New proofs of retrievability using locally decodable codes

Julien Lavauzelle, Françoise Levy-Dit-Vehel

New Proofs of Retrievability using Locally Decodable Codes

Julien Lavauzelle
LIX and INRIA Saclay
Bâtiment Alan Turing
1 rue Honoré d'Estienne d'Orves
91120 Palaiseau
lavauzelle@lix.polytechnique.fr

Françoise Levy-dit-Vehel
ENSTA ParisTech
828 boulevard des Maréchaux
91762 Palaiseau
INRIA Saclay and LIX
levy@ensta.fr

***Abstract*—Proofs of retrievability (PoR) are probabilistic protocols which ensure that a client can recover a file he previously stored on a server. Good PoRs aim at reaching an efficient trade-off between communication complexity and storage overhead, and should be usable an unlimited number of times.**

We present a new unbounded-use PoR construction based on a class of locally decodable codes, namely the lifted codes of Guo *et. al.*. Our protocols feature sublinear communication complexity and very low storage overhead. Moreover, the various parameters can be tuned so as to minimize the communication complexity (resp. the storage overhead) according to the setting of concern.

***Index Terms*—proofs of retrievability, locally decodable codes, lifted codes, cryptographic protocols, data storage**

I. Introduction

Proofs of Retrievability (PoR) are cryptographic protocols aiming at addressing the following issue: given a client who stored a file on a server and erased its local copy, how can the client check if he is able to retrieve his file from the server, without any loss? PoRs thus involve two parties: a client (or verifier) and a server (or prover). Three main phases can be highlighted: first, the client encodes and/or encrypts his file and sends it to the server; then, client and server run challenge-response rounds to check the file's extractibility; finally, if he is convinced, the client runs an extraction procedure to recover his file. Given a security bound on the extraction reliability, PoRs are targeting to minimize the communication cost between both parties, their computation complexity and their storage overhead.

PoRs were formally introduced by Juels and Kaliski (JK) [1] in 2007. Their construction is based on randomly chosen *sentinel symbols* the client secretly keeps before uploading his file. Then the verification process consists in checking the integrity of some sentinels. Since the sentinel positions are unknown to the server, if the client file is not retrievable, then w.h.p. some sentinels should be corrupted, and the client can notice it. Therefore, this scheme has low communication but remains bounded-use, as the number of verifications depends on how many sentinels the client actually possess.

Aiming at unboundedness, Shacham and Waters (SW) [2] proposed to append to the file some authenticator symbols, and then to check random linear combinations of file symbols and authenticators. Verifying the file just consists in comparing parity symbols and the decryption of the encrypted ones. File extraction is then possible by the additional use of an efficient erasure code.

Part of BJO scheme can be seen as a double-layer encoding: an *inner code* is used to recover information symbols from parity-check ones; an *outer code* helps with correcting erasures the inner code could have left. Dodis *et. al.* [3] follow this idea to formalize the verification process as a request to a *PoR-code*, *i.e.* an error correcting code which models the space of possible answers to a challenge. Varying the underlying codes, they find good theoretical PoRs (using list-decodability bounds). However, most of their constructions are bounded-use, and the unbounded-use one does not really improve on SW scheme, despite lighter cryptographic assumptions.

Following Dodis *et. al.*, we carry on with using codes in PoRs. We investigate code constructions which are naturally suited for these protocols, namely *locally decodable codes* (LDC) and more specifically the high rate *lifted codes* introduced by Guo, Kopparty and Sudan in [4]. In the sequel, we introduce PoRs whose verification process is based on the local structure of these codes (section III), and whose security is analysed in section IV. We then give theoretical parameters and practical settings for our PoRs in section V. We concede that our practical results are not especially better than previous ones. Nevertheless, using LDCs in PoRs is new and opens a new line of research.

II. Definitions and security model

Proofs of retrievability involve two parties: a client who owns a (usually huge) file F, and a server on which F is stored. We essentially follow and lighten BJO [5] definitions. A PoR system is composed of three main procedures:

- *An initialisation phase.* The client encodes his file F with an initialisation function $\mathbf{Init}(F) = (\tilde{F}, \texttt{data})$. He keeps `data` (*e.g.* keys, authenticator symbols, etc.) for himself, then he sends $\tilde{F}$ to the server and erases F.

- *A verification phase.* The client produces a challenge c with a randomized $\mathbf{Chall}$ function, then he sends c to the server. The latter creates a response $r = \mathbf{Resp}(\tilde{F}, c)$ and sends it back to the client. The client checks if r is correct by running $\mathbf{Verif}(c, r)$, which also access `data`, and outputs `true` if r is considered as a correct answer, `false` otherwise.
- *An extraction phase.* If the client has been convinced by the verification phase, he can use his $\mathbf{Extract}$ algorithm to recover his whole file with high probability.

The client wants to be (almost) sure that he retrieves his file undamaged when using the $\mathbf{Extract}$ algorithm. But in practice, the server can always make the file unretrievable, for example by erasing it entirely. The point is that we want to model the fact that, if the server's answers to client's challenges make him look to own the file, then the client must be able to recover it entirely. Thus, an adversarial model (representing the server's behaviour) must be defined first. Once again, we follow BJO [5] context.

Definition II.1 (ϵ-adversary)**.** Let $\mathcal{P}$ be a PoR system and X be the space of challenges generated by $\mathbf{Chall}$. An ϵ-adversary $\mathcal{A}$ for $\mathcal{P}$ is an algorithm such that, for all files F:

$$\mathbb{P}_{x \in X}\big[\, \mathbf{Verif}(x, \mathcal{A}(x)) = \texttt{false} \,\big] \leq \epsilon\,.$$

The key idea is the following: the client models the server as an ϵ-adversary, and his verification process is used to maintain an approximation of ϵ. Depending on this estimate, he can decide whether his file is retrievable or not. We thus define a way to measure PoRs' security:

Definition II.2 (PoR security)**.** Let $\epsilon, \rho \in [0, 1]$. A PoR system is said to be (ϵ, ρ)-sound if, for all ϵ-adversaries $\mathcal{A}$ and for all files F, we have:

$$\mathbb{P}\big[\, \mathbf{Extract}^{\mathcal{A}} = F \,\big] \geq \rho\,,$$

where the probability is taken over the internal randomness of $\mathbf{Extract}^{\mathcal{A}}$.

Parameter ρ represents the soundness of the scheme: we want $\rho > 1 - 2^{-\lambda}$, where λ is a desired security parameter.

III. Our PoR Construction

In [3], error correcting codes are used as a framework to model the verification process. We will show that some codes which naturally possess a local decodability property can have a greater role in PoRs.

A. Locally decodable codes and lifted codes

Error correcting codes are usually used to ensure reliable communication/storage in a noisy environment. To fully decode a message, best decoding algorithms reach almost-linear complexity in the length of the code. However, with the emergence of massive data storage, the question of correcting errors in sublinear time appeared. Hadamard code and Reed-Muller codes are known for a long time to provide this *local decodability* property, but their rate is stuck below $1/2$. Major breakthroughs were rencently made by Kopparty *et. al.* [6], Guo *et. al.* [4], and Hemenway *et. al.* [7]. They all give high-rate *locally decodable codes* (LDC) constructions, in a sense that, asymptotically in the code length, the code rate can be made close to 1. For more information on LDCs, see Yekhanin's survey [8]. It has to be pointed out that high rates can practically be reached with reasonable lengths, as we show in section V.

Here we especially focus on Guo *et. al.* construction, named *lifted codes* [4]. Let $\mathbb{F}_q$ denote the finite field with q elements and S be any finite set. From now on we identify a function $f : S \to \mathbb{F}_q$ with its vector of evaluations $(f(x))_{x \in S}$. Thus, some error correcting codes can be represented as subsets of functions. For example, $\{f : \mathbb{F}_q \to \mathbb{F}_q,\ \deg f < k\}$ is the q-ary Reed-Solomon code of length $n = q$ and minimum distance $d = q - k + 1$.

Definition III.1 (Affine-invariant code)**.** Let $\mathbb{F}_Q$ be an extension of $\mathbb{F}_q$. A code $\mathcal{C} \subseteq \{f : \mathbb{F}_Q^t \to \mathbb{F}_q\}$ is said to be affine-invariant if, for all affine permutations $T : \mathbb{F}_Q^t \to \mathbb{F}_Q^t$, we have $f \circ T \in \mathcal{C}$.

GKS [4] builds a LDC by picking an affine-invariant *base code* $\mathcal{C} \subseteq \{f : \mathbb{F}_Q \to \mathbb{F}_q\}$, and then *lifting* it to a subset $\mathcal{L}$ of $\{g : \mathbb{F}_Q^m \to \mathbb{F}_q\}$, such that each affine one-dimensional restriction of functions in $\mathcal{L}$ belongs to $\mathcal{C}$. More formally:

Definition III.2 (GKS affine lifting [4])**.** Let $\mathcal{C} \subseteq \{f : \mathbb{F}_Q \to \mathbb{F}_q\}$ be an affine-invariant code. The m-th affine lift of $\mathcal{C}$ is the affine-invariant code:

$$\mathrm{Lift}_m(\mathcal{C}) = \{g : \mathbb{F}_Q^m \to \mathbb{F}_q \,|\, \forall \text{ affine injections } T : \mathbb{F}_Q \to \mathbb{F}_Q^m,\ g \circ T \in \mathcal{C}\}\,.$$

It is clear that these lifted codes have a local decodability property: to locally decode a codeword $c \in \mathcal{L} = \mathrm{Lift}_m(\mathcal{C})$, one just restricts c to an affine line and uses the decodability of the base code to recover some symbols.

B. Our construction

We now build our PoR based on lifted codes. Recall that in a PoR scheme, we want to know with high probability and sublinear communication whether a large file is still retrievable. Our main idea is to use the local structure of a lifted code to check the file integrity. Thus, the client encodes his file F as a lifted codeword $\tilde{F} \in \mathrm{Lift}_m(\mathcal{C})$, and the verification procedure essentially consists in checking if symbols aligned on a random line form a codeword of $\mathcal{C}$. However, as the client wants to prevent from malicious strategies, he will also need to encrypt his encoded file.

We thus assume the client has access to a family of symmetric stream ciphers ϕ_κ, where κ belongs to some key space $\mathcal{K}$ chosen according to the context, with the following property:

one can get the i-th key stream symbol in constant time. For instance, block ciphers in counter (CTR) mode are suitable for such stream ciphers (see example 9.9 in [9]).

We sum up the initialisation procedure in the algorithm of Fig. 1.

Input: a file F.
Output: an encryption key κ and the ready-to-send file $\tilde{F}$.
1: Pick uniformly at random an encryption key $\kappa \in \mathcal{K}$.
2: Encode F to F' using an encoding procedure of a lifted code $\mathrm{Lift}_m(\mathcal{C})$.
3: Encrypt each symbol c_i of F' with a block-cipher ϕ_κ : $\tilde{c}_i = \phi_\kappa(c_i, i)$.
4: **return** $(\texttt{data} = \kappa,\ \tilde{F} = \{\tilde{c}_i\}_i)$.

Fig. 1. Initialisation algorithm. Code $\mathcal{C}$, integer m and key space $\mathcal{K}$ are system parameters.

After his file is sent, the client can check its integrity: he simply needs to ask for a random 1-dimensional restriction of $\tilde{F}$, and check that the associated decrypted word lies in $\mathcal{C}$. We summarize the verification phase in Fig. 2.

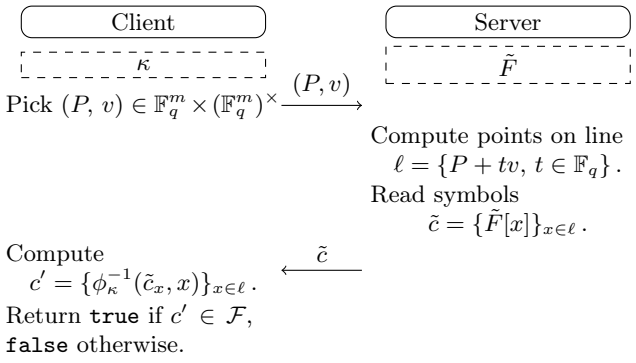

Fig. 2. Verification protocol

Finally, if the verification protocol succeeds a sufficient number of times, the extraction algorithm of Fig. 3 ensures that with very high probability, the file will be recovered.

IV. Security analysis

In this section, we prove that our PoR succeeds with very high probability. We denote by $(\mathcal{C}, m, \gamma)$**-LiftPoR** our PoR construction which uses a base code $\mathcal{C} \subseteq \{\mathbb{F}_Q \to \mathbb{F}_q\}$, a lifting parameter m and voting parameter γ.

Informally, our PoR security is based on the fact that it is implausible for the server to create answers that pass the verification test but do not correspond to correct symbols. To represent these answers, we need to introduce what we call *spurious codewords*:

Definition IV.1. Let $\mathcal{P}$ = $(\mathcal{C}, m, \gamma)$**-LiftPoR** be our PoR scheme, and $\tilde{F}$ the encrypted encoded file held by the server. A *spurious codeword* for $(\mathcal{P}, \tilde{F})$ related to a line $\ell \subset \mathbb{F}_Q^m$ is a word $w \in \mathbb{F}_Q^Q$ such that $w \neq \tilde{F}_{|\ell}$ and $\mathbf{Verif}(\ell, w) = \texttt{true}$.

Input: $\gamma \in [0, 1]$ a parameter for voting ($\gamma = 0.5$ means majority voting).
Output: the file $\tilde{F}$.
——— *1. Initialisation* ———
1: **for all** $x \in \mathbb{F}_q^m$ **do**
2: $v_x \leftarrow [0]_{t \in \mathbb{F}_q}$
3: **end for**
——— *2. Client-server interaction* ———
4: **for all** lines $\ell \subset \mathbb{F}_q^m$ **do**
5: Run the verification procedure with some random line ℓ. Let $\tilde{c}$ be the decrypted server's answer.
6: **if** $\mathbf{Verif}_{\texttt{data}}(\ell, \tilde{c}) = \texttt{true}$ **then**
7: **for all** $x \in \ell$ **do**
8: Increment $v_x[\tilde{c}_x]$
9: **end for**
10: **end if**
11: **end for**
——— *3. Vote* ———
12: **for all** $x \in \mathbb{F}_q^m$ **do**
13: **if** $\exists s \in \mathbb{F}_q,\ v_x[s] > \gamma L$ **then**
14: $\texttt{Tab}[x] = \mathrm{argmax}_s\, v_x[s]$
15: **else**
16: $\texttt{Tab}[x] = \perp$
17: **end if**
18: **end for**
——— *4. Decoding* ———
19: **for all** $x \in \mathbb{F}_q^m$ such that $\texttt{Tab}[x] = \perp$ **do**
20: Find a line ℓ_x passing through x such that $\#\{y \in \ell_x,\ \texttt{Tab}[y] = \perp\} < d$
21: Decode the word $\texttt{Tab}_{|\ell_x}$ with $\mathcal{C}$'s decoding algorithm, and update `Tab` on ℓ_x with the codeword.
22: **end for**
23: **return** `Tab`.

Fig. 3. Extraction algorithm. The parameters q, m and the code $\mathcal{C}$ are public. We set $L = \frac{q^m - 1}{q - 1}$ the number of lines passing through any point of $\mathbb{F}_q^m$.

Let us try to estimate the best chance for any adversary $\mathcal{A}$ to provide a spurious codeword on a challenge ℓ. Upon receiving ℓ, the adversary may know all the encrypted symbols $y = (\phi_\kappa(c_x, x))_{x \in \ell}$ asked by the client. If he wants to fool the client, the adversary needs to modify y on some positions in order to construct $y' = (\phi_\kappa(c'_x, x))_{x \in \ell}$ with $c' \in \mathcal{C} \setminus \{c\}$. Now, if we assume that the cipher ϕ_κ looks to $\mathcal{A}$ like a random permutation, changing $y_x = \phi_\kappa(c_x, x)$ into $y'_x = \phi_\kappa(c'_x, x)$ is the same as uniformly picking a random $c'_x \in \mathbb{F}_q \setminus \{c_x\}$.

Thus, with this assumption on ϕ_κ, the *only* strategy for $\mathcal{A}$ is:

1) Pick a random subset $I \subseteq \ell$;
2) For all $i \in I$, change y_x into random $y'_x \in \mathbb{F}_q \setminus \{y_x\}$.

Hence, the only degree of freedom for $\mathcal{A}$ to optimize his strategy remains in his choice of the subset I.

In order to make the security analysis easier, we specify our PoR construction by instantiating the base code $\mathcal{C}$ as

a Reed-Solomon code. Besides their MDS property, they are practically the best base codes w.r.t. the rate of their lifts.

Proposition IV.2. *Let $\mathcal{A}$ be an adversary for a $(\mathcal{C}, m, \gamma)$-**LiftPoR** where $\mathcal{C}$ is the q-ary Reed-Solomon code of minimum distance d. Then, for any line $\ell \subset \mathbb{F}_q^m$, the probability (over $\mathcal{A}$'s randomness) that $\mathcal{A}$ produces a spurious codeword on challenge ℓ is upper bounded by $(q-1)^{-d+1}$.*

Proof. The proof is quite long and technical so we only sketch it here. For $I \subseteq \mathbb{F}_q$, let W_I denote words in $\mathbb{F}_q^q$ with support I. Following our previous remarks on possible adversarial strategies, the aimed probability is upper bounded by:

$$\max_{I \subseteq \mathbb{F}_q} \frac{|W_I \cap \mathcal{C}|}{|W_I|} .$$

Now remark that $W_I \cap \mathcal{C}$ corresponds to maximum-weight codewords in the code $\mathcal{C}$ punctured on $\mathbb{F}_q \setminus I$. Studying the weight distribution of this punctured code (it is well-known when $\mathcal{C}$ is a Reed-Solomon code, see example 4.6 in [10]) and a careful computation provide the expected result. □

Informally, Proposition IV.2 shows that it is unlikely that adversary $\mathcal{A}$ produces a spurious codeword. The voting phase in the extraction algorithm amplifies this hardness.

Proposition IV.3. *Let $L = \frac{q^m-1}{q-1}$. Then in the extraction algorithm of Fig. 3, we have for all $x \in \mathbb{F}_q^m$,*

$$\mathbb{P}[\texttt{Tab}[x] \text{ is wrong after step } 3] \leq 2^{-L(\gamma(d-1)\log(q-1)-1)} .$$

Proof. Thanks to proposition IV.2, the aimed probability is bounded by the one defined by the binomial distribution with parameters $n = L$ and $p = (q-1)^{-d+1}$. Adapting Lemma 4.13 of [10] on error distributions on codewords, we obtain the following bound:

$$\mathbb{P}[\texttt{Tab}[x] \text{ is wrong after step } 3] \leq 2^{-L \cdot D(\gamma||p)} ,$$

where $D(\gamma||p)$ is the Kullback-Leibler divergence. In our case, a computation gives $D(\gamma||p) > \gamma(d-1)\log(q-1) - 1$ and achieves the proof. □

Union bound implies that the probability there exists a position $x \in \mathbb{F}_q^m$ where $\texttt{Tab}[x]$ is wrong after step 3 is bounded by $2^{-L(\gamma(d-1)\log(q-1)-1)+m\log q}$. Thus, we just bounded the probability of having an error after the third step of the extraction algorithm. Let us now focus on potential erasures.

Lemma IV.4. *Let $\gamma \in [0, 1]$ and $S \subseteq \mathbb{F}_q^m$ with $|S| = k$. We set $L = \frac{q^m-1}{q-1}$. Let $\mathcal{D}$ be a set of lines of $\mathbb{F}_q^m$ such that $\forall s \in S$, $|\{\ell \in \mathcal{D}, s \in \ell\}| \geq (1-\gamma)L$. Then,*

$$|\mathcal{D}| \geq (1-\gamma)kL - \frac{k(k-1)}{2} .$$

Proof. By induction on k. Case $k = 1$ follows from the assumption on $\mathcal{D}$. Assume now the proposition is true for k. For $S \subseteq \mathbb{F}_q^m$, let $\Delta_S = \{\text{lines } \ell \subseteq \mathbb{F}_q^m, |\ell \cap S| \geq (1-\gamma)L\}$. Let $S \subseteq \mathbb{F}_q^m$, $|S| = k+1$, and $y \in S$. We then have:

$$\begin{aligned} |\Delta_S| &= |\Delta_{S\setminus\{y\}}| + |\Delta_{\{y\}}| - |\Delta_{S\setminus\{y\}} \cap \Delta_{\{y\}}| \\ &\geq (1-\gamma)kL - \frac{k(k-1)}{2} + (1-\gamma)L - k \\ &\geq (1-\gamma)(k+1)L - \frac{(k+1)k}{2} . \end{aligned}$$

□

Note that L is exactly the number of lines passing through a point of $\mathbb{F}_q^m$. Hence, there remains an unfilled position (represented by an *erasure* $\perp$) in $s \in \mathbb{F}_q^m$ if and only if at least $(1-\gamma)L$ challenges passing through s have been rejected by the **Verif** procedure.

Let S denote the set of remaining erasures and $\mathcal{D}$ the set of challenges (lines) rejected by **Verif**. We then have $\forall s \in S$, $|\{\ell \in \mathcal{D}, s \in \ell\}| \geq (1-\gamma)L$. Thus, by Lemma IV.4,

$$(1-\epsilon) \underbrace{q^{m-1}L}_{\#\text{ lines } \subset \mathbb{F}_q^m} \geq |\mathcal{D}| \geq (1-\gamma)kL - \frac{k(k-1)}{2} ,$$

with $k = |S|$. It implies, if $\epsilon < (1-\gamma)/2$, that $k < L(d-1)$.

Now, let $s \in S$. Then, there exists a line $\ell_s \subset \mathbb{F}_q^m$ passing through s such that ℓ_s has no more than $d-1$ erasures (otherwise, we refute $k < L(d-1)$), and we can decode the symbol on position s with $\mathcal{C}$'s decoding algorithm which can decode up to $d-1$ erasures.

Bringing together all our results, we have the following theorem:

Theorem IV.5. *Let $\mathcal{P}$ be a $(\mathcal{C}, m, \gamma)$-**LiftPoR** with $\mathcal{C}$ a q-ary Reed-Solomon code of minimum distance d. Then, for all $\epsilon < (1-\gamma)/2$, $\mathcal{P}$ is $(\epsilon, 1-2^{-\lambda})$-sound with $\lambda = L((d-1)\gamma\log(q-1) - 1) - m\log q$.*

Proof. The discussion after Proposition IV.3 implies that with the claimed probability, for any adversary $\mathcal{A}$, there is no error in the temporary file $\texttt{Tab}$ after step 3 of the extraction algorithm. Lemma IV.4 and following discussion show how to deal with potential erasures. □

V. Lifted-code-PoR: theory and practice

A. Theoretical parameters

Fig. 4 gives the exact and asymptotic values of client and server storage overhead as well as the communication complexity of our PoRs.

We also emphasize that our PoR is unbounded-use, as a consequence of its "information-theoretical security" feature. Indeed, challenges are randomly chosen among lines of $\mathbb{F}_q^m$ and uniformly cover the file symbols; thus, the adversary cannot learn anything from past challenges. Fig. 5 asymptotically compares our scheme to Bowers *et. al.* [5] and Shacham-Waters [2] ones.

	Exact value	Asymptotics ($\lvert F\rvert \to \infty$)
C. storage overhead	$\lvert\kappa\rvert$	$O(1)$
S. storage overhead	$\left(\frac{1}{R}-1\right)\lvert F\rvert$	$O(\lvert F\rvert)$
C. $\to$ S. comm.	$2m\log q$	$O(\log\lvert F\rvert)$
S. $\to$ C. comm.	$q\log q$	$O(\lvert F\rvert^{1/m})$

Fig. 4. Our PoR parameters. $|F|$ denotes the file size in bits, q the field size, $m \geq 2$ the lifting parameter, R the lifted code rate and $|\kappa|$ the key size. We also have $Rq^m \log q = |F|$.

Construction	BJO [5]	SW [2]	Our work
Unbounded-use	No (N uses)	Yes	Yes
Client storage	$\lvert\kappa\rvert$	$\lvert F\rvert^{\beta} + \lvert\kappa\rvert$	$\lvert\kappa\rvert$
Server overhead	$\lvert\kappa\rvert N + R^*\lvert F\rvert$	$\frac{\lvert F\rvert^{1-\beta}}{R} + R^*\lvert F\rvert$	$R^*\lvert F\rvert$
Communication	$\lvert\kappa\rvert$	$\lvert F\rvert^{\beta} + \frac{\lvert\kappa\rvert}{R^*}$	$\left(\frac{\lvert F\rvert}{R}\right)^{1/m}$
Remark		$\beta \in]0,1[$	$m \geq 2$

Fig. 5. Asymptotic comparison of some existing PoRs. log terms have been removed for clarity. Notations are identical to those in Fig. 4, except that $R^* = \frac{1}{R} - 1 > 0$ is the redundancy of the code.

B. Practical settings

Lifted codes are known to reach rates close to 1 when their length tends to infinity [4]. However, we need to highlight some good practical codes (not too long but with good rate and known encoding/decoding algorithms) in order to use them in real PoRs.

Recall that we instantiate our base codes with Reed-Solomon codes defined over the whole field $\mathbb{F}_q$. Thus, we have three parameters to deal with: field size q (which is also the base code length), minimum distance $d < q$ and lifting parameter $m \geq 2$. Fig. 6 presents some efficient choices for these parameters. On the one hand, the top grey row shows a very storage-efficient setting with only 3% server storage overhead; on the other hand, the bottom grey row highlights a good low-communication PoR with challenges of size less than $1/10000$ fraction of the file size.

PoR param.			Results				
m	q	d	$\lvert F\rvert$ (bits)	$\frac{1}{R}-1$	comm. C. $\to$ S.	comm. S. $\to$ C.	comm./$\lvert F\rvert$
2	128	2	14197	0.154	28	896	0.0651
	256		58975	0.111	32	2048	0.0352
	512		242461	0.081	36	4608	0.0192
	1024		989527	0.060	40	10240	0.0104
	2048		4017157	0.044	44	22528	0.0056
	4096		16245775	0.033	48	49152	0.0030
	8192		65514541	0.024	52	106496	0.0016
	64	4	2761	0.484	24	384	0.148
	256		53509	0.225	32	2048	0.0389
	1024		940321	0.115	40	10240	0.0109
	4096		15802909	0.062	48	49152	0.0031
	1024	8	887842	0.181	40	10240	0.0116
	4096		15330526	0.094	48	49152	0.0032
	4096	16	14759761	0.137	48	49152	0.0033
	4096	32	14031670	0.196	48	49152	0.0035
3	64	2	118873	1.21	36	384	3.53×10^{-3}
	128		1059339	0.980	42	896	8.85×10^{-4}
	256		9263777	0.811	48	2048	2.26×10^{-4}
	512		79837411	0.681	54	4608	5.84×10^{-5}
4	64	2	2717766	5.173	48	384	1.59×10^{-4}
	128		49578831	4.414	56	896	1.92×10^{-5}

Fig. 6. Some practical settings for our PoR construction.

VI. Conclusion

To sum up, we have defined new unbounded-use Proofs of Retrievability with sublinear communication complexity, constant client storage and low server storage overhead. A first implementation also provides a proof of concept of our PoR; more efficient implementations of lifted codes underlying algorithms could then lead to competitive results. Besides, since the idea to use locally decodable codes in PoRs is quite generic, it would be interesting to design PoRs based on other such codes.

VII. Acknowledgements

The authors would like to thank Daniel Augot for his valuable and helpful comments.

References

[1] A. Juels and B. S. Kaliski, Jr., "PORs: Proofs of Retrievability for Large Files," in *Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28-31, 2007*, P. Ning, S. D. C. di Vimercati, and P. F. Syverson, Eds., 2007, pp. 584–597.

[2] H. Shacham and B. Waters, "Compact proofs of retrievability," in *Advances in Cryptology - ASIACRYPT 2008, 14th International Conference on the Theory and Application of Cryptology and Information Security, Melbourne, Australia, December 7-11, 2008.*, J. Pieprzyk, Ed., 2008, pp. 90–107.

[3] Y. Dodis, S. P. Vadhan, and D. Wichs, "Proofs of Retrievability via Hardness Amplification," in *Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009, San Francisco, CA, USA, March 15-17, 2009. Proceedings*, O. Reingold, Ed., 2009, pp. 109–127.

[4] A. Guo, S. Kopparty, and M. Sudan, "New Affine-Invariant Codes from Lifting," in *Proceedings of Innovations in Theoretical Computer Science, ITCS '13, Berkeley, CA, USA, January 9-12, 2013*, R. D. Kleinberg, Ed., 2013, pp. 529–540.

[5] K. D. Bowers, A. Juels, and A. Oprea, "Proofs of Retrievability: Theory and Implementation," in *Proceedings of the first ACM Cloud Computing Security Workshop, CCSW 2009, Chicago, IL, USA, November 13, 2009*, R. Sion and D. Song, Eds., 2009, pp. 43–54.

[6] S. Kopparty, S. Saraf, and S. Yekhanin, "High-rate codes with sublinear-time decoding," in *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, L. Fortnow and S. P. Vadhan, Eds., 2011, pp. 167–176.

[7] B. Hemenway, R. Ostrovsky, and M. Wootters, "Local correctability of expander codes," in *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part I*, F. V. Fomin, R. Freivalds, M. Z. Kwiatkowska, and D. Peleg, Eds., 2013.

[8] S. Yekhanin, "Locally decodable codes," *Foundations and Trends in Theoretical Computer Science*, vol. 6, no. 3, pp. 139–255, 2012.

[9] B. Schneier, *Applied Cryptography (2nd Ed.): Protocols, Algorithms, and Source Code in C*. John Wiley & Sons, Inc., 1995.

[10] R. M. Roth, *Introduction to coding theory*. Cambridge University Press, 2006.