Efficient Proofs of Retrievability using Expander Codes

Françoise Levy-Dit-Vehel, Maxime Roméas

Efficient Proofs of Retrievability using Expander Codes

Françoise Levy-dit-Vehel[1] and Maxime Roméas[2]

[1] LIX, ENSTA Paris, INRIA, IPP, 91120 Palaiseau, France
levy@ensta.fr
[2] LIX, École polytechnique, INRIA, IPP, 91120 Palaiseau, France
romeas@lix.polytechnique.fr

Abstract. Proofs of Retrievability (PoR) protocols ensure that a client can fully retrieve a large outsourced file from an untrusted server. Good PoRs should have low communication complexity, small storage overhead and clear security guarantees. We design a good PoR based on a family of graph codes called expander codes. We use expander codes based on graphs derived from point-line incidence relations of finite affine planes. Høholdt *et al.* showed that, when using Reed-Solomon codes as inner codes, these codes have good dimension and minimum distance over a relatively small alphabet. Moreover, expander codes possess very efficient unique decoding algorithms. We take advantage of these results to design a PoR scheme that extracts the outsourced file in quasi-linear time and features better concrete parameters than state-of-the-art schemes w.r.t storage overhead and size of the outsourced file.

Keywords: Proofs of Retrievability · Expander Codes · Outsourced Storage

1 Introduction

1.1 Context and state-of-the-art

With the continuous increase in data creation, individuals and business entities call upon remote storage providers to outsource their data. This new dependency raises some issues, as storage providers can try to read or modify the client's data. Besides, when a client does not often access his data, service providers can delete it to make room for another client's data. In this context, it appears important to deploy client side protections designed to bring security guarantees like confidentiality and integrity. In this work, we focus on the following problem: given a client who stored a file on a server and erased its local copy, how can he check if he is able to retrieve his file from the server in full? Addressing this issue is the goal of cryptographic protocols called Proofs of Retrievability (PoRs).

The first PoR scheme was proposed in 2007 by Juels and Kaliski [8] and was based on checking the integrity of some sentinel symbols secretly placed by the client before uploading its file. This scheme has low communication but

its drawback is that it is bounded-use only. Shacham and Waters [15] proposed to correct this drawback by appending some authenticator symbols to the file. Verification consists in checking random linear combinations of file symbols and authenticators. Then comes a few PoR schemes based on codes. Bowers *et al.* [3] proposed a double-layer encoding with the use of an inner code to recover information symbols and an outer code to correct the remaining erasures. Dodis *et al.* [4] formalize the verification process as a request to a code which models the space of possible answers to a challenge. In 2013, Paterson [13] laid the foundation for studying PoR schemes using a coding theoretic framework. Following these ideas, Lavauzelle and Levy-dit-Vehel [9] (2016) used the local structure of the lifted codes introduced by Guo *et al.* [5] to build a PoR scheme, that compares favourably to those presented above w.r.t. storage overhead. In 2022, Levy-dit-Vehel and Roméas [10] proposed a framework for the design of secure and efficient PoR schemes based on Locally Correctable Codes. They also reevaluated the security and the parameters of the [9] PoR scheme.

We design a PoR scheme based on expander codes. In 1996, Sipser and Spielman [16] introduced these codes that are based on expander graphs. Expander codes possess very efficient unique decoding algorithms. We will use an erasure decoding algorithm derived from [16,19] during the extraction phase of our PoR.

The expander codes used for our PoR scheme are based on a family of graphs with excellent expansion. These graphs are derived from point-line incidence relations in the affine plane $\mathbb{F}_q^2$. The expansion of these graphs was studied by Tanner in 1984 [18]. A line of work of Høholdt *et al.* [6,7,2] studied the dimension and minimum distance of expander codes based on the previously mentioned graphs with Reed-Solomon codes as inner code. Finally, we use an audit procedure for generic erasure codes adapted from [8,15] in the Constructive Cryptography (CC) framework of [11] by Badertscher and Maurer [1]. We also prove the security of our PoR scheme using the CC security model for PoRs of [1].

1.2 Contributions

We use an audit procedure from [8,15] translated in the CC framework by Badertscher and Maurer [1] to design a PoR scheme based on expander codes. Recall that an expander code is constructed using a regular expander graph, a so-called inner linear code and, for every vertex v of the graph, an ordering on the edges incident to v. A codeword is a labeling of the edges such that, for every vertex v of the graph, the vector supported by the edges incident to v is a codeword of the inner code. By encoding the client's file with a well-chosen expander code, we manage to design a PoR scheme with storage overhead linear in the size of the outsourced file $|F|$ and communication complexity in $\mathcal{O}(|F|^{1/3} \cdot \log |F| \cdot \sigma)$ for σ bits of statistical security. Furthermore, we give concrete parameters for file sizes ranging from a few MB to hundreds of GB. Our parameters are a lot better than the ones of [9]. When using the same alphabet size and security parameters as other code-based PoRs [8,15,9], our scheme is capable of reaching higher rates and storing larger files. We give parameters and comparisons with other PoRs in sec. 3.4. We optimize the parameters of our scheme using expander

codes based on q-regular graphs derived from point-line incidence relations in finite geometries. The properties of these graphs and of expander codes based on these graphs are studied in [18,6,7,2]. We chose these graphs because they have very good expansion and a good ratio between their regularity q and their number of edges q^3. We show that these two facts when combined permit us to reach lower communication complexity and storage overhead than previous code-based PoRs. Moreover, these graphs exist for every prime power q. By choosing q to be a power of 2 and a Reed-Solomon code of length q as inner code, we can use the erasure decoder for Reed-Solomon codes of [17] with complexity $\mathcal{O}(q \log^2 q)$. Using this decoder along with a fast unique erasure decoding algorithm for expander codes [16,19], we are able to extract the outsourced file in quasi-linear time $\mathcal{O}(q^3 \log^2 q)$ in the input size $Rq^3 \log q$, where $0 < R < 1$ is the rate of the code. For $q = 512$, our PoR stores files of size 124MB vs 35MB for the PoR of [9] with storage overhead of 21% vs 319% for [9].

Organization of the paper. In sec. 2, we give the required background. In sec. 3, we describe our audit procedure and we optimize our PoR. Finally, we compare the performance of our PoR against other schemes in sec. 3.4.

2 Background

2.1 The Constructive Cryptography model

The CC model, introduced by Maurer [12], aims at asserting the real security of cryptographic primitives. To do so, it redefines them in terms of so-called *resources* and *converters*. Starting from a basic resource (e.g. communication channel, shared key, memory server...), a converter (a cryptographic protocol) aims at constructing an enhanced resource, *i.e.* one with better security guarantees. The starting resource, lacking the desired security guarantees, is often called the *real* resource and the obtained one is often called the *ideal* resource, since it does not exist as is in the real world. An example of ideal resource is a confidential server, where the data stored by a client is readable by this client only. The only information that leaks to other parties is its length. This resource does not exist, but it can be emulated by an insecure server on which the client uses an encryption protocol where the encryption scheme is $\mathsf{IND\text{-}CPA}$ secure. We say that this *construction* of the confidential server is secure if the real world - namely, the insecure server together with the protocol - is *just as good* as the ideal world - namely, the confidential server. This means that, whatever the adversary can do in the real world, it could as well do in the ideal world.

We recall the constructions of [1,10] that we will use in this work. The first resource is the authenticated server-memory resource denoted by $\mathbf{aSMR}_{\Sigma,n}$ where Σ is the alphabet and n the memory size. The resource allows the client to read and write data blocks that are encoded as elements of a finite alphabet Σ via its interface C. The interface C_0 is the initialization interface used to set up the initial state of the resource. The server can be "honest but curious" by obtaining the entire history of accesses made by the clients (a log file) and reading their data at interface S_H. The server can also be intrusive by deleting or restoring

previously deleted data using its interface S_I when the resource is set into a special write mode. A deleted data block is indicated by the special symbol ϵ. Thus, if we store a codeword on the **aSMR**, the adversary can only introduce erasures and not errors. We use the **aSMR** specification of [10] because it is tailored for code-based PoRs with its $\mathcal{O}(\log n)$ communication complexity per read query. Indeed, code-based PoRs require a large number of read queries and only one write query to outsource the encoded file. The **aSMR** resource is described in fig. 1 and is constructed in [10] using a simple MAC-based protocol. Each symbol is stored alongside a MAC tag, this yields a storage overhead of κn where κ is the length of a MAC tag.

Resource aSMR$_{\Sigma,n}$

The **aSMR** definition is identical to **SMR** of [1] except for the influence of an adversary at interface S_I and the addition of a version number ctr.

Interface C

Input: $(\texttt{read}, i) \in [1, n]$
  if ACTIVE **and not** INTRUSION **then**
    HIST $\leftarrow$ HIST $||$ $(\texttt{R}, i)$
    return $\mathbb{M}[i]$

Input: $(\texttt{write}, i, x) \in [1, n] \times \Sigma$
  if ACTIVE **and not** INTRUSION **then**
    $ctr \leftarrow ctr + 1$
    HIST $\leftarrow$ HIST $||$ $(\texttt{W}, i, x, ctr)$
    $\mathbb{M}[i] \leftarrow x$

Interface S_I

Input: $(\texttt{delete}, i) \in [1, n]$
  if INTRUSION **then**
    $\mathbb{M}[i] \leftarrow \epsilon$

Input: $(\texttt{restore}, i) \in [1, n]$
  if INTRUSION **then**
    if $\exists k, x$: HIST$[k] = (\texttt{W}, i, x, ctr)$ **then**
      $\mathbb{M}[i] \leftarrow x$

Fig. 1: The authentic SMR of [10] (only interfaces C and S_I are shown)

2.2 Proofs of Retrievability

Proofs of Retrievability (PoR) are cryptographic protocols whose goal is to guarantee that a file stored by a client on a server remains retrievable in full. PoRs thus involve two parties: a client who owns a file F and a server, here modelled as an SMR, on which F is stored. We use the CC based definition of PoR security as presented in [1]. Namely, a PoR scheme is composed of a pair of converters $\mathsf{por} := (\mathsf{por\_init}, \mathsf{por\_audit})$ and works in three phases:

- *An initialization phase.* The client converter init encodes the file F into $\mathbf{Init}(F) = (\tilde{F}, \texttt{data})$. The converter sends $\texttt{data}$ (*e.g.* keys, etc.) to the client, then it sends $\tilde{F}$ to the SMR with a $\texttt{write}$ query and erases F from the client's memory.
- *An audit phase.* The client converter audit probes some symbols of the server's memory and outputs $\texttt{accept}$ if it believes that the file is retrievable in full and $\texttt{reject}$ otherwise.

- *An extraction phase.* If the client has been convinced by the audit phase, he can send `read` to recover his whole file with high probability.

A PoR scheme is considered secure if it constructs an ideal abstraction of a PoR (introduced in [1]). This abstraction consists of an ideal SMR $\mathbf{aSMR}^{\mathsf{audit}}_{\Sigma,n}$ that considers the client's interface augmented with an `audit` mechanism. On an `audit` request, the resource checks whether the current memory content is indeed the newest version that the client wrote to the storage. If a single data block has changed, the ideal audit will detect this and output `reject` to the client. In case of a successful audit (returning `accept`), this guarantee holds until the server gains write-access to the storage, in which case a new audit has to reveal whether modifications have been made.

2.3 Expander graphs and expander codes

We recall the definitions and well known properties of expander graphs and expander codes. We follow the presentation of [14]. Let $G := (V, E)$ be an undirected d-regular graph on n vertices. The *expansion* of G is $\lambda := \max\{\lambda_2, |\lambda_n|\}$, where $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$ are the eigenvalues of the adjacency matrix of G. We say that G is a *Ramanujan* graph if $\lambda \leq 2\sqrt{d-1}$. For a vertex $v \in V$, let $\Gamma(v)$ be the set of vertices adjacent to v. Let $\mathcal{C}_0 \subseteq \mathbb{F}_q^d$ be a linear code, called the *inner code*. Fix an order on the edges incident to each vertex of G, and let $\Gamma_i(v)$ be the i-th neighbor of v. Using the graph G and the inner code $\mathcal{C}_0$ we can construct a new code, called an *expander code*. The expander code $\mathcal{C} := \mathcal{C}(G, \mathcal{C}_0)$ is defined as the set of all labelings of the edges of G that respect the inner code $\mathcal{C}_0$. It has length $nd/2$. More precisely, we have the following definition.

Definition 1 (Expander Code). *Let $\mathcal{C}_0 \subseteq \mathbb{F}_q^d$ be a linear code, and let $G = (V, E)$ be a d-regular expander graph on n vertices. The expander code $\mathcal{C}(G, \mathcal{C}_0) \subseteq \mathbb{F}_q^E$ is a linear code of length $nd/2$, so that for $c \in \mathbb{F}_q^E$, $c \in \mathcal{C}$ if and only if, for all $v \in V$, $(c_{(v,\Gamma_1(v))}, \ldots, c_{(v,\Gamma_d(v))}) \in \mathcal{C}_0$.*

If $\mathcal{C}_0$ is a linear code of rate R_0, then $\mathcal{C}(G, \mathcal{C}_0)$ is a linear code of rate at least $2R_0 - 1$. We say that an undirected graph $G = (L \cup R, E)$ is bipartite if, for all vertices $v \in L$, we have $\Gamma(v) \cap L = \emptyset$ and, for all vertices $v \in R$, we have $\Gamma(v) \cap R = \emptyset$. It is known that expander codes constructed from bipartite graphs have good distance [16,19]:

Proposition 1. *Let $\mathcal{C}_0 \subseteq \mathbb{F}_q^d$ be a linear code with relative distance δ, and let $G = (L \cup R, E)$ be a d-regular bipartite expander graph with expansion λ. Then the expander code $\mathcal{C}(G, \mathcal{C}_0)$ has distance at least $\delta(\delta - \lambda/d)$.*

Moreover, $\mathcal{C}$ can be efficiently decoded up to this fraction of erasures [16,19].

Proposition 2. *Let $\mathcal{C}_0 \subseteq \mathbb{F}_q^d$ be a linear code with relative distance δ. Let $\mathcal{D}(d)$ be the time needed to uniquely decode $\mathcal{C}_0$ from $\delta - 1/d$ erasures. Let $G = (L \cup R, E)$ be a d-regular bipartite expander graph on n vertices with expansion λ. Let $\epsilon > 0$ and suppose that $\frac{\lambda}{d} < \frac{\delta}{2}$. Then, the decoder of [14] uniquely decodes the expander code $\mathcal{C}(G, \mathcal{C}_0)$ from up to $(1 - \epsilon)\delta(\delta - \lambda/d)$ erasures in time $n \cdot \mathcal{D}(d)/\epsilon$.*

3 PoR with expander codes

3.1 Audit

Our scheme will use a generic audit for erasure codes presented in the CC framework by Badertscher and Maurer in [1]. We describe how [1] implemented the ideas of [8,15] to construct an $\mathbf{aSMR}^{\mathsf{audit}}_{\Sigma^k,1}$ from an $\mathbf{aSMR}_{\Sigma,n}$. Let $(\mathsf{enc}, \mathsf{dec})$ be an (n,k,d) erasure code and $F \in \Sigma^k$ be the client's file. We describe the PoR scheme $\mathsf{ecPor} := (\mathsf{ecInit}, \mathsf{ecAudit})$ for erasure codes. On input `init` to ecInit, the converter sends `init` to $\mathbf{aSMR}_{\Sigma,n}$ and computes the encoding $\bar{F} := \mathsf{enc}(F) \in \Sigma^n$. Then, for all $i \in [n]$, the converter sends $(\mathtt{write}, i, \bar{F}_i)$ to $\mathbf{aSMR}_{\Sigma,n}$.

On input $(\mathtt{read})$ to either ecInit or $\mathsf{ecAudit}$, the converter retrieves the whole memory content via $(\mathtt{read}, i)$ requests and obtains for each location, either a symbol $v_i \in \Sigma$ or the erasure symbol $\perp$. If v_i is returned, set $W_i := v_i$, else set $W_i := \perp$. If $|\{i \in [n] \mid W_i = \perp\}| > d-1$, the converter outputs ϵ at its outside interface, otherwise it computes $F := \mathsf{dec}(W)$, and outputs F.

Finally, on a query `audit`, the converter $\mathsf{ecAudit}$ chooses a random subset $S \subseteq [n]$ of size t and outputs $(\mathtt{read}, i)$ to $\mathbf{aSMR}$ for each $i \in S$. If all read instructions for $i \in S$ returned a non-erased symbol, the converter outputs `accept`. Otherwise, it outputs `reject`. The integer t is chosen according to the security level we want to achieve. The security of the scheme is given by:

Theorem 1 ([1]). *Let $n, k, d \in \mathbb{N}$. Let $(\mathsf{enc}, \mathsf{dec})$ be an (n,k,d) erasure code for alphabet Σ and erasure symbol $\perp$. Let $\rho := 1 - \frac{d-1}{n}$ be the minimum fraction of symbols needed to recover the file. Then, the above protocol $\mathsf{ecPor} := (\mathsf{ecInit}, \mathsf{ecAudit})$ that chooses a random subset of size t during the audit, constructs the $\mathbf{aSMR}^{\mathsf{audit}}_{\Sigma^k,1}$ from the $\mathbf{aSMR}_{\Sigma,n}$. More specifically, there exists a simulator sim such that for all distinguishers $\mathbf{D}$ performing at most q audits,*

$$\Delta^{\mathbf{D}}(\mathsf{ecInit}_{\mathsf{C}_0}\mathsf{ecAudit}_{\mathsf{C}}\,\mathbf{aSMR}_{\Sigma,n}, \mathsf{sim}^{\mathsf{S}}\,\mathbf{aSMR}^{\mathtt{audit}}_{\Sigma^k,1}) \leq q \cdot \rho^t$$

3.2 Description of our PoR with expander codes: the general case

Let $\mathcal{C}_0$ be a linear code of length d, relative distance δ_0 and rate R_0. Using the Singleton bound, we have $\delta_0 \leq 1 + \frac{1}{d} - R_0$. Let G be a d-regular bipartite graph on n vertices with expansion λ. We instantiate the PoR scheme $\mathsf{ecPor} := (\mathsf{ecInit}, \mathsf{ecAudit})$ with the expander code $\mathcal{C}(G, \mathcal{C}_0)$.

In the following, we determine the number t of edges probed during the audit needed to reach a given security level. If we suppose that $\frac{\lambda}{d} < \frac{\delta_0}{2}$, using the Singleton bound, we must have $R_0 < 1 + \frac{1}{d} - \frac{2\lambda}{d}$. Moreover, if $\mathcal{C}_0$ is Maximum Distance Separable, this implication becomes an equivalence. This is why, from now on, we will suppose that the inner code $\mathcal{C}_0$ is MDS. We take G to be a bipartite expander graph with expansion λ such that $\frac{\lambda}{d} < \frac{\delta_0}{2}$. Using prop. 1, the minimum distance $\delta_{\mathcal{C}}$ of $\mathcal{C}(G, \mathcal{C}_0)$ is at least $\delta_0(\delta_0 - \lambda/d) > 2\lambda^2/d^2$.

Let $\epsilon > 0$. If we want to correct a $(1-\epsilon)\delta_{\mathcal{C}}$ fraction of erasures, the minimum fraction of valid edges needed to recover our file is

$$\rho = 1 + \frac{1}{nd} - (1-\epsilon)\delta_{\mathcal{C}} \leq 1 + \frac{1}{nd} - (1-\epsilon)\frac{2\lambda^2}{d^2}$$

Let σ be a statistical security parameter and t be the number of edges probed during the audit. Our scheme is considered secure if $\rho^t \leq 2^{-\sigma}$. We want to choose t such that $t \geq -\sigma/\log\rho$. *Approximation*: If $\frac{1}{nd} - (1-\epsilon)\frac{2\lambda^2}{d^2} \approx 0$, we have

$$\frac{-\sigma}{\log\rho} \approx \frac{nd^2\sigma}{2(1-\epsilon)n\lambda^2 - d} = \frac{d^2\sigma}{2(1-\epsilon)\lambda^2 - \frac{d}{n}}$$

Moreover, if G is Ramanujan, we have $\lambda \leq 2\sqrt{d-1}$ and $\frac{-\sigma}{\log\rho} \approx \frac{d\sigma}{8(1-\epsilon)}$. If G has expansion $\sqrt{d}$ instead, we have $\frac{-\sigma}{\log\rho} \approx \frac{d\sigma}{2(1-\epsilon)}$.

Note that our scheme requires the adversary to only introduce erasures (and not errors). We enforce this using an **aSMR**. After a successful audit, the client can extract its file by running the decoder of prop. 2 which runs in time $\mathcal{O}(n \cdot \mathcal{D}(d)/\epsilon)$, where $\mathcal{D}(d)$ is the complexity of $\mathcal{C}_0$'s decoder.

3.3 Instantiation with the point-line incidence graph of the plane

Let Γ be the point-line incidence graph of the affine plane over $\mathbb{F}_q$ without the vertical lines. This graph is q-regular, has $2q^2$ vertices and expansion $\sqrt{q}$ (see the work of Tanner [18]). We have $\Gamma := (V_1 \cup V_2, E)$ where

$$V_1 := \{(x,y) \mid x,y \in \mathbb{F}_q\}, \quad V_2 := \{(a,b) \mid a,b \in \mathbb{F}_q\}, \text{ and}$$
$$E := \{((x,y),(a,b)) \mid (x,y) \in V_1, (a,b) \in V_2, ax + b - y = 0\}$$

This graph is an excellent choice for our PoR scheme. Recall that the rate of the inner code is upper bounded by $1 + \frac{1}{d} - \frac{2\lambda}{d}$ and the rate of the expander code is lower bounded by $2R_0 - 1$. The graph Γ also has a nice ratio between its regularity q and its number of edges q^3. Since we need to probe a number of edges linear in q, this ensures that our PoR scheme has communication complexity of order cubic root of the size of the outsourced file. This is in line with or even better than other code-based PoR schemes, such as [9] (which has communication complexity of order square root of the file size for $m = 2$). Our inner code $\mathcal{C}_0$ will be a Reed-Solomon code of rate $R_0 < 1 + \frac{1}{d} - \frac{2\lambda}{d}$. This code is MDS, and thus we can use the decoder of prop. 2 for our extraction phase. Moreover, because our inner code is a Reed-Solomon code, we can use the following result of Beelen *et al.* [2]. Let $\mathbb{F}_q := \{\alpha_1, \alpha_2, \ldots, \alpha_q\}$. We use the following labeling (of [2]) for the edges of Γ : if $(x,y) \in V_1$, $\Phi_{(x,y)}(i) := (x, y, \alpha_i, y - x\alpha_i)$ and, if $(a,b) \in V_2$, $\Phi_{(a,b)}(i) := (\alpha_i, a\alpha_i + b, a, b)$. When q is a power of 2 or a prime, Beelen *et al.* [2] showed that when using this labeling on the graph Γ with a Reed-Solomon code of rate $1/2 < R_0 \leq 1$ as inner code, we obtain an expander code of rate *exactly* $R := R_0^3 + R_0(1-R_0)(2R_0-1)$.

3.4 Parameters

Let σ be the statistical security parameter ($\sigma = 40$) and κ be the computational security parameter[3] ($\kappa = 128$). Set q, a power of 2. Let G be the q-regular point-line incidence graph over $\mathbb{F}_q^2$. This graph has $2q^2$ vertices, q^3 edges and expansion

[3] of the MAC used to construct the **aSMR**

$\lambda := \sqrt{q}$. Let the inner code $\mathcal{C}_0$ be a Reed-Solomon code of length q and rate $R_0 = \max\{\frac{k}{q} \mid k \in \mathbb{N} \text{ and } \frac{k}{q} < 1 + \frac{1}{q} - \frac{2\lambda}{q}\}$. We take R_0 to be as big as possible to reduce the storage overhead of the PoR while still having a quasi-linear time decoder for the expander code. Indeed, since q is a power of 2, $\mathcal{C}_0$ can be erasure decoded in time $\mathcal{O}(q \log^2 q)$ thanks to the decoder of Tang and Lin [17].

Our expander code $\mathcal{C}(G, \mathcal{C}_0)$ has length q^3, rate $R := R_0^3 + R_0(1-R_0)(2R_0-1)$ and alphabet $\mathbb{F}_q$. Let $|F|$ be the size of the outsourced file in bits. It is such that $|F| = Rq^3 \log(q)$. Using prop. 2, we get a decoder for $\mathcal{C}(G, \mathcal{C}_0)$ (and thus an extraction phase) running in time $\mathcal{O}(2q^3 \log^2 q)$ which is quasi-linear in the input size $q^3 \log q$. The storage overhead is given by $1/R - 1$, which is the redundancy of the code. The parameters of our PoR and their asymptotic behavior are given in table 1. Even though our PoR has the same asymptotic behavior than the PoR of [9], we show in table 2 that we get much better parameters in practice.

	Exact value	Asymptotics ($\lvert F\rvert \to \infty$)
C. storage overhead	κ	$\mathcal{O}(1)$
S. storage overhead	$(\frac{1}{R} - 1)\lvert F\rvert + q^3\kappa$	$\mathcal{O}(\lvert F\rvert)$
comm. C. $\to$ S.	$\frac{q\sigma}{2} \log(q^3)$	$\mathcal{O}(\lvert F\rvert^{\frac{1}{3}} \log \lvert F\rvert)$
comm. S. $\to$ C.	$\frac{q\sigma}{2}(\kappa + \log q)$	$\mathcal{O}(\lvert F\rvert^{\frac{1}{3}} \log \lvert F\rvert)$

Table 1: The parameters of our scheme when using the point-line incidence graph over $\mathbb{F}_q^2$ and a Reed-Solomon code as inner code.

In table 2, we give concrete parameters of our PoR scheme for different values of q. We compare our scheme with the PoR of [9]. For $q = 512$, [9, Fig. 6] and its new security analysis by [10] gives a PoR with codewords of length q^3 that stores 35.9MB files with storage overhead of 319%. In table 2, we see that for $q = 512$ and codeword length q^3, our PoR stores 124MB files with storage overhead of 21%, the same communication complexity as [9] and a quasi-linear time extraction phase. Do note that using any Ramanujan graph instead of the point-line incidence graph yields substantially worse parameters.

q	R_0	R	$2q^2$	$\lvert F\rvert$	$\frac{1}{R} - 1$	comm./$\lvert F\rvert$
256	0.878	0.758	131,072	12MB	0.320	2×10^{-4}
512	0.913	0.827	524,288	124MB	0.210	6×10^{-5}
1024	0.938	0.876	2,097,152	1.176GB	0.141	1×10^{-5}
2048	0.956	0.912	8,388,608	10.772GB	0.096	3×10^{-6}
4096	0.968	0.936	33,554,432	96.485GB	0.068	8×10^{-7}
8192	0.978	0.956	134,217,728	854.055GB	0.046	2×10^{-7}

Table 2: Effective parameters of our PoR using the point-line incidence graph over $\mathbb{F}_q^2$ for different values of q and Reed-Solomon codes as inner code. The graph is q-regular with $2q^2$ vertices. We choose the largest possible rate yielding a quasi-linear time decoder. The statistical security parameter is 40.

References

1. Badertscher, C., Maurer, U.: Composable and robust outsourced storage. In: Topics in Cryptology - CT-RSA 2018 - The Cryptographers' Track at the RSA Conference 2018, San Francisco, CA, USA, April 16-20, 2018, Proceedings. pp. 354–373 (2018). https://doi.org/10.1007/978-3-319-76953-0_19
2. Beelen, P., Høholdt, T., Piñero, F., Justesen, J.: On the dimension of graph codes with reed-solomon component codes. In: 2013 IEEE International Symposium on Information Theory. pp. 1227–1231 (2013). https://doi.org/10.1109/ISIT.2013.6620422
3. Bowers, K.D., Juels, A., Oprea, A.: Proofs of retrievability: Theory and implementation. In: Proceedings of the 2009 ACM Workshop on Cloud Computing Security. pp. 43–54. CCSW '09, ACM, New York, NY, USA (2009). https://doi.org/10.1145/1655008.1655015
4. Dodis, Y., Vadhan, S., Wichs, D.: Proofs of retrievability via hardness amplification. In: Proceedings of the 6th Theory of Cryptography Conference on Theory of Cryptography. pp. 109–127. TCC '09, Springer-Verlag, Berlin, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00457-5_8
5. Guo, A., Kopparty, S., Sudan, M.: New affine-invariant codes from lifting. In: Proceedings of the 4th Conference on Innovations in Theoretical Computer Science. pp. 529–540. ITCS '13, ACM, New York, NY, USA (2013). https://doi.org/10.1145/2422436.2422494
6. Høholdt, T., Justesen, J.: Graph codes with reed-solomon component codes. In: 2006 IEEE International Symposium on Information Theory. pp. 2022–2026 (2006). https://doi.org/10.1109/ISIT.2006.261904
7. Høholdt, T., Justesen, J.: The minimum distance of graph codes. In: Proceedings of the Third International Conference on Coding and Cryptology. p. 201âĂŞ212. IWCC'11, Springer-Verlag, Berlin, Heidelberg (2011)
8. Juels, A., Kaliski, Jr., B.S.: Pors: Proofs of retrievability for large files. In: Proceedings of the 14th ACM Conference on Computer and Communications Security. pp. 584–597. CCS '07, ACM, New York, NY, USA (2007). https://doi.org/10.1145/1315245.1315317
9. Lavauzelle, J., Levy-Dit-Vehel, F.: New proofs of retrievability using locally decodable codes. In: International Symposium on Information Theory ISIT 2016. pp. 1809 – 1813. Barcelona, Spain (2016). https://doi.org/10.1109/ISIT.2016.7541611
10. Levy-Dit-Vehel, F., Roméas, M.: A framework for the design of secure and efficient proofs of retrievability. Cryptology ePrint Archive, Report 2022/064 (2022), https://ia.cr/2022/064
11. Maurer, U.: Constructive Cryptography - A New Paradigm for Security Definitions and Proofs, pp. 33–56. Theory of Security and Applications, Springer Berlin Heidelberg (2012)
12. Maurer, U., Renner, R.: Abstract cryptography. In: In Innovations In Computer Science. Tsinghua University Press (2011)
13. Paterson, M., Stinson, D., Upadhyay, J.: A coding theory foundation for the analysis of general unconditionally secure proof-of-retrievability schemes for cloud storage. Journal of Mathematical Cryptology **7**(3), 183–216 (2013). https://doi.org/doi:10.1515/jmc-2013-5002, https://doi.org/10.1515/jmc-2013-5002
14. Ron-Zewi, N., Wootters, M., Zémor, G.: Linear-time erasure list-decoding of expander codes. In: 2020 IEEE International Symposium on Information Theory (ISIT). pp. 379–383 (2020). https://doi.org/10.1109/ISIT44484.2020.9174325

15. Shacham, H., Waters, B.: Compact proofs of retrievability. In: Pieprzyk, J. (ed.) Advances in Cryptology - ASIACRYPT 2008. pp. 90–107. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)
16. Sipser, M., Spielman, D.: Expander codes. IEEE Transactions on Information Theory **42**(6), 1710–1722 (1996). https://doi.org/10.1109/18.556667
17. Tang, N., Lin, Y.: Fast encoding and decoding algorithms for arbitrary (n, k) Reed-Solomon codes over $\mathbb{F}_{2^m}$. IEEE Communications Letters **24**(4), 716–719 (2020). https://doi.org/10.1109/LCOMM.2020.2965453
18. Tanner, R.M.: Explicit concentrators from generalized n-gons. SIAM Journal on Algebraic Discrete Methods **5**(3), 287–293 (1984). https://doi.org/10.1137/0605030, https://doi.org/10.1137/0605030
19. Zémor, G.: On expander codes. IEEE Transactions on Information Theory **47**(2), 835–837 (2001), https://doi.org/10.1109/18.910593