$, where $k = \log_{q^t} c$.

Definition 5 *A $[q, c, n, t, d]$ rank distance code $\mathcal{C}_R$ with $t \geq n$ is called the maximum-rank-distance code(MRD code), if $\mathcal{C}_R$ achieves the bound $d = n - k + 1$, where $k = \log_{q^t} c$.*

Definition 6 *A $[q, k, n, t]$ Gabidulin code $\mathcal{C}_R$ with $t \geq n$ consists of a generation matrix $G_{k\times n}$, a parity check matrix $H_{n\times(n-k)}$ over $\mathbb{F}_{q^t}$ and two algorithms (GabiEncode, GabiDecode).*

Generation Matrix $\mathbf{G}_{k\times n}$/Parity Check Matrix $\mathbf{H}_{n\times(n-k)}$. *The generation matrix $G_{k\times n}$ is determined by n elements $(g_1, g_2, \ldots, g_n)$, with $g_i \in \mathbb{F}_{q^t}$, $i = 1, 2, \cdots, n$, and all the n elements are linearly independent over $\mathbb{F}_q$. The parity check matrix $H_{n\times(n-k)}$ is determined by the generation matrix $G_{k\times n}$ satisfying $G_{k\times n} \cdot H_{n\times(n-k)} = 0$. In [17], a fast algorithm was shown to find $h_i \in \mathbb{F}_{q^t}$, $i = 1, 2, \cdots, n$ such that*

$$G_{k\times n} = \begin{pmatrix} g_1 & g_2 & \cdots & g_n \\ g_1^q & g_2^q & \cdots & g_n^q \\ \vdots & \vdots & & \vdots \\ g_1^{q^{k-1}} & g_2^{q^{k-1}} & \cdots & g_n^{q^{k-1}} \end{pmatrix}, \quad H_{n\times(n-k)} = \begin{pmatrix} h_1 & h_1^q & \cdots & h_1^{q^{n-k-1}} \\ h_2 & h_2^q & \cdots & h_2^{q^{n-k-1}} \\ \vdots & \vdots & & \vdots \\ h_n & h_n^q & \cdots & h_n^{q^{n-k-1}} \end{pmatrix}. \quad (1)$$

GabiEncode($\vec{\mathsf{m}}, \mathbf{G}_{k\times n}$). *Taking as input the generation matrix $G_{k\times n}$ and the message vector $\vec{m} = (m_1, m_2, \cdots, m_k) \in (\mathbb{F}_{q^t})^k$, it computes $\vec{c} = \vec{m} \cdot G_{k\times n}$, and outputs the codeword $\vec{c} = (c_1, c_2, \cdots, c_n)$.*

GabiDecode($\vec{\mathsf{r}}, \mathbf{H}_{n\times(n-k)}$). *Taking as input the parity check matrix $H_{n\times(n-k)}$ and a vector $\vec{r} = (r_1, r_2, \cdots, r_n) \in (\mathbb{F}_{q^t})^n$, it outputs $\vec{m} = (m_1, m_2, \cdots, m_k)$. Refer to [9] for the specific decoding algorithm.*

A $[q, k, n, t]$ Gabidulin code $\mathcal{C}_R$ is a $[q, q^{tk}, n, t, n-k+1]$ MRD code and $\mathcal{C}_R = \{\vec{\mathsf{c}} \mid \vec{\mathsf{c}} = \vec{\mathsf{m}} \cdot \mathsf{G}_{k\times n}, \vec{\mathsf{m}} = (\mathsf{m}_1, \mathsf{m}_2, \cdots, \mathsf{m}_k), \mathsf{m}_i \in \mathbb{F}_{q^t}, i = 1, 2, \cdots, k\} \subseteq (\mathbb{F}_{q^t})^n$.

The *correctness* of Gabidulin Code requires that $\vec{\mathsf{m}} = \mathbf{GabiDecode}(\vec{\mathsf{c}}, \mathsf{H}_{n\times(n-k)})$ if $\vec{\mathsf{c}} \leftarrow \mathbf{GabiEncode}(\vec{\mathsf{m}}, \mathsf{G}_{k\times n})$ for all $\vec{\mathsf{m}} \in (\mathbb{F}_{q^t})^k$.

Lots of work has been done on the efficient decoding of Gabidulin and other MRD Codes. Decoding algorithms have been proposed to decode erasures or errors or both of them simultaneously.

A Gabidulin codeword $\vec{\mathsf{c}}$ satisfies $\vec{\mathsf{c}} \cdot \mathsf{H}_{n\times(n-k)} = \mathsf{0}$. Let $\mathsf{E}_{t\times n}$ be the noise matrix over $\mathbb{F}_q$, which is added to the codeword. There are three types of noise matrix.

- Row erasures. Let x denote the number of row erasures. Row erasures means the locations of erroneous rows are known.
- Column erasures. Let y denote the number of column erasures. Column erasures means the locations of erroneous columns are known.
- Unknown errors. Let z be the rank of error matrix $\mathsf{E}_{t\times n}$, where the error locations are unknown.

Efficient (PPT) decoding algorithms $\vec{\mathsf{m}} \leftarrow \mathsf{GabiDecode}(\vec{\mathsf{r}}, \mathsf{H}_{n\times(n-k)})$ exist for erasure and error-correction decoding [9], which correct x-fold row erasures, y-fold column erasures, and z-fold rank errors simultaneously, provided that $x + y + 2z \leq d - 1$, where $d = n - k + 1$.

4 PoDP/PoR Scheme from MRD Codes

Throughout the paper, we will use Gabidulin Code as an instantiation of MRD code. The only reason to use Gabidulin Code instead of a general MRD code is that Gabidulin Code has explicit generation matrix and parity check matrix, which helps us to analyze the performance of our proposal.

The primitives involved in the PoDP/PoR scheme are

- a Pseudo-Random Function $\mathsf{PRF} : \mathcal{K}_{prf} \times \{0,1\}^* \to \mathbb{F}_{q^t}$, which uses a seed k', randomly chosen from space $\mathcal{K}_{prf}$, to generate pseudo-random keys with $\mathsf{PRF}(k', i)$ for $i = 1, 2, \cdots$. The length of the seed k' is determined by the security parameter λ.
- A $[q, k, n, t]$ Gabidulin code $\mathcal{C}_R$ with $t \geq n$, which is associated with ($\mathsf{G}_{k\times n}, \mathsf{H}_{n\times(n-k)}$, GabiEncode, GabiDecode) defined in Definition 6.

The PoDP/PoR scheme constructed from Gabidulin codes consists of the following algorithms.

- $(sk, pp) \leftarrow$ **KeyGen**(1^λ)**:** The key generation algorithm takes as input the security parameter λ, chooses random elements $\mathsf{k} \in \mathbb{F}_{q^n}$ and $k' \in \mathcal{K}_{prf}$. Choose a Pseudo-Random Function (PRF) $\mathsf{PRF} : \mathcal{K}_{prf} \times \{0,1\}^* \to \mathbb{F}_{q^t}$ and a $[q, k, n, t]$ Gabidulin code $\mathcal{C}_R$. Output the public parameter $pp = (\mathsf{PRF}, \mathcal{C}_R)$ and the secret key $sk = (\mathsf{k}, k')$.

- $M^* \leftarrow$ **Encode**(sk, pp, M)**:** The storage encoding algorithm takes as input the secret key sk, the public parameter pp and the original file M. It will compute an encoded file M^* as follows.

  (1) Block Division: The original file M is divided into w blocks, denoted by $(M_1, M_2, \cdots, M_w)^{\mathsf{T}}$, each block $M_i = (\mathsf{m}_{i1}, \mathsf{m}_{i2}, \cdots, \mathsf{m}_{ik}) \in (\mathbb{F}_{q^t})^k$ consisting of k elements of $\mathbb{F}_{q^t}$. Express each $\mathsf{m}_{ij} \in \mathbb{F}_{q^t}$ as a column vector of dimension t over $\mathbb{F}_q$, i.e., $\mathsf{m}_{ij} = \left(m_{ij}^{(1)}, m_{ij}^{(2)}, \cdots, m_{ij}^{(t)}\right)^T$. Then each block M_i can be expressed as a $t \times k$ matrix over $\mathbb{F}_q$, i.e.

$$M_i = \begin{pmatrix} m_{i1}^{(1)} & m_{i2}^{(1)} & \cdots & m_{ik}^{(1)} \\ m_{i1}^{(2)} & m_{i2}^{(2)} & \cdots & m_{ik}^{(2)} \\ \vdots & \vdots & & \vdots \\ m_{i1}^{(t)} & m_{i2}^{(t)} & \cdots & m_{ik}^{(t)} \end{pmatrix}.$$

  (2) Gabidulin Encoding: The original file M is encoded into Gabidulin codewords block by block with $C_i \leftarrow$ **GabiEncode**$(M_i, \mathsf{G}_{k\times n})$, $i = 1, 2, \cdots, w$. More precisely, given block $M_i = (\mathsf{m}_{i1}, \mathsf{m}_{i2}, \cdots, \mathsf{m}_{ik}) \in (\mathbb{F}_{q^t})^k$, the codeword $C_i = (\mathsf{c}_{i1}, \mathsf{c}_{i2}, \cdots, \mathsf{c}_{in}) \in (\mathbb{F}_{q^t})^n$ is computed as $C_i = (\mathsf{c}_{i1}, \mathsf{c}_{i2}, \cdots, \mathsf{c}_{in}) = (\mathsf{m}_{i1}, \mathsf{m}_{i2}, \cdots, \mathsf{m}_{ik}) \cdot \mathsf{G}_{k\times n}$. It should be noted that the encoding is a matrix multiplication over finite field $\mathbb{F}_{q^t}$.

  Similarly, $\mathsf{c}_{ij} = \left(c_{ij}^{(1)}, c_{ij}^{(2)}, \cdots, c_{ij}^{(t)}\right)^T$, which is a vector over $\mathbb{F}_q$, and each codeword C_i can be expressed as a $t \times n$ matrix over $\mathbb{F}_q$, i.e.

$$C_i = \begin{pmatrix} c_{i1}^{(1)} & c_{i2}^{(1)} & \cdots & c_{in}^{(1)} \\ c_{i1}^{(2)} & c_{i2}^{(2)} & \cdots & c_{in}^{(2)} \\ \vdots & \vdots & & \vdots \\ c_{i1}^{(t)} & c_{i2}^{(t)} & \cdots & c_{in}^{(t)} \end{pmatrix}.$$

  (3) Tag Generation: Compute a tag $\boldsymbol{\sigma}_i$ for each codeword C_i with the secret key $sk = (\mathsf{k}, k')$.

  (i) Express $\mathsf{k} \in \mathbb{F}_{q^n}$ as a vector of dimension n over $\mathbb{F}_q$, i.e., $\mathsf{k} = \left(k^{(1)}, k^{(2)}, \cdots, k^{(n)}\right)^T$.

  (ii) Compute $\mathsf{k}_i = \mathsf{PRF}(k', i)$ for $i = 1, 2, \cdots, w$. Express $\mathsf{k}_i \in \mathbb{F}_{q^t}$ as a vector of dimension t over $\mathbb{F}_q$, i.e., $\mathsf{k}_i = \left(k_i^{(1)}, k_i^{(2)}, \cdots, k_i^{(t)}\right)^T$.

  (iii) For each codeword C_i, a tag $\boldsymbol{\sigma}_i \in \mathbb{F}_{q^t}$ is computed with

$$\boldsymbol{\sigma}_i = C_i \cdot \mathsf{k} + \mathsf{k}_i = \begin{pmatrix} c_{i1}^{(1)} & c_{i2}^{(1)} & \cdots & c_{in}^{(1)} \\ c_{i1}^{(2)} & c_{i2}^{(2)} & \cdots & c_{in}^{(2)} \\ \vdots & \vdots & & \vdots \\ c_{i1}^{(t)} & c_{i2}^{(t)} & \cdots & c_{in}^{(t)} \end{pmatrix} \cdot \begin{pmatrix} k^{(1)} \\ k^{(2)} \\ \\ \vdots \\ k^{(n)} \end{pmatrix} + \begin{pmatrix} k_i^{(1)} \\ k_i^{(2)} \\ \\ \vdots \\ k_i^{(t)} \end{pmatrix}, \tag{2}$$

  where all the operations are in the basic field $\mathbb{F}_q$.

  (4) Storage File M^* : Each block M_i is encoded as $(C_i || \boldsymbol{\sigma}_i) = (\mathsf{c}_{i1}, \mathsf{c}_{i2}, \cdots, \mathsf{c}_{in}, \boldsymbol{\sigma}_i)$. The final encoded file M^* consists of blocks attached with tags:

$$M^* = (C_1 || \boldsymbol{\sigma}_1, C_2 || \boldsymbol{\sigma}_2, \cdots, C_w || \boldsymbol{\sigma}_w)^{\mathsf{T}} = \begin{pmatrix} \mathsf{c}_{11} & \mathsf{c}_{12} & \cdots & \mathsf{c}_{1n} & \boldsymbol{\sigma}_1 \\ \mathsf{c}_{21} & \mathsf{c}_{22} & \cdots & \mathsf{c}_{2n} & \boldsymbol{\sigma}_2 \\ \vdots & \vdots & & \vdots & \vdots \\ \mathsf{c}_{w1} & \mathsf{c}_{w2} & \cdots & \mathsf{c}_{wn} & \boldsymbol{\sigma}_w \end{pmatrix},$$

  where each entry in the matrix is an element from $\mathbb{F}_{q^t}$.

- $\{M, \perp\} \leftarrow \textbf{Decode}(sk, pp, M^*)$**:** Parse M^* as $M^* = (C_1'||\boldsymbol{\sigma}_1', C_2'||\boldsymbol{\sigma}_2', \cdots, C_w'||\boldsymbol{\sigma}_w')^{\mathsf{T}}$.

  for $i = 1, 2, \cdots, w$ **do**

  If $C_i' \cdot \mathsf{H}_{n\times(n-k)} \neq 0$ return "$\perp$".

  If $\boldsymbol{\sigma}_i' \neq C_i' \cdot \mathsf{k} + \mathsf{k}_i$ return "$\perp$".

  $M_i' \leftarrow \mathsf{GabiDecode}(C_i', \mathsf{H}_{n\times(n-k)})$.

  end for

  Return $M = (M_1', M_2', \cdots, M_w')^{\mathsf{T}}$.

- $\{(ch, re), \perp\} \leftarrow \textbf{Interaction}(\mathcal{P}(M^*, pp) \leftrightharpoons \mathcal{V}(sk, pp))$**:** The interactive protocol consists of three algorithms.

  ♦ $ch \leftarrow \textbf{Challenge}(1^\lambda)$**:** The challenge $ch = (v_1, v_2, \cdots, v_w)$ is chosen uniformly at random from $(\mathbb{F}_q)^w$.

  ♦ $re \leftarrow \textbf{Response}(pp, M^*, ch)$**: (i)** First the encoded file M^* is segmented into w blocks. Let $M^* = (C_1||\boldsymbol{\sigma}_1, C_2||\boldsymbol{\sigma}_2, \cdots, C_w||\boldsymbol{\sigma}_w)^{\mathsf{T}}$.

  (ii) Then the response re is computed with aggregation.

$$re = ch \circ M^* = (v_1, v_2, \cdots, v_w) \circ \begin{pmatrix} \mathsf{c}_{11} & \cdots & \mathsf{c}_{1n} & \boldsymbol{\sigma}_1 \\ \mathsf{c}_{21} & \cdots & \mathsf{c}_{2n} & \boldsymbol{\sigma}_2 \\ \vdots & & \vdots & \vdots \\ \mathsf{c}_{w1} & \cdots & \mathsf{c}_{wn} & \boldsymbol{\sigma}_w \end{pmatrix}$$
$$= \left(\sum_{i=1}^{w} v_i \circ \mathsf{c}_{i1}, \cdots, \sum_{i=1}^{w} v_i \circ \mathsf{c}_{in}, \sum_{i=1}^{w} v_i \circ \boldsymbol{\sigma}_i \right),$$

  where "$\circ$" is scalar multiplication over vector space on $\mathbb{F}_q$. More precisely, express $\mathsf{c}_{ij} = \left(c_{ij}^{(1)}, c_{ij}^{(2)}, \cdots, c_{ij}^{(t)}\right)^T$ as a t-dimentional vector over $\mathbb{F}_q$. Then $v_i \circ \mathsf{c}_{ij} = \left(v_i \cdot c_{ij}^{(1)}, v_i \cdot c_{ij}^{(2)}, \cdots, v_i \cdot c_{ij}^{(t)}\right)^T$.

  ♦ $b \leftarrow \textbf{Verification}(pp, sk, ch, re)$**:** The input is given by the public parameter $pp = (\mathsf{PRF}, \mathcal{C}_R)$, where $\mathcal{C}_R$ is associated with the generation matrix $\mathsf{G}_{k\times n}$ and parity check matrix $\mathsf{H}_{n\times(n-k)}$, the secret key $sk = (\mathsf{k}, k') \in \mathbb{F}_{q^n} \times \mathcal{K}_{prf}$, the challenge $ch = (v_1, v_2, \cdots, v_w) \in (\mathbb{F}_q)^w$, and response $re = (\bar{\mathsf{c}}_1, \bar{\mathsf{c}}_2, \cdots, \bar{\mathsf{c}}_n, \bar{\boldsymbol{\sigma}}) \in (\mathbb{F}_{q^t})^{n+1}$.

  (i) Parse re as $(\bar{C}, \bar{\boldsymbol{\sigma}})$, where $\bar{C} = (\bar{\mathsf{c}}_1, \bar{\mathsf{c}}_2, \cdots, \bar{\mathsf{c}}_n)$. Check whether $\bar{C}$ is an MRD codeword by testing $\bar{C} \cdot \mathsf{H}_{n\times(n-k)} = 0$. If not, output $b = 0$.

  (ii) Express $\mathsf{k} \in \mathbb{F}_{q^n}$ as a vector of dimension n over $\mathbb{F}_q$, i.e., $\mathsf{k} = \left(k^{(1)}, k^{(2)}, \cdots, k^{(n)}\right)^T$. Compute $\mathsf{k}_i = \mathsf{PRF}(k', i) \in \mathbb{F}_{q^t}$ for $i = 1, 2, \cdots, w$. Express k_i as a vector of dimension t over $\mathbb{F}_q$, i.e., $\mathsf{k}_i = \left(k_i^{(1)}, k_i^{(2)}, \cdots, k_i^{(t)}\right)^T$.

  (iii) Express $\bar{C}$ as a matrix over $\mathbb{F}_q$, i.e., $\bar{C} = \begin{pmatrix} \bar{c}_1^{(1)} & \bar{c}_2^{(1)} & \cdots & \bar{c}_n^{(1)} \\ \bar{c}_1^{(2)} & \bar{c}_2^{(2)} & \cdots & \bar{c}_n^{(2)} \\ \vdots & \vdots & & \vdots \\ \bar{c}_1^{(t)} & \bar{c}_2^{(t)} & \cdots & \bar{c}_n^{(t)} \end{pmatrix}$ and $\bar{\boldsymbol{\sigma}} = \begin{pmatrix} \bar{\sigma}^{(1)} \\ \bar{\sigma}^{(2)} \\ \vdots \\ \bar{\sigma}^{(t)} \end{pmatrix}$.

  (iv) Check

$$\bar{\boldsymbol{\sigma}} = \begin{pmatrix} \bar{c}_1^{(1)} & \bar{c}_2^{(1)} & \cdots & \bar{c}_n^{(1)} \\ \bar{c}_1^{(2)} & \bar{c}_2^{(2)} & \cdots & \bar{c}_n^{(2)} \\ \vdots & \vdots & & \vdots \\ \bar{c}_1^{(t)} & \bar{c}_2^{(t)} & \cdots & \bar{c}_n^{(t)} \end{pmatrix} \cdot \begin{pmatrix} k^{(1)} \\ k^{(2)} \\ \vdots \\ k^{(n)} \end{pmatrix} + \begin{pmatrix} \sum_{i=1}^{w} v_i \cdot k_i^{(1)} \\ \sum_{i=1}^{w} v_i \cdot k_i^{(2)} \\ \vdots \\ \sum_{i=1}^{w} v_i \cdot k_i^{(t)} \end{pmatrix}, \tag{3}$$

  where all the operations are in the field $\mathbb{F}_q$. If Eq.(3) holds, return 1, otherwise return 0.

- **{0,1}$\leftarrow$ Audit.** The client $\mathcal{V}$ executes the $\mathsf{Interaction}$ protocol u times.

  $b = 0$;

  for $i = 1$ to u **do** $\mathsf{Interaction}(\mathcal{P}(M^*, pp) \leftrightharpoons \mathcal{V}(sk, pp))$

  $ch \leftarrow \mathsf{Challenge}(1^\lambda)$;

  $re \leftarrow \mathsf{Response}(pp, M^*, ch)$;

  If $\mathsf{Verification}(pp, ch, re) = 1$ then $b \leftarrow b + 1$;

  end for

  If $(b/u > \frac{1}{q})$ return 1, otherwise return 0.

In the audit, $b/u > \frac{1}{q}$ means $b/u - \frac{1}{q}$ is non-negligible, since u is polynomial in λ.

The *Correctness* of the above PoDP/PoR scheme is guaranteed by the following facts. For all (sk, pp, M^*) such that $(sk, pp) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and $M^* \leftarrow \mathsf{Encode}(sk, pp, M)$,

- the original file M is always recovered from the correctly encoded file M^*, due to the correctness of the encoding/decoding algorithms of Gabidulin codes;
- the interaction protocol between the honest $\mathcal{P}, \mathcal{V}$ results in $1 \leftarrow \mathsf{Verification}(pp, sk, ch, re)$, hence the interaction protocol outputs a challenge/response, i.e, $(ch, re) \leftarrow \mathsf{Interaction}(\mathcal{P}(M^*, pp) \leftrightharpoons \mathcal{V}(sk, pp))$, due to the following facts:

$$\sum_{i=1}^{w} v_i \circ \boldsymbol{\sigma}_i = (v_1, v_2, \cdots, v_w) \circ \begin{pmatrix} \boldsymbol{\sigma}_1 \\ \boldsymbol{\sigma}_2 \\ \vdots \\ \boldsymbol{\sigma}_w \end{pmatrix} = (v_1, v_2, \cdots, v_w) \circ \left(\begin{pmatrix} C_1 \\ C_2 \\ \vdots \\ C_w \end{pmatrix} \cdot \mathsf{k} + \begin{pmatrix} \mathsf{k}_1 \\ \mathsf{k}_2 \\ \vdots \\ \mathsf{k}_w \end{pmatrix} \right)$$

$$= \left(\sum_{i=1}^{w} v_i \circ \mathsf{c}_{i1}, \sum_{i=1}^{w} v_i \circ \mathsf{c}_{i2}, \cdots, \sum_{i=1}^{w} v_i \circ \mathsf{c}_{in} \right) \cdot \mathsf{k} + \sum_{i=1}^{w} v_i \circ \mathsf{k}_i = (\bar{\mathsf{c}}_1, \bar{\mathsf{c}}_2, \cdots, \bar{\mathsf{c}}_n) \cdot \mathsf{k} + \sum_{i=1}^{w} v_i \circ \mathsf{k}_i,$$

  where $\bar{\mathsf{c}}_j = \left(\bar{c}_j^{(1)}, \bar{c}_j^{(2)}, \cdots, \bar{c}_j^{(t)}\right)^T$, $\mathsf{k} = \left(k^{(1)}, k^{(2)}, \cdots, k^{(n)}\right)^T$ and $\mathsf{k}_i = \left(k_i^{(1)}, k_i^{(2)}, \cdots, k_i^{(t)}\right)^T$, which are vectors over $\mathbb{F}_q$.

5 Performance Analysis

To evaluate the efficiency of the proposed PoDP/PoR scheme, we will analyze the local storage of the client, the expansion rate θ of the storage, which is defined as the ratio of the length of encoded file M^* to that of the original file M, the length of the challenge, denoted by $|ch|$, the length of the response, denoted by $|re|$, the computational complexity of the storage provider to compute the response, and the computational complexity of the client to verify the response.

Let $\mathsf{Mul}_{\mathbb{F}_q}$ denote a multiplication and $\mathsf{Add}_{\mathbb{F}_q}$ denote an addition over finite field $\mathbb{F}_q$.

We will use a pseudo-random function $\mathsf{PRF} : \{0,1\}^\lambda \times \{0,1\}^* \to \mathbb{F}_{q^t}$ as an instantiation.

Local storage: The client will store the key $\mathsf{k} \in \mathbb{F}_{q^n}$, the seed $k' \in \{0,1\}^\lambda$ of PRF, and the parity check matrix $\mathsf{H}_{n\times(n-k)}$ of the $[q, k, n, t]$ Gabidulin code. The parity check matrix is determined by $\mathsf{h}_i \in \mathbb{F}_{q^t}$, $i = 1, 2, \cdots, n$ according to Eq.(1). Hence the local storage is $(n \log q + \lambda + nt \log q)$ bits.

Storage expansion rate θ: $\theta = \frac{|M^*|}{|M|} = \frac{n+1}{k}$.

Length of challenge: $ch \in (\mathbb{F}_q)^w$, hence $|ch| = w \log q$ bits.

Length of response: re consists of a Gabidulin codeword and a tag, hence $|re| = (n+1)t \log q$ bits.

Computational complexity of the storage server: To compute re, the storage server will do $(n+1)tw(q-1)/q$ multiplications and $(n+1)t(w-1)(q-1)/q$ additions over $\mathbb{F}_q$ on average. Hence there are totally $(n+1)tw(q-1)/q\mathsf{Mul}_{\mathbb{F}_q} + (n+1)t(w-1)(q-1)/q\mathsf{Add}_{\mathbb{F}_q}$ on average.

Computational complexity of the client: To test the consistency of (ch, re), the client needs $(n-k)n$ multiplications and $(n-k)(n-1)$ additions over $\mathbb{F}_{q^t}$ to test Gabidulin codeword, and another $tn + wt(q-1)/q$ multiplications and $t(n-1) + t(w-1)(q-1)/q$ additions over $\mathbb{F}_q$ on average. Hence there are totally $(n-k)n\mathsf{Mul}_{\mathbb{F}_{q^t}} + (n-k)(n-1)\mathsf{Add}_{\mathbb{F}_{q^t}} + (tn + wt(q-1)/q)\mathsf{Mul}_{\mathbb{F}_q} + (t(n-1) + t(w-1)(q-1)/q)\mathsf{Add}_{\mathbb{F}_q}$.

Now we instantiate the PoDP/PoR scheme with security parameter $\lambda = 80$, and a $[q, k, n, t]$ Gabidulin Code with $q = 2, k = 128, n = t = 256$ and $d = 129$. $q = 2$ means that the addition over the binary field is reduced to XOR.

We consider a file M of size $1GB$. The number of blocks is $w = 2^{18}$.

Local	Rate θ	$\lvert ch\rvert$	$\lvert re\rvert$	Server Comput.	Client Comput.
$\lvert sk\rvert$= 336 bits, $\lvert pp\rvert$=8KB	≈ 2	32KB	$\approx$ 8KB	$O(2^{33})$ bit-XOR	$O(2^{31})$ bit-XOR

Table 2: Instantiation of PoDP/PoR Scheme with [2, 128, 256, 256]-Gabidulin Code

During an interaction, the server will implement 2^{33} bit-XOR to prepare a response re. To verify the consistency of the response, the client's computation is dominated by $2^{15}\mathsf{Mul}_{\mathbb{F}_{2^{256}}}$. One $\mathsf{Mul}_{\mathbb{F}_{2^{256}}}$ needs $O(256 \cdot 256) = O(2^{16})$ bit operations. Hence the client's computation complexity is $O(2^{31})$.

Consider a modular exponentiation in RSA with a 1024-bit modulus. It will take 1024 modular squares and about 512 modular multiplications. One modular multiplication needs $O(1024 \cdot 1024) = O(2^{20})$ bit operations. So a modular exponentiation in RSA with a 1024-bit modulus has computation complexity of $O(2^{30})$.

Therefore, to audit the storage of $1GB$ file, the computational complexity of both client and server in an interaction is comparable to 2 and 8 modular exponentiations with a 1024-bit modulus respectively.

6 The Security of the Proposed PoDP/PoR Scheme

Remember that in the audit of the PoDP/PoR scheme, the audit strategy is determined by a threshold $\frac{1}{q}$. Only if ϵ fraction , $\epsilon > \frac{1}{q}$, of the u executions of $\mathsf{Interaction}$ protocol is successful, does the audit claim success.

The *Unforgeability* of the PoDP/PoR scheme will justify the necessity of the threshold $\frac{1}{q}$. We will prove that a malicious storage provider, who deletes at least one block of M^*, replies a response re that passes verification in an $\mathsf{Interaction}$ protocol with probability at most $\frac{1}{q}$.

On the other hand, the *ϵ-soundness* of the PoDP/PoR scheme justifies the sufficiency of the threshold $\frac{1}{q}$. We will prove that as long as a storage provider replies a response re that passes verification with probability $\epsilon > \frac{1}{q}$, there exists a PPT extractor recovering the original file M as long as the number of executions of $\mathsf{Interaction}$ protocol is big enough.

Definition 7 *A response re is called* **valid** *to a challenge ch if $re = \mathsf{Response}(pp, M^*, ch)$ and the challenge/response (ch, re) is called a* **valid** *pair. A response re is called* **consistent** *to a challenge ch if $\mathsf{Verification}(pp, sk, ch, re) = 1$, and the challenge/response (ch, re) is called a* **consistent** *pair.*

Given a challenge ch, there may exists many **consistent** responses, but there is only one **valid** response due to the deterministic algorithm $\mathsf{Response}$. Hence, a consistent pair is not necessarily a valid pair.

Before the formal proof of Unforgeability and Soundness of the MRD-based PoDP/PoR scheme, we will slightly change the PoDP/PoR scheme. The key sequence $\mathsf{k}_i = \mathsf{PRF}(k', i), i = 1, 2, \cdots, w$, which is used to compute tags for blocks and to verify the consistency of a challeng/response pair, is replaced with a truly random sequence over $\mathbb{F}_{q^t}$. Then we have the following two lemmas.

Lemma 1 *Suppose that $\mathsf{k}_i, i = 1, 2, \cdots, w$, which are involved in algorithms $\mathsf{Encode}(sk, pp, M)$ and $\mathsf{Verification}(pp, ch, re)$ of the above PoDP/PoR scheme, is randomly chosen from $\mathbb{F}_{q^t}$ instead of setting*

$\mathsf{k}_i = \mathsf{PRF}(k', i)$. *Suppose that some blocks of* M^* *are missing. In an execution of* Interaction *Protocol of the PoDP/PoR scheme, suppose that the challenge vector* $ch = (v_1, v_2, \cdots, v_w) \in (\mathbb{F}_q)^w$ *hits at least one deleted block of* M^**, i.e, there exists an index* j *such that* $v_j \neq 0$ *and the* j*-th block of* M^* *is missing, then any adversary* $\mathcal{A}$ *presents a consistent response* $re = (\bar{\mathsf{c}}_1, \bar{\mathsf{c}}_2, \cdots, \bar{\mathsf{c}}_n, \bar{\boldsymbol{\sigma}})$ *with probability at most* $1/q^n$.

Proof: Suppose that there are e deleted blocks, and let $M^*|_{un} = \left(C_{i_1}||\boldsymbol{\sigma}_{i_1}, C_{i_2}||\boldsymbol{\sigma}_{i_2}, \cdots, C_{i_{w-e}}||\boldsymbol{\sigma}_{i_{w-e}}\right)^{\mathsf{T}}$ denote the remaining $(w-e)$ undeleted blocks. According to Eq.(2), we have

$$\boldsymbol{\sigma}_{i_l} = C_{i_l} \cdot \mathsf{k} + \mathsf{k}_{i_l}, l = 1, 2, \cdots, w - e. \tag{4}$$

The challenge vector is $ch = (v_1, v_2, \cdots, v_w)$. For any adversary $\mathcal{A}$ who gives a consistent response $re = (\bar{\mathsf{c}}_1, \bar{\mathsf{c}}_2, \cdots, \bar{\mathsf{c}}_n, \bar{\boldsymbol{\sigma}})$ satisfying $\mathsf{Verification}(pp, sk, ch, re) = 1$, let $\bar{C} = (\bar{\mathsf{c}}_1, \bar{\mathsf{c}}_2, \cdots, \bar{\mathsf{c}}_n)$, then we have

$$\bar{\boldsymbol{\sigma}} = \bar{C} \cdot \mathsf{k} + \sum_{l=1}^{w} v_l \circ \mathsf{k}_l. \tag{5}$$

Let $\{j_1, j_2, \cdots, j_e\}$ denote the indices of the blocks missing from $M^*|_{un}$. Eq.(5) and Eq.(4) gives

$$\bar{\boldsymbol{\sigma}} - \sum_{l=1}^{w-e} v_{i_l} \circ \boldsymbol{\sigma}_{i_l} = \left(\bar{C} - \sum_{l=1}^{w-e} v_{i_l} \circ C_{i_l}\right) \cdot \mathsf{k} + \sum_{l=1}^{e} v_{j_l} \circ \mathsf{k}_{j_l}. \tag{6}$$

To the adversary $\mathcal{A}$ who knows the deleted version $M^*|_{un}$, forging a consistent response $re = \left(\bar{C}, \bar{\boldsymbol{\sigma}}\right)$ satisfying Eq.(5) is equivalent to forging a pair $(\tilde{C}, \tilde{\boldsymbol{\sigma}})$ satisfying

$$\tilde{\boldsymbol{\sigma}} = \tilde{C} \cdot \mathsf{k} + \sum_{l=1}^{e} v_{j_l} \circ \mathsf{k}_{j_l}. \tag{7}$$

Fixing a pair $(\tilde{C}, \tilde{\boldsymbol{\sigma}})$, now we count how many k and k_{j_l}, $l = 1, 2, \cdots, e$, satisfy Eq.(7). There exists at least one nonzero entry in $(v_{j_1}, v_{j_2}, \cdots, v_{j_e})$, since the challenger vector hits at least one deleted block. Without loss of generality, we assume that $v_{j_e} \neq 0$. When k and k_{j_l}, $l = 1, 2, \cdots, e-1$, are freely chosen, k_{j_e} will be uniquely determined by Eq.(7). That counts to $q^{n+t(e-1)}$ for Eq.(7). Since k and k_{j_l} are uniformly distributed, an adversary $\mathcal{A}$ can present a consistent response with probability $q^{n+t(e-1)}/q^{n+te} = q^{-t} \leq q^{-n}$, due to $t \geq n$. □

Lemma 2 *Suppose that* $\mathsf{k}_i, i = 1, 2, \cdots, w$ *is randomly chosen from* $\mathbb{F}_{q^t}$ *instead of setting* $\mathsf{k}_i = \mathsf{PRF}(k', i)$*. In an execution of* Interaction *Protocol of the PoDP/PoR scheme, given a valid challenge/response pair* (ch, re)*, any adversary* $\mathcal{A}$ *outputs a consistent but invalid response* re' *with respect to the same challenge* ch*, i.e.,* $re' \neq re$ *but* $\mathsf{Verification}(pp, sk, ch, re') = 1$*, with probability at most* $1/q^d$*, where* d *is the minimum rank distance of the MRD Code.*

Proof: Let $M^* = (C_1||\boldsymbol{\sigma}_1, C_2||\boldsymbol{\sigma}_2, \cdots, C_w||\boldsymbol{\sigma}_w)^{\mathsf{T}}$ be the storage file generated by $\mathsf{Encode}(sk, pp, M)$. Let $ch = (v_1, v_2, \cdots, v_w)$ be the challenge vector.

Let $re = \left(\sum_{i=1}^{w} v_i \circ \mathsf{c}_{i1}, \cdots, \sum_{i=1}^{w} v_i \circ \mathsf{c}_{in}, \sum_{i=1}^{w} v_i \circ \boldsymbol{\sigma}_i\right)$ be the valid response computed by $\mathsf{Response}(pp, M^*, ch)$. Define $\bar{\mathsf{c}}_j = \sum_{i=1}^{w} v_i \circ \mathsf{c}_{ij}$ and $\bar{\boldsymbol{\sigma}} = \sum_{i=1}^{w} v_i \circ \boldsymbol{\sigma}_i$, then $re = (\bar{\mathsf{c}}_1, \bar{\mathsf{c}}_2, \cdots, \bar{\mathsf{c}}_n, \bar{\boldsymbol{\sigma}})$.

Let $re' = \left(\bar{\mathsf{c}}'_1, \bar{\mathsf{c}}'_2, \cdots, \bar{\mathsf{c}}'_n, \bar{\boldsymbol{\sigma}'}\right)$ be the consistent but invalid response forged by adversary $\mathcal{A}$ for challenge ch. The invalidity of re' implies $re \neq re'$. The consistency of re' requires that $\bar{\boldsymbol{\sigma}'}$ is determined by $(\bar{\mathsf{c}}'_1, \bar{\mathsf{c}}'_2, \cdots, \bar{\mathsf{c}}'_n)$. Consequently, $re \neq re'$ means $(\bar{\mathsf{c}}_1, \bar{\mathsf{c}}_2, \cdots, \bar{\mathsf{c}}_n) \neq (\bar{\mathsf{c}}'_1, \bar{\mathsf{c}}'_2, \cdots, \bar{\mathsf{c}}'_n)$.

Now we analyze the probability the pair (ch, re') is invalid but consistent.

First we assume that $(\bar{\mathsf{c}}'_1, \bar{\mathsf{c}}'_2, \cdots, \bar{\mathsf{c}}'_n)$ is also a Gabidulin codeword, otherwise it would have been rejected by the first step of Verification.

The fact $\mathsf{Verification}(pp, sk, ch, re) = 1$ and $\mathsf{Verification}(pp, sk, ch, re') = 1$ gives us the following equations.

$$\bar{\boldsymbol{\sigma}} = (\bar{\mathsf{c}}_1, \bar{\mathsf{c}}_2, \cdots, \bar{\mathsf{c}}_n) \cdot \mathsf{k} + \sum_{i=1}^{w} v_i \circ \mathsf{k}_i, \quad \bar{\boldsymbol{\sigma}'} = (\bar{\mathsf{c}}'_1, \bar{\mathsf{c}}'_2, \cdots, \bar{\mathsf{c}}'_n) \cdot \mathsf{k} + \sum_{i=1}^{w} v_i \circ \mathsf{k}_i. \tag{8}$$

Define $\mathsf{k}' = \sum_{i=1}^{w} v_i \circ \mathsf{k}_i$. We know that k_i is randomly chosen from $\mathbb{F}_{q^t}$ for $i = 1, 2, \cdots, w$, hence k' is also a random element in $\mathbb{F}_{q^t}$. Then Eq. (8) are reformed with

$$\bar{\boldsymbol{\sigma}} = (\bar{\mathsf{c}}_1, \bar{\mathsf{c}}_2, \cdots, \bar{\mathsf{c}}_n) \cdot \mathsf{k} + \mathsf{k}', \quad \bar{\boldsymbol{\sigma}'} = (\bar{\mathsf{c}}'_1, \bar{\mathsf{c}}'_2, \cdots, \bar{\mathsf{c}}'_n) \cdot \mathsf{k} + \mathsf{k}'. \tag{9}$$

Define $\Delta\bar{\mathsf{c}}_j = \bar{\mathsf{c}}_j - \bar{\mathsf{c}}'_j$ and $\Delta\bar{\boldsymbol{\sigma}} = \bar{\boldsymbol{\sigma}} - \bar{\boldsymbol{\sigma}'}$, then Eq. (9) are equivalent to the following two equations

$$\bar{\boldsymbol{\sigma}} = (\bar{\mathsf{c}}_1, \bar{\mathsf{c}}_2, \cdots, \bar{\mathsf{c}}_n) \cdot \mathsf{k} + \mathsf{k}', \quad \Delta\bar{\boldsymbol{\sigma}} = (\Delta\bar{\mathsf{c}}_1, \Delta\bar{\mathsf{c}}_2, \cdots, \Delta\bar{\mathsf{c}}_n) \cdot \mathsf{k}. \tag{10}$$

Next we determine the number of solutions for the unknowns k and k'. To satisfy the left equation of Eq. (10), each freely chosen value of k uniquely determines the value of k', and there are totally q^n choices for k.

As to the right equation of Eq. (10), we express all the entries as vectors over $\mathbb{F}_q$, then it turns to

$$\begin{pmatrix} \Delta\bar{\sigma}^{(1)} \\ \Delta\bar{\sigma}^{(2)} \\ \vdots \\ \Delta\bar{\sigma}^{(t)} \end{pmatrix} = \begin{pmatrix} \Delta\bar{c}_1^{(1)} & \Delta\bar{c}_2^{(1)} & \cdots & \Delta\bar{c}_n^{(1)} \\ \Delta\bar{c}_1^{(2)} & \Delta\bar{c}_2^{(2)} & \cdots & \Delta\bar{c}_n^{(2)} \\ \vdots & \vdots & & \vdots \\ \Delta\bar{c}_1^{(t)} & \Delta\bar{c}_2^{(t)} & \cdots & \Delta\bar{c}_n^{(t)} \end{pmatrix} \cdot \begin{pmatrix} k^{(1)} \\ k^{(2)} \\ \vdots \\ k^{(n)} \end{pmatrix} \triangleq \Delta\bar{C} \cdot \begin{pmatrix} k^{(1)} \\ k^{(2)} \\ \vdots \\ k^{(n)} \end{pmatrix}. \tag{11}$$

Due to the linear property of the Gabidulin code, $(\Delta\bar{\mathsf{c}}_1, \Delta\bar{\mathsf{c}}_2, \cdots, \Delta\bar{\mathsf{c}}_n)$ should also be a non-zero Gabidulin codeword. Therefore, the matrix $\Delta\bar{C}$ has rank at least d. Consequently, Eq. (11) has a solution space of size at most q^{n-d} for k.

Since k and k' are chosen uniformly at random, the probability that the adversary $\mathcal{A}$ forges an invalid but consistent (ch, re') pair, given the valid pair (ch, re) in an Interaction Protocol, is upper-bounded by $q^{n-d}/q^n = \frac{1}{q^d}$. □

6.1 The Unforgeability of the PoDP/PoR Scheme

Now we are going to prove that if some blocks of the file M^* are missing, a PPT storage provider cannot present a consistent response with probability ζ, such that $\zeta - 1/q$ is non-negligible. This justifies the necessity of the threshold $1/q$ in the audit strategy of the PoDP/PoR scheme.

Theorem 3 *The MRD-code-based PoDP/PoR system is (δ, ζ)-unforgeable with $\zeta \leq 1/q^{\delta w} + 1/q^n + \mathsf{Adv}_{\mathsf{PRF}}(\lambda)$. Here, w is the number of blocks of the storage file, and q is the size of the finite field on which the MRD code is constructed.*

Proof: For any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, any $(sk, pp) \leftarrow \mathsf{KeyGen}(1^\lambda)$, any $M \leftarrow \mathcal{A}_1(pp)$, any $M^* \leftarrow \mathsf{Encode}(sk, pp, M)$, any $\tilde{M}^* \leftarrow \mathcal{A}_1(pp, M, M^*)$, where $\tilde{M}^*$ is a deleted version of M^* with δ fraction missing, and any $ch \leftarrow \mathsf{Challenge}(1^\lambda)$, we consider the probability that $\mathcal{A}_2(pp, \tilde{M}^*, ch)$ outputs a response re' satisfying $\mathsf{Verification}(pp, sk, ch, re') = 1$.

We prove the theorem via two indistinguishable games, Game 0, and Game 1. Let Game 0 be the original Game $\mathsf{Unforge}_{\mathcal{A}}^{PoDP}(\lambda)$. Define Win_i as the event that $\mathcal{A}$ wins in Game i. Our aim is to determine $\Pr[\mathsf{Win}_0]$.

Game 0. This is the original Game $\mathsf{Unforge}_{\mathcal{A}}^{PoDP}(\lambda)$ defined in subsection 2.3.

Game 1. This game is the same as Game 0 except for the Key generation. In Game 0, the secret key generated by the Challenger is $sk = (\mathsf{k}, k')$, where $\mathsf{k} \leftarrow \mathbb{F}_{q^n}$ and $k' \leftarrow \mathcal{K}_{prf}$. To encode M to M^*, k' is used as a seed in a Pseudo-Random Function to generate a pseudo-random sequence $\mathsf{k}_i = \mathsf{PRF}(k', i)$, $i = 1, 2, \cdots, w$, which is used to generate tags for each block. The same pseudo-random sequence is used for testing the consistence of responses in **Verification**.

In Game 1, the challenger picks truly random elements from $\mathbb{F}_{q^t}$ instead of the PRF output for k_i, $i = 1, 2, \cdots, w$, and sets $sk = (\mathsf{k}, \mathsf{k}_1, \mathsf{k}_2, \cdots, \mathsf{k}_w)$, which are used directly in **Tag Generation** and **Verification**.

Any difference in the adversary's success probability between Game 0 and Game 1 results in algorithm breaking the security of the PRF. Therefore,

$$|\Pr[\mathsf{Win}_0] - \Pr[\mathsf{Win}_1]| \leq \mathsf{Adv}_{\mathsf{PRF}}(\lambda). \tag{12}$$

In Game 1, the adversary $\mathcal{A}_2$ possesses $\tilde{M}^*$, which is a deleted version of M^* with δ fraction missing.

Let Hit denote the event that the challenge vector $ch = (v_1, v_2, \cdots, v_w)$ hits at least one missing block of $\tilde{M}^*$. In other words, there exists at least an index j such that $v_j \neq 0$ and j-th block is missing in $\tilde{M}^*$.

There are about δw blocks missing in $\tilde{M}^*$. Since v_i is chosen independently and uniformly random from $\mathbb{F}_q$, v_i does not hit the i-th block with probability $1/q$. Hence $\Pr[\neg\mathsf{Hit}] = 1/q^{\delta w}$.

If Hit does happen, then Win_1 implies the adversary $\mathcal{A}_2$ presents a consistent response with $\tilde{M}^*$. According to Lemma 1, this probability is at most $1/q^n$.

Therefore,

$$\begin{aligned}\Pr[\mathsf{Win}_1] =& \Pr[\mathsf{Win}_1 \mid \neg\mathsf{Hit}]\Pr[\neg\mathsf{Hit}] + \Pr[\mathsf{Win}_1 \mid \mathsf{Hit}]\Pr[\mathsf{Hit}] \\ \leq& \Pr[\neg\mathsf{Hit}] + \Pr[\mathsf{Win}_1 \mid \mathsf{Hit}] \leq 1/q^{\delta w} + 1/q^n.\end{aligned} \tag{13}$$

Combining (13) with (12), we have $\zeta = \Pr[\mathsf{Win}_0] \leq 1/q^{\delta w} + 1/q^n + \mathsf{Adv}_{\mathsf{PRF}}(\lambda)$. □

When $q^n \geq 2^\lambda$, then $1/q^n + \mathsf{Adv}_{\mathsf{PRF}}(\lambda)$ is negligible due to the security of PRF. The minimal value of δ is $1/w$, in which case there is only one block missing in $\tilde{M}^*$. In that case, $\zeta - 1/q$ is negligible. This testifies the necessity of the threshold $1/q$ in the audit strategy of our scheme.

6.2 ϵ-Soundness of the PoDP/PoR Scheme

Even a truthful storage provider may malfunction in an audit due to some problem that it is capable of fixing later. For example, in a distributed storage system, several storage servers access the same data at the same time. It may overkill to claim an audit fails just because only one or two responses fail the verification. To be more robust, it is desirable that the PoDP/PoR system supports an ϵ-admissible storage prover, where the storage prover convincingly answers an ϵ fraction of challenges in an audit. The problem is how to determine the lower bound of ϵ so that we can still prove the retrievability of the file M. This is characterized by the *ϵ-soundness of Proof of Retrievability (PoR)* in subsection 2.4.

Now we consider how a PPT $\mathsf{Extractor}$ extracts the original file M through executions of the $\mathsf{Interaction}$ protocol with an ϵ-admissible storage provider. The idea is as follows: ϵ should be big enough to ensure that $\mathsf{Extractor}$ always gains information about M as the number of interactions with $\tilde{\mathcal{P}}_\epsilon$ increases. When the number of interactions in the audit is big enough, M can be recovered by solving equations.

Theorem 4 *The MRD-code-based PoDP/PoR system is ϵ-sound for $\epsilon > \frac{1}{q}$, given secure PRF and $[q, k, n, t]$ Gabidulin code with $t \geq n$ and $n - k + 1 = \Omega(\lambda)$.*

Proof: We want to construct a PPT $\mathsf{Extractor}$, such that for any PPT adversary $\mathcal{A}$, any $(sk, pp) \leftarrow \mathsf{KeyGen}(1^\lambda)$, any $M \leftarrow \mathcal{A}(pp)$, any $M^* \leftarrow \mathsf{Encode}(sk, pp, M)$, and any $\tilde{\mathcal{P}}_\epsilon \leftarrow \mathcal{A}\big(pp, M, M^*, \text{access to } \mathcal{V}(sk, pp)\big)$, where $\tilde{\mathcal{P}}_\epsilon$ is an ϵ-admissible prover, the $\mathsf{Extractor}$ is able to recover the original file M by querying $\tilde{\mathcal{P}}_\epsilon$ except with negligible probability.

In Game $\mathsf{Sound}^{PoR}_{\mathcal{A}}(\lambda)$, define Fail to be the event that the file M' extracted by the $\mathsf{Extractor}$ is not equal to the original M.

Similarly, we prove the theorem via two indistinguishable games, Game 0, and Game 1. Let Game 0 be the original $\mathsf{Sound}^{PoR}_{\mathcal{A}}(\lambda)$ Game. Define Fail_i to be the event Fail in Game i. Our aim is to prove $\Pr[\mathsf{Fail}_0]$ is negligible.

Game 0. This is the original Game $\mathsf{Sound}^{PoR}_{\mathcal{A}}(\lambda)$ defined in subsection 2.4.

Game 1. This game is the same as Game 0 except for the generation of the secret key and the employment of secret key in the Encode and $\mathsf{Verification}$ algorithm. In Game 1, the Challenger sets $sk = (\mathsf{k}, \mathsf{k}_1, \mathsf{k}_2, \cdots, \mathsf{k}_w)$, with k_i, $i = 1, 2, \cdots, w$, being a truly random sequence over $\mathbb{F}_{q^t}$. This truly random sequence, instead of the pseudo-random sequence generated by $\mathsf{PRF}(k', i)$ in Game 0, is used to encode M to M^* and to verify the validity of a response in $\mathsf{Verification}$. Any difference in the adversary's success probability between Game 0 and Game 1 results in an algorithm breaking the security of the PRF. Therefore,

$$|\Pr[\mathsf{Fail}_0] - \Pr[\mathsf{Fail}_1]| \leq \mathsf{Adv}_{\mathsf{PRF}}(\lambda). \tag{14}$$

In Game 1, suppose the adversary $\mathcal{A}$ interacts with $\mathcal{V}(sk, pp)$ for h times, where h is polynomial in λ, and outputs an ϵ-admissible prover $\tilde{\mathcal{P}}_\epsilon$. Then the Extractor interacts with $\tilde{\mathcal{P}}_\epsilon$ for $u = \frac{w\lambda}{\epsilon - 1/q}$ times. For each challenge sent by $\mathcal{V}$ or by the Extractor, the adversary $\mathcal{A}$ or $\tilde{\mathcal{P}}_\epsilon$ replies a response. Suppose there are a consistent responses in total, then $a \leq h + u$. According to Lemma 2, each consistent response is invalid with probability at most $1/q^d$. Let Bad be the event at least one of the a consistent pairs is invalid in Game 1. By a union bound, we have $\Pr[\mathsf{Bad}] \leq \frac{a}{q^d} \leq \frac{h+u}{q^d}$.

If Bad does not happen, all the consistent responses are also valid responses. Then the Extractor can get at least $u\epsilon$ valid challenge/response pairs through interacting with $\tilde{\mathcal{P}}_\epsilon$, and it is able to use the $u\epsilon$ valid pairs to extract the original file M as follows.

The Extractor filters the output of the u executions of Interaction Protocol, and only records the output of successful executions. Let S denote a set, X be a binary variable, and V_i denote a vector space over $\mathbb{F}_q$ generated by the vector $ch_i = (v_{i1}, v_{i2}, \cdots, v_{iw})$. Let $V_1 + V_2 = \{\mathsf{x} + \mathsf{y} \mid \mathsf{x} \in V_1, \mathsf{y} \in V_2\}$ denote the sum of two vector spaces V_1 and V_2. The Extractor works as follows.

$\mathsf{Extractor}(pp, sk, \tilde{\mathcal{P}}_\epsilon)$
Step 1. Collect enough valid consistent challenge/response pairs:
$S \leftarrow \emptyset$; $V \leftarrow \emptyset$; $X \leftarrow 0$; $b = 0$;
for $i = 1$ to u **do**
$output \leftarrow \mathsf{Interaction}(\tilde{\mathcal{P}}_\epsilon(pp, \cdot) \leftrightharpoons \mathcal{V}(sk, pp))$;
if $output \neq \perp$
parse $output$ as (ch_i, re_i);
$V_i \leftarrow \langle ch_i \rangle$ (V_i is the vector space generated by ch_i);
if $V_i \nsubseteq V$
$V \leftarrow V + V_i$; $b \leftarrow b + 1$; $S \leftarrow S \cup \{(ch_i, re_i)\}$;
if $b = w$
$X = 1$; Return(X, S);
end if
end if
end if
end for
Return(X, S).
Step 2. Solve a system of linear equations to recover M^*, and decode M^* to get M.

In the algorithm, S collects (ch_i, re_i) with independent vector ch_i, b records the number of independent ch_i in S, while X denotes whether the challenge vectors in S spans the whole space, i.e, there exists w independent challenge vectors in S. If $X = 1$, the w independent challenge vectors in S determine a non-singular matrix of w by w, and the Extractor can recover the encoded file M^* with the corresponding re_i, as will show later in the proof.

Among the u executions of the Interaction Protocol, an ϵ fraction of queries are given valid responses. Each valid response is helpless ($V_i \subseteq V$) with probability at most $1/q$. Therefore, there remains at least an $\epsilon - 1/q$ fraction of challenges are helpful to the Extractor. Now we will prove that $X = 1$ happens with overwhelming probability.

In i-th loop, $i = 1, \cdots, u$, define a binary variable

$$Y_i = \begin{cases} 1 & , \text{ if } output \neq \perp \ \wedge \ V_i \nsubseteq V; \\ 0 & , \text{ otherwise.} \end{cases}$$

Then we have

$$\begin{cases} \Pr[Y_i = 0 \mid Y_1, Y_2, \cdots, Y_{i-1}] \leq 1 - \epsilon + \frac{1}{q}; \\ \Pr[Y_i = 1 \mid Y_1, Y_2, \cdots, Y_{i-1}] \geq \epsilon - \frac{1}{q}. \end{cases}$$

Define a new binary variable Z_i (for $i = 1, 2, \cdots, u$) with independent identical probability distribution

$$\begin{cases} \Pr[Z_i = 0] = 1 - \epsilon + \frac{1}{q}; \\ \Pr[Z_i = 1] = \epsilon - \frac{1}{q}. \end{cases}$$

It is easy to see that $\Pr\left[\sum_{i=1}^{u} Y_i < w\right] \leq \Pr\left[\sum_{i=1}^{u} Z_i < w\right]$. Consequently, using the Chernoff bound and $u = \frac{w\lambda}{\epsilon - 1/q}$, we get

$$\begin{aligned}\Pr[X=0] \quad &= \Pr\left[\sum_{i=1}^{u} Y_i < w\right] \leq \Pr\left[\sum_{i=1}^{u} Z_i < w\right] \leq \Pr\left[\sum_{i=1}^{u} Z_i \leq (1-(1-1/\lambda))w\lambda\right] \\ &\leq e^{-\frac{1}{2}(1-1/\lambda)^2 w\lambda} \leq e^{-\frac{1}{8}w\lambda}.\end{aligned}$$

Hence, with probability at least $1 - e^{-\frac{1}{8}w\lambda}$, the set S collects w valid pairs (ch_{i_k}, re_{i_k}) for $k = 1, 2, \cdots, w$, and ch_{i_1}, ch_{i_2}, $\cdots$, ch_{i_w} are independent. In this case, the $\mathsf{Extractor}$ will extract blocks of M^* by solving a system of linear equations as follows.

Let $ch_{i_k} = (v_{i_k 1}, v_{i_k 2}, \cdots, v_{i_k w})$ be the challenge vector, and $re_{i_k} = (\bar{\mathsf{c}}_{i_k 1}, \bar{\mathsf{c}}_{i_k 2}, \cdots, \bar{\mathsf{c}}_{i_k n}, \bar{\boldsymbol{\sigma}}_{i_k})$ be the corresponding valid response, for $k = 1, 2, \cdots, w$. Then $\bar{\mathsf{c}}_{i_k j} = \sum_{s=1}^{w} v_{i_k s} \circ \mathsf{c}_{sj}$, $1 \leq k \leq w$, $1 \leq j \leq n$.

When considering $\mathsf{c}_{sj} = (c_{sj}^{(1)}, c_{sj}^{(2)}, \cdots, c_{sj}^{(t)})^T$ and $\bar{\mathsf{c}}_{i_k j} = (\bar{c}_{i_k j}^{(1)}, \bar{c}_{i_k j}^{(2)}, \cdots, \bar{c}_{i_k j}^{(t)})^T$, $1 \leq k \leq w$, $1 \leq j \leq n$, as vectors of dim t over $\mathbb{F}_q$, we have $\bar{c}_{i_k j}^{(l)} = \sum_{s=1}^{w} v_{i_k s} \cdot c_{sj}^{(l)}$, for $1 \leq k \leq w$, $1 \leq j \leq n$, $1 \leq l \leq t$.

For $l = 1, \cdots, t$, with

$$\begin{pmatrix} c_{11}^{(l)} & c_{12}^{(l)} & \cdots & c_{1n}^{(l)} \\ c_{21}^{(l)} & c_{22}^{(l)} & \cdots & c_{2n}^{(l)} \\ \vdots & \vdots & & \vdots \\ c_{w1}^{(l)} & c_{w2}^{(l)} & \cdots & c_{wn}^{(l)} \end{pmatrix} = \begin{pmatrix} v_{i_1 1} & v_{i_1 2} & \cdots & v_{i_1 w} \\ v_{i_2 1} & v_{i_2 2} & \cdots & v_{i_2 w} \\ \vdots & \vdots & & \vdots \\ v_{i_w 1} & v_{i_w 2} & \cdots & v_{i_w w} \end{pmatrix}^{-1} \cdot \begin{pmatrix} \bar{c}_{i_1 1}^{(l)} & \bar{c}_{i_1 2}^{(l)} & \cdots & \bar{c}_{i_1 n}^{(l)} \\ \bar{c}_{i_2 1}^{(l)} & \bar{c}_{i_2 2}^{(l)} & \cdots & \bar{c}_{i_2 n}^{(l)} \\ \vdots & \vdots & & \vdots \\ \bar{c}_{i_w 1}^{(l)} & \bar{c}_{i_w 2}^{(l)} & \cdots & \bar{c}_{i_w n}^{(l)} \end{pmatrix},$$

data $(C_1, C_2, \cdots, C_w)$ is recovered and the original file $M = (M_1, M_2, \cdots, M_w)$ can be extracted with $\mathsf{GabiDecode}$ algorithm $M_i \leftarrow \mathsf{GabiDecode}(C_i, \mathsf{H}_{n\times(n-k)})$.

Concluding the above comments, we have

$$\begin{aligned}\Pr[\mathsf{Fail}_1] \quad &= \Pr[\mathsf{Fail}_1 \mid \mathsf{Bad}] \Pr[\mathsf{Bad}] + \Pr[\mathsf{Fail}_1 \mid \neg\mathsf{Bad}] \Pr[\neg\mathsf{Bad}] \leq \Pr[\mathsf{Bad}] + \Pr[\mathsf{Fail}_1 \mid \neg\mathsf{Bad}] \\ &\leq \frac{h+u}{q^d} + e^{-\frac{1}{8}w\lambda} = \frac{h + w\lambda/(\epsilon - 1/q)}{q^d} + e^{-\frac{1}{8}w\lambda}. \end{aligned} \tag{15}$$

Combining (15) with (14), the $\mathsf{Extractor}$ fails to extract the original file with probability at most $\Pr[\mathsf{Fail}_0] \leq \mathsf{Adv}_{\mathsf{PRF}}(\lambda) + \frac{h+w\lambda/(\epsilon-1/q)}{q^d} + e^{-\frac{1}{8}w\lambda}$, which is negligible since $h + \frac{w\lambda}{\epsilon - 1/q}$ is polynomial in λ and $d = n - k + 1 = \Omega(\lambda)$. □

7 Self-Audit & Error-Correction of the Storage Provider

To provide dependable data storage service, another requirement for the storage provider is to recover the data from data losses or corruption due to hardware failure or external threats. This involves the provider's capability of detecting and correcting errors in the data.

The MRD code used in the PoDP/PoR scheme makes it convenient for the storage provider to audit itself: to determine the existence of errors, locate the errors and finally correct the errors.

Self-Audit & Error-Correction of the Storage Provider

1. $(v_1, v_2, \cdots, v_w) \leftarrow (\mathbb{F}_q)^w$;
2. Parse the file M^* block-wise as $M^* = (C_1 || \boldsymbol{\sigma}_1, C_2 || \boldsymbol{\sigma}_2, \cdots, C_w || \boldsymbol{\sigma}_w)^{\mathsf{T}}$, where C_i is a $t \times n$ matrix over $\mathbb{F}_q$, $i = 1, 2, \cdots, w$;
3. $S \leftarrow \emptyset$; $a = 1$; $b = w$; $\vec{v} = (v_1, v_2, \cdots, v_w)$;

4. BinarySearch$(a, b, M^*, \vec{v})$

{ $\bar{C} = \sum_{i=a}^{b} v_i \circ C_i$;

if $\bar{C}$ is not a MRD codeword

if $a \neq b$

$f = \lceil \frac{b}{2} \rceil$;

BinarySearch$(a, f, M^*, \vec{v})$;

BinarySearch$(f+1, b, M^*, \vec{v})$;

else $S \leftarrow S \cup \{C_a\}$

end if

end if }

5. If $S = \emptyset$, output ("Success"), otherwise for each element $\tilde{C}_i \in S$, do error-correction to recover the original MRD codeword C_i.

In the above algorithm, set S records the corrupted MRD codewords.

Due to the linearity of the MRD codes, linear combinations of codewords remains to be MRD codewords. The storage provider is able to audit the storage himself by checking whether a random linear combination of codewords is still a codewords, rather than checking each codeword one by one. This greatly reduces the computational complexity of the storage provider. If the linear combination is not a MRD codeword, the provider will use binary search to locate the corrupted codewords, and it needs only $\log w$ searches to locate one corrupted codewords. According to the property of MRD codes, it can correct x-fold row erasures, y-fold column erasures, and z-fold rank errors in $\tilde{C}_i$ simultaneously, provided that $x + y + 2z \leq d - 1$, where d is the minimum rank distance of the MRD Code.

8 Conclusion

In this paper, we propose a PoDP/PoR scheme based on MRD codes. The MRD codes helps in three ways. Firstly, the MRD code can be applied over small field, and that help the storage provider efficiently computes responses in an audit. As to the binary field, the computation can be as simple as XOR. Secondly, the rank property of the MRD codewords helps the security proof of the PoDP/PoR scheme. When one or more blocks are missing, the storage provider can forge a response which passes the client's verification with negligible probability, and this helps to prove the unforgeability of the PoDP/PoR scheme. Even if the storage provider knows a correct response with respect to a challenge, it still cannot forge a different response to pass the client verification. This helps to prove the soundness of the PoDP/PoR scheme. Finally, the error correction ability of MRD code helps the storage provider audit itself to find errors and correct them among the stored data.

References

[1] G. Maximilien, Y. Zhiyuan. Properties of Codes with the Rank Metric[J]. Arxiv preprint cs/0610099, 2006.

[2] R. M. Roth, Maximum-Rank Array Codes and Their Application to Crisscross Error Correction[J]. Information Theory, IEEE Transactions on, Mar 1991, 32(2): 328-336.

[3] Gabidulin E M. Theory of code with maximum rank distance[J]. Problems of Information Transmission, 1985, 21(1): 1-12.

[4] P. Delsarte, Bilinear Forms over a Finite Field with Applications to Coding Theory[J]. Journal of Combinatorial Theory, Series A, 1978, 25(3): 226-241.

[5] Y. Deswarte, J.-J. Quisquater, and A. Saidane. Remote Integrity Checking. In Proc. of Conference on Integrity and Internal Control in Information Systems (IICIS03), November 2003.

[6] Ateniese, G., Burns, R., Curtmola, R., Herring, J., Kissner, L., Peterson, Z. and Song, D. (2007) Provable data possession at untrusted stores. In: De Capitani di Vimercati, S., Syverson, P. (eds.) *CCS '07 Proceedings of the 14th ACM conference on Computer and communications security*, 598-609. ACM Press, New York.

[7] Dodis, Y., Vadhan S. and Wichs, D. (2009) Proofs of retrievability via hardness amplification, *Theory of Cryptography, LNCS* 5444, 109-127.

[8] FILHO, D. and BARRETO, P. (2006) Demonstrating data possession and uncheatable data transfer. *Cryptology ePrint Archive, Report 2006/150*, http://eprint.iacr.org/

[9] E. M. Gabidulin, N. I. Pilipchuk, Error and erasure correcting algorithms for rank codes, Des. Codes Cryptogr. (2008) 49: 105-122.

[10] Juels, A. and Kaliski, B. (2007) PORs: proofs of retrievability for large files. In: De Capitani di Vimercati, S., Syverson, P. (eds.) *Proceedings of CCS 2007*. 584-597. ACM Press, New York.

[11] Naor, M. and Rothblum, G. (2005) The complexity of online memory checking. In: Tardos, E. (ed.) *Proceedings of FOCS 2005*. 573-584. IEEE Computer Society, Los Alamitos.

[12] Shoup, V. Sequences of games: a tool for taming complexity in security proofs. IACR Cryptology ePrint Archive 2004: 332 (2004).

[13] Schwarz, T. and Miller, E. (2006) Store, forget, and check: Using algebraic signatures to check remotely administered storage. In: Ahamad, M., Rodrigues, L. (eds.) *Proceedings of ICDCS 2006*. 12-12. IEEE Computer Society, Los Alamitos.

[14] Shacham, H. and Waters, B. (2008) Compact proofs of retrievability. In *Proceedings of Asiacrypt 2008, LNCS* 5350, 90-107. Springer-Verlag.

[15] H. Wang, C. Xing, and R. Safavi-Naini, Linear Authentication Codes: Bounds and Constructions, IEEE Transactions On Information Theory, Vol. 49, No. 4, pp. 866-872, April 2003.

[16] Wang, C., Wang, Q., Ren, K., Cao, N., and Lou, W (2012), Toward Secure and Dependable Storage Services in Cloud Computing, *IEEE Transactions on Services Computing,* Vol. 5 , Issue. 2, pp. 220-232.

[17] Wachter, A., Sidorenko, A., Bossert, M., A Fast Linearized Euclidean Algorithm for Decoding Gabidulin Codes, In Twelfth International Workshop on Algebraic and Combinatorial Coding Theory (ACCT 2010), pp. 298-303, September 2010.