Exploiting Rateless Codes in Cloud Storage Systems

Cosimo Anglano, Rossano Gaeta, and Marco Grangetto, *Senior Member, IEEE*

Abstract—Block-Level Cloud Storage (BLCS) offers to users and applications the access to persistent block storage devices (virtual disks) that can be directly accessed and used as if they were raw physical disks. In this paper we devise ENIGMA, an architecture for the back-end of BLCS systems able to provide adequate levels of access and transfer performance, availability, integrity, and confidentiality, for the data it stores. ENIGMA exploits LT rateless codes to store fragments of sectors on storage nodes organized in clusters.
We quantitatively evaluate how the various ENIGMA system parameters affect the performance, availability, integrity, and confidentiality of virtual disks. These evaluations are carried out by using both analytical modeling (for availability, integrity, and confidentiality) and discrete event simulation (for performance), and by considering a set of realistic operational scenarios. Our results indicate that it is possible to simultaneously achieve all the objectives set forth for BLCS systems by using ENIGMA, and that a careful choice of the various system parameters is crucial to achieve a good compromise among them. Moreover, they also show that LT coding-based BLCS systems outperform traditional BLCS systems in all the aspects mentioned before.

Index Terms—Cloud storage, rateless codes, availability, confidentiality, integrity, performance.

1 Introduction

Block-Level Cloud Storage (BLCS) [1] is a storage paradigm offering to users and applications the access to persistent block storage devices (named *virtual disks (VD)*) that can be directly accessed and used as if they were raw physical disks. BLCS systems offer various advantages over alternative data storage solutions (e.g. distributed file systems), namely: (a) they support cloud applications that require access to persistent raw block devices (e.g., DBMSes), (b) they are natively supported by any off-the-shelf operating system, and (c) the optimizations they provide are transparently available to any off-the-shelf file system used to format them.

In order to be usable in practical settings, a BLCS must provide adequate levels of *performance* (i.e. low access times, high transfer throughput and number of I/O operations per second), *availability* (i.e., the probability that each sector is available when it is requested), *integrity* (i.e., the ability of ensuring data trustworthiness [2]), and *confidentiality* (i.e. the ability of concealing data to unauthorized users [2]).

The typical architecture of a BLCS system features a *front-end*, providing users/applications with a set of abstract disk operations, and a *back-end*, that implements these operations over a set of physical storage resources. Traditional back-ends are typically implemented by aggregating a pool of *volume servers*, and by provisioning each VD on a single server.

• *Rossano Gaeta and Marco Grangetto are with Università degli Studi di Torino, Dipartimento di Informatica, Torino, Italia. Cosimo Anglano is with Università degli Studi del Piemonte Orientale, DiSIT-Computer Science Institute, Alessandria, Italia.*
E-mail: {first.last}@di.unito.it {first.last}@di.unipmn.it

This solution, however, is affected by performance and availability problems, since storage servers are a single point of failure, and may easily become a bottleneck.

In this paper we investigate how a family of codes, known as *Luby Transform (LT)* codes [3], can be used as the enabling technology for BLCS systems able to solve the problems mentioned above. In coding-based storage system, the data units are first split into k equal size *information fragments*. These are subsequently encoded into n equally sized *coded fragments* ($n \geq k$) such that a suitably-chosen subset of them suffices to reconstruct the sector. Finally, the coded fragments are spread across a set of independent physical nodes.

LT codes are *rateless*, that is the ratio k/n (the *rate* of the code) is not fixed at design-time, but can be instead adjusted at run time. Compared to alternative coding schemes (e.g., Reed-Solomon [4]) LT codes are better suited to the needs of a BLCS system because of their much lower complexity, and their ability to adjust redundancy at run time.

Thanks to these properties, the usage of LT codes in the back-end of a BLCS system provides many benefits. In particular, in this paper we show that:

- low sector access time and high transfer throughput can be achieved by exploiting the simultaneous fetch of sector fragments from independent storage resources, thus exploiting the availability of many independent network paths;
- there is no single point of failure, as the coded fragments are stored on various independent storage resources, and lost fragments (caused by a failed resource) can be regenerated on-the-fly without the constraint of recreating exactly the lost data;
- suitable levels of confidentiality can be achieved by keeping secret the random generation process

and the addresses of the storage resources where fragments are stored.

We carry out our investigation by discussing the design of ENIGMA, a distributed storage infrastructure based on LT codes, that encompasses various mechanisms enabling it to obtain suitable levels of performance, availability, integrity, and confidentiality and that, as such, can be effectively used as the back-end of a BLCS system.

Using coding for cloud storage is not a new idea [5], and several works have studied in depth solutions to achieve each one of the goals of BLCS systems (i.e., adequate performance, availability, integrity, and confidentiality). However, to the best of our knowledge, none of the existing works attempts to achieve all these goals at the same time.

Simultaneously achieving all these goals, however, is not trivial, as they typically conflict with each other. For instance, as shown later, higher levels of availability require larger values of n, while higher levels of performance require lower values of n. Therefore, several trade-offs emerge when all these objectives must be simultaneously achieved.

To understand and evaluate these trade-offs, we quantitatively evaluate how the system parameters of ENIGMA affect the performance, availability, integrity and confidentiality of virtual disks. These evaluations are carried out by using both analytical modeling (for availability, integrity, and confidentiality) and discrete-event simulation (for performance), and by considering a set of realistic operational scenarios. Our results indicate that it is possible to simultaneously achieve all the objectives set forth for BLCS systems using ENIGMA, and that a careful choice of the various system parameters is crucial to achieve a good compromise among them. Moreover, they also show that coding-based BLCS systems outperform traditional BLCS systems in all the aspects mentioned before.

Our contributions

To summarize, the contributions of this paper [1] are as follows:

1) we devise an architecture of a back-end for a BLCS system that is able to achieve suitable levels of performance, availability, integrity and confidentiality thanks to the efficient use of LT codes;
2) we evaluate the performance of this system with realistic workloads (consisting of real disk access traces collected on production, enterprise-level storage system), and we compare them against a realistic model of a traditional BLCS system;
3) we devise various analytical models of availability and integrity of virtual disks.

This paper is organized as follows. In Sec. 2 we describe related works, while in Sec. 3 we present the encoding method we use in ENIGMA. In Sec. 4 we present the architecture and the operation of ENIGMA. In Sec. 5 we devise suitable analytical models that express availability, integrity, and confidentiality in terms of its main design parameters, and we use them to study how the various system parameters affect the availability, integrity, and confidentiality of virtual disks. In Sec. 6 we study their performance by means of discrete-event simulation. Finally, in Sec. 7 we conclude the paper and outline future work.

1. A preliminary version of ENIGMA has been presented in [6].

2 Related work

Block-level cloud storage systems have been actively investigated in the recent past, and various solutions have been proposed (e.g., Amazon EBS [7], Eucalyptus Block-Based Storage Abstraction [8], Open Nebula [9], Virtual Block Store [10], VBS-Lustre [11],and Orthus [12]). The back-ends of these systems, however, is based on volume servers, and therefore suffer from the performance and availability problems mentioned in the Introduction. In contrast, as discussed in subsequent sections of this paper, ENIGMA is able to suitably address the issues affecting the above systems

Furthermore, the systems mentioned above do not adopt an holistic approach to the simultaneous provision of adequate levels of data availability, integrity, and confidentiality. This implies that exogenous mechanisms must be adopted, but these solutions are often satisfactory only in part. For instance, whole-sector replication provides much lower availability levels than coding-based solutions at a much higher storage cost. Furthermore, state-of-the-art confidentiality mechanisms for cloud storage are based on encryption [13], [14], [15] that are computationally expensive (and, thus, are rarely used in real-world settings [16]), while the solution provided by ENIGMA incurs in a much lower computational cost. Finally, state-of-the-art integrity mechanisms focus on just *checking* the integrity of data, while ENIGMA is able to *tolerate* the modification of sector fragments.

The usage of coding techniques for distributed storage has been already explored in the literature (see [5] for a survey). The emphasis of these works, however, is placed on the problem of reconstructing fragments stored on nodes that permanently leave the storage infrastructure.

In [17] rateless codes have been used to devise the file based cloud storage system (as opposed to our BLCS). This work states that LT codes can be exploited to achieve high availability and security and mainly deals with data integrity and data repair. Moreover, they propose to use multiple LT encoding and decoding checks to avoid LT decoding failures. This process involves only coding vectors and represents a one-time preprocess whose result could be reused. Nevertheless, the number of required encoding and decoding checks is equal to $\binom{n}{k}$ therefore the data outsourcing operation may become rather complex as the values of n and k increase.

3 LT CODES FOR CLOUD STORAGE

3.1 Preliminaries

As mentioned in the introduction, in this paper we will rely on Luby Transform (LT) codes [3] for sector encoding. LT codes are a family of asymptotically optimal rateless codes over the binary Galois field GF(2), i.e. the coding process is based on the simple exclusive or (XOR) operation. As opposed to traditional erasure codes, it is not required to fix the rate a priori. Given a chunk of data, these are fragmented into k symbols (or blocks) of equal length. The symbol length can be fixed as desired, e.g. compliant with the transport payload being used for transport, real-time constraints, etc., and does not impact on the theoretical features of the code. The encoding process works in 3 steps:

- randomly choose the degree d of the encoding symbol according to the so called degree distribution;
- uniformly select at random d distinct input symbols
- compute the encoded symbol as the XOR of the d selected symbols.

The encoding process is rateless in that an arbitrary sequence of coded symbols can be randomly constructed. The coding process is clearly driven by a pseudo-random generator initialized with a given seed. The seed can be used also on the receiver side as a mean to repeat the generation of the degree d and of the selected input symbols in order to run the decoding algorithm. The decoding performance of rateless codes is evaluated in terms of decoding overhead, i.e. the amount of coded symbols required in excess of k to decode the input symbols. LT codes have been shown to be *asymptotically optimal* for large block size k. Such performance is a direct consequence of the used degree distribution, known as Robust Soliton distribution (RSD), that achieves both simple decoding and vanishing overhead. The RSD depends on k and two additional parameters, usually denoted as c and δ (see [3] for their precise definition), that we omit from discussion for conciseness. In the context of this paper the coded symbols will be stored and retrieved in groups of x, parameter that we term *grouping factor*. Moreover, the maximum number of coded symbols to be stored is fixed to the value of $n \geq k$. In our settings the LT decoding performance can be analyzed introducing the decoding distribution $\varepsilon_{n,k,x}(i)$, with $i = k, k+1, \ldots$, that represents the probability to accomplish decoding from exactly i coded symbols, when n symbols are stored in group of x. One can also introduce the cumulative decoding distribution $\sigma_{n,k,x}(i) = \sum_{j=k}^{i} \varepsilon_{n,k,x}(j)$ that is the probability of decoding with less or equal than i coded symbols. In the rest of this paper we define the *average decoding overhead* as $\bar{\epsilon}_{n,k,x} = \frac{1}{k}\sum_{i=k}^{\infty}(i-k)\varepsilon_{n,k,x}(i)$, i.e., as the average number of extra fragments required to decode the input symbols over k. The previous distributions and the average overhead depend on the value of k, n, x and on the two RSD parameters c and δ. The asymptotic property of LT codes guarantees that $\lim_{k\to\infty} \bar{\epsilon}_{n,k,x} = 0$.

Finally, it is worth noting that if one limits the maximum number of LT coded symbols to be stored to a fixed value n, this implies that decoding can fail with a probability $p_f(n) = \sum_{j=n+1}^{\infty} \varepsilon_{n,k,x}(j) = 1 - \sigma_{n,k,x}(n)$. This is due to the fact that LT coded symbols are generated randomly and decoding is guaranteed only in probability. A countermeasure to this event is proposed in Sec. 3.2, where a modified encoder yielding $p_f(n) = 0$ is presented.

It is worth noticing that LT codes provide several advantages over other rateless schemes, e.g. Raptor codes [18], namely: (a) they are non systematic and as such no original sector fragments are disclosed to the storage resources, hence they guarantee higher confidentiality levels; (b) they permit simpler repair protocols [19]; (c) the modified version of the encoder we propose (see next section) has similar overhead and computational cost for small values of k.

3.2 ENIGMA LT coding strategy

The practical application of LT codes to ENIGMA clearly requires to choose suitable values for k and n. If one wishes to keep the coding overhead low, then k must be chosen as large as possible. Unfortunately, in ENIGMA large values of k are undesirable since: (a) they adversely affect the network communication overhead, given the constraints on reasonable sector sizes (see definition of p_{size} in Sect. 4), (b) they yield higher encoding and decoding computational costs.

Hence, we are forced to use limited values of k (and n) that not only make the coding overhead larger, but also (and even worse) yield a significantly higher decoding failure probability $p_f(n)$.

To deal with the above issues we enhance the standard LT encoding and decoding mechanisms to achieve affordable coding overhead and zero decoding failure probability under typical storage settings, i.e. allowed sector size. LT codes have been originally proposed as asymptotically optimal codes with a simple Belief Propagation (BP) decoder [3], i.e. based on the solution of the underlying linear system with recursive cancellation of the degree 1 equations. For small values of k both the average decoding overhead and the decoding failure probability, make BP decoder useless in our scenario. This is shown in Tab. 1 where $\bar{\epsilon}_{n,k,x}$ and $p_f(2k)$ have been experimentally estimated in the case of LT codes with RSD parameters $c = 0.05$, $\delta = 0.01$ for $k \leq 48$: an average overhead $\bar{\epsilon}_{n,k,x}$ larger than 0.6, i.e. 60% of k, clearly rules out the BP decoder in our setting.

Therefore we resort to another decoder, known as On-the-Fly Gaussian Elimination (OFG) [20], that is able to improve the decoding performance for limited value of k without impacting dramatically in terms of computational cost. Indeed, in Tab. 1 it can be shown that OFG yields an average decoding overhead of less than 0.05. Unfortunately the standard LT encoding process still does not allow one to exclude the decoding failure event, even if its probability vanishes increasing k.

TABLE 1
Decoding overhead $\bar{\epsilon}_{n,k,x}$ of LT codes with belief propagation decoder (BP), OFG decoder, and ENIGMA LT codes.

k	LT w. BP		LT w. OFG		LT ENIGMA	
	$\bar{\epsilon}_{n,k,x}$	$p_f(2k)$	$\bar{\epsilon}_{n,k,x}$	$p_f(2k)$	$\bar{\epsilon}_{n,k,x}$	$p_f(2k)$
8	1.062	$3.93\ 10^{-1}$	0.591	$1.50\ 10^{-1}$	0.206	0
16	0.879	$2.98\ 10^{-1}$	0.131	$5.36\ 10^{-4}$	0.119	0
32	0.712	$1.49\ 10^{-1}$	0.071	$< 10^{-9}$	0.065	0
48	0.621	$7.83\ 10^{-2}$	0.049	$< 10^{-9}$	0.045	0

Since a storage system cannot afford such a probabilistic decoding failure, since data availability is crucial, we also modify the LT encoder so as to guarantee that $p_f(n) = 0$, i.e. that any sector can be recovered if all its fragments can be retrieved from the storage nodes. In other words, we want to exclude a read failure when all the storage nodes are assumed to be reliable. Such goal has been achieved by exploiting the incremental decoding property of the adopted OFG decoder. In fact, OFG has the advantage of being a sequential decoder and it is able to incrementally consume and decode the received coded fragments. This is achieved by maintaining a $k \times k$ decoding matrix that represents the set of collected linearly independent equations.

OFG can be used to incrementally filter the coded fragments to guarantee that $p_f(n) = 0$. This is achieved by imposing an additional constraint on the decodability of the fragments: ideally we want every coded fragment to be innovative, i.e., the corresponding equation shall not be linearly dependent on the previously generated equations. To this end, we use a standard LT encoder cascaded with the OFG incremental decoder; each new coded fragment is fed to the OFG stage that checks whether it is innovative or not. This goal is achieved exploiting the mechanism used by OFG to build the decoding matrix without any additional modification; in particular, the j-th equation is marked as innovative if it can be used to fill a new row of the OFG decoding matrix [20]. Clearly, a set of k innovative fragments (that we term as *decoding set*) is progressively obtained by this process. Then, to create redundant coded fragments, the OFG is reset and the process is re-initiated up to the generation of the desired number of coded fragments. If n is not multiple of k this amounts to interrupt the generation of the last decoding set.

It is worth pointing out that, since the coded fragments will be stored and retrieved from storage nodes randomly (in group of x), i.e. picking up coded fragments from different decoding sets, there is no guarantee to successfully decode from every subset of k coded fragments. The corresponding decoding distribution $\sigma_{n,k,x}$ can be used to analyze the OFG decoding overhead of the modified LT encoding; the modified encoding process guarantees by construction that $\sigma_{n,k,x}(n) = 1$, i.e. $p_f(n) = 0$. To illustrate our results, in Tab. 1 the modified ENIGMA LT codes are compared in terms failures and overhead versus standard LT codes. The results are obtained by performing 1,000 encoding trials with $x = 1$. In each trial, 100,000 decoding attempts are performed using different random orderings of the n fragments to emulate their random arrival from the storage nodes. It can be noted that the proposed process eliminates failure while slightly improving also in terms of average decoding overhead.

Finally, we must note that the devised solution requires running a standard LT encoder and the OFG algorithm in parallel adding some computational cost. For instance, when $k = 32$ standard and modified LT codes with OFG decoder take on average 0.054 and 0.248 ms per sector (measured on an Intel i7 processor), respectively. Moreover, as will be discussed in Sec. 4 ENIGMA employs caching to tolerate network latency and therefore the encoding of a sector is required only when it must be expunged from the cache.

4 System architecture and operations

In this section, we describe the architecture of ENIGMA (Sec. 4.1) and the sector read and write protocols (Sec. 4.2).

4.1 Architecture

ENIGMA provides access to an arbitrary set of VDs, each one consisting into a set of consecutively-numerated sectors. Its architecture, schematically shown in Fig. 1, consists of a set of N_S *storage nodes* and a set of N_P *proxies*.

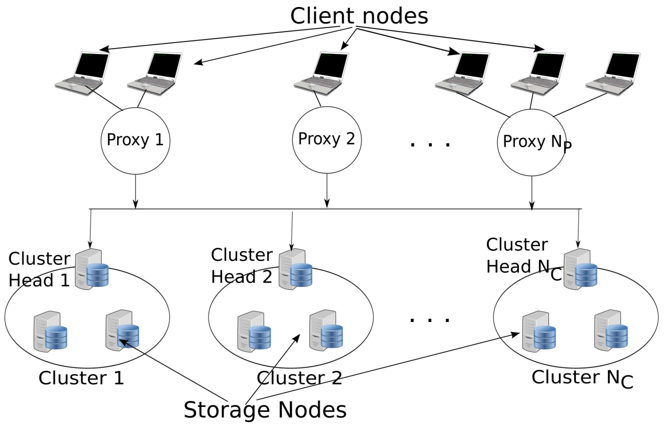


Fig. 1. System architecture

Each proxy provides a set of clients with the access to a set of VDs, and uses the storage nodes to store the corresponding sectors after having encoded them as discussed in Sec. 3.2. Each VD is exclusively managed by a single proxy, that maintains all the meta-data information needed to properly operate it. The number of VDs (and, consequently, of clients) that can be supported by a single proxy is not limited by architectural features, but instead depends only on the amount of physical resources of the machine where it runs. To enhance read and write performance, each proxy exploits caching (a vanilla LRU cache is used), operation overlapping, and suitable sector read/write protocols. In this paper, we assume that proxies are perfectly reliable, trusted, and secure.

Storage nodes are a set of hosts that use their local storage to store fragments of virtual sectors. We assume that storage nodes are characterized by their *machine availability* p ($0 \leq p \leq 1$), a measure of the probability that the fragments sent to the requesting proxy are actually received (for instance, these fragments might not be delivered because of a temporary network problem or machine unavailability). We assume that fragment unavailability is only transient (i.e., a node that leaves the system eventually rejoins it), so it is not necessary to perform any reconstruction of missing fragments [2].

For scalability purposes, and in order to achieve a good load balancing, storage nodes are arranged into a two-level hierarchical structure consisting of N_C equally-sized *clusters*, each one includes S_C storage nodes (each storage node belongs to a single cluster). Each cluster is managed by a *cluster head*, that is responsible for orchestrating, across the nodes of the corresponding cluster, the execution of the operations issued by proxies. More specifically, it receives the read/write requests for the sectors stored in the cluster, and dispatches them to the storage nodes belonging to its cluster.

We assume that nodes are assigned to clusters at random, without following any specific criterion related to their identity or connectivity, and that cluster heads are perfectly reliable, trusted, and secure.

All the fragments of a given sector s_i (whose address is i) are stored on storage nodes belonging to the same cluster, that are chosen as follows:

1) *cluster selection*: s_i is assigned (by the corresponding proxy) to cluster $C(s_i) = i \mod N_C$; this corresponds to a round-robin assignment that balances the load among cluster heads.
2) *storage nodes selection*: $C(s_i)$ head chooses at random, a set of m storage nodes of its cluster, and spreads the fragments of s_i over them. Note that different sectors may correspond to different subsets of storage nodes of the same cluster.
   The mapping between s_i and the corresponding set of storage nodes is stored locally on the cluster head. To reduce space requirements, mappings are not explicitly stored into a table, but are instead recalculated each time a sector is referenced by an operation. More precisely, the identifiers of the m storage nodes corresponding to fragments of s_i are computed as a sequence of m pseudo-random numbers whose generator uses i as seed. We stress that this operation takes very little time on a modern processor (for instance, just about $20 \mu sec.$ on an Intel i7 processor).

2. In case of permanent failure of a storage node, one of the various proposals for efficiently repairing lost fragments, e.g., [19], [21], could be used to devise a customized version of a regenerating code to support replacement of missing fragments.

As anticipated in Sec. 3, the fragments of each sector are stored and retrieved in groups of x, whereby each group is stored on a different storage node. Thus, we have that $m = \lceil n/x \rceil$. The value of x is set when the virtual disk is created, and is kept fixed during the lifetime of that disk; different virtual disks may choose a different value of x. To simplify notation, and without loss of generality, in the following we assume that n is an integer multiple of x (i.e., $n \mod x = 0$), so that we can drop the ceiling operator.

The use of the grouping factor allows us to control the number and the size of the messages exchanged between the proxy and the storage nodes for each operation (see Sec. 4.2). In this way it is possible to avoid using many small messages that, as shown in Sec. 6, adversely affect performance.

In particular, denoting as D_V the sector size of a VD, we have that the size of each fragment is $f_s = D_V/k$, so each message sent by a storage node to the proxy carries $p_{size} = x \cdot D_V/k$ bytes as payload. Thus, for given values of D_V and k, a proper choice of x allows one to reduce the number of messages and to increase their payload to a size resulting in efficient usage of network resources.

As will be shown later, the value of x affects the performance, availability and integrity of data that can be attained by ENIGMA. We will explore the relationships among these quantities in Sect.6 and in Appendix A of the supplement.

4.2 Read and Write Protocols

The read and write protocols specify how the read and write operations issued by the proxy are carried out by ENIGMA.

The read and write operations are implemented as follows, where we denote as s_i the sector referenced by the operation:

- *write*: the proxy encodes s_i using the LT coding procedure described in Sect. 3.2, and sends the resulting n coded fragments $f_{i,j}$, $j = 1, \ldots, n$ to the cluster head of the corresponding cluster $C(s_i)$, that in turn forwards them to a suitable set of storage nodes (chosen as discussed in Sec. 4.1). Each storage node directly notifies the requesting proxy, that considers as completed the operation as soon as the first decoding set has been successfully stored (recall that the ENIGMA LT encoder organizes the coded fragments into independent decoding sets), that in turn notifies the requesting client.
  Each coded fragment $f_{i,j}$is sent along with a header, allowing the proxy to recover the sector index i and the coding index j, respectively. Given the index j, the proxy is the only entity that knows the value of the seed used to run the pseudo-random generation process, whose knowledge allows to repeat the selection of the degree d and of set of the combined original fragments. In the following we refer to the

seed value as *coding key*. Only the proxy knows the coding key and is able to to decode a sector.

- *read*: the proxy sends a read request to the cluster head of the corresponding cluster $C(s_i)$, that in turn forwards the request to each of the n/x storage nodes that store the coded fragments. Each one of the involved storage nodes reacts to that message by directly sending to the proxy the x fragments of s_i that it holds. The proxy processes the received fragments and progressively decodes s_i using the OFG algorithm. When the proxy is able to decode s_i, it notifies to the requesting client the completion of the read operation, and discards all the fragments received after the sector has been fully decoded.

5 System Properties

In this section we characterize the levels of availability, confidentiality, and integrity attained by ENIGMA virtual disks, and we devise suitable analytical models that express these properties in terms of the system parameters. More specifically, we devise models for the properties of a single cluster, given that all clusters in an ENIGMA instantiation are identical to each other. Appendix A in the supplement complements this section by discussing numerical results based on these models. Furthermore models of ENIGMA storage and communication overheads are presented.

5.1 Sector Availability

In ENIGMA, the unavailability of sector fragments caused by the unavailability of the storage nodes holding them is dealt with by choosing suitable values for k and n, so that it is possible to reconstruct each original sector (i.e., its original k fragments) even in face of these events.

In order to choose these values, we develop an analytical model of the *availability* α of each sector, expressed as the probability of retrieving enough fragments to allow correct decoding of a sector stored in a cluster, that relates this quantity to the various design parameters of ENIGMA (namely k, n, and x) and the availability p of individual storage nodes. Our model can be used to determine the values of k, n, and x that allow ENIGMA to obtain a desired α value.

Recalling that all the fragments of a given sector are stored on the storage nodes belonging to a single cluster, the availability of that sector can be expressed as function of the properties of the corresponding cluster.

We denote as $g_{n,x,p}(r)$ the probability that r out of n fragments are gathered by the proxy when all n fragments are requested to decode a sector. As discussed in Sec. 4, these fragments are placed on n/x storage nodes (all belonging to the same cluster). Without loss of generality and for the sake of simplicity in the following we assume that n is an integer multiple of x, i.e., $n \bmod x = 0$. Recalling that the x fragments requested to each storage node are obtained by the proxy with probability p (and none of them with probability $1-p$), and that $n \bmod x = 0$, we have that $g_{n,x,p}(r)$ follows a binomial probability distribution when r is an integer multiple of x, i.e.,

$$g_{n,x,p}(r) = \begin{cases} \binom{n/x}{r/x} p^{r/x} (1-p)^{(n-r)/x} & \text{if } r \bmod x = 0 \\ 0 & \text{if } r \bmod x \neq 0. \end{cases}$$

The probability of decoding a sector by using exactly r fragments is given by the product $g_{n,x,p}(r)\sigma_{n,k,x}(r)$. We can then define the availability $\alpha_{n,x,p}(k)$ of a sector as the probability of retrieving enough coded blocks to allow decoding for any possible value of r, i.e.:

$$\alpha_{n,x,p}(k) = \sum_{r=k}^{n} g_{n,x,p}(r)\sigma_{n,k,x}(r). \tag{1}$$

By using Eq. 1, it is possible to determine suitable values for k, n, and x, as well as to evaluate data availability for specific values of the above design parameters. For the complete evaluation please refer to the supplement. Here we limit ourselves to remark that our results indicate that, to ensure high availability, x should be chosen as small as possible. Furthermore, the proposed encoding strategy improves availability for large values of x.

It is worth noting that our fragment placement policy corresponds to a simple and intuitive symmetric allocation with maximal spreading using the terminology of [22]. Results in [22] are obtained under the hypothesis of optimal coding schemes, i.e., coding schemes where decoding is guaranteed whenever the ratio between the amount of retrieved coded fragments and k is ≥ 1. This assumption is equivalent to assume that the code overhead is identically equal to 0. Unfortunately, this assumption is not satisfied in ENIGMA where rateless codes only allow for probabilistic decoding according to the code overhead distribution described by $\varepsilon_{n,k,x}(j)$. Nevertheless, the symmetric allocation with maximal spreading is proved to be asymptotically optimal, i.e., to maximize the sector availability when the number of storage nodes increases, in the case of independent probabilistic access to each storage node with probability p if $p > \frac{k}{n}$. Furthermore, [22] shows that for $p \geq \frac{4}{3\lfloor n/k \rfloor}$ (the values of p we consider for ENIGMA in Appendix A satisfy this constraint) symmetric allocation with maximal spreading is indeed one of the optimal solutions. It follows that the fragment allocation policy we used in ENIGMA yields close to optimal solution to the fragment allocation problem, at least for the values of p we consider in our paper.

5.2 Data Confidentiality

Storing data on third-party resources potentially exposes them to unauthorized accesses, while preventing the owner of those data to enforce specific access policies. Thus, it is necessary to ensure the *confidentiality* of data, that entails to make sure that they are not disclosed to unauthorized users.

In the following, we discuss how we obtain confidentiality in face of a threat model in which an attacker is able to break only storage nodes while we assume that proxies are trusted entities that cannot be compromised.

In ENIGMA confidentiality is achieved by disclosing only coded information, whereas the coding key (i.e. the seed of the random generator used for random LT encoding) is safely stored by the proxy only. This implies that the coding vector for each sector s_i (see Sec. 4.2) is private information of the owner of the data. Therefore, by keeping the coding key secret, and by limiting the number of fragments stored on each node, ENIGMA is able to resist to different types of attacks to confidentiality, that we discuss in detail in Appendix A of the supplement.

Our scheme is inspired by [23] where dispersal of information is proposed as an efficient mean for achieving suitable level of data security. More precisely, in [23] it is shown that linear coding can be exploited to achieve security as long as an attacker has access to less than k fragments. On the other hand, if an attacker collects enough fragments to decode the sector, security is obtained by making information reconstruction computationally hard, since an attacker cannot access the coding key as long as it is kept secret.

It follows that our scheme provides the so called *computational security* relying on the external and safe storage of the coding key. Higher security levels could be achieved by encrypting the coding key. Since in the long term keys can be compromised [24] a further improvement could be to adopt schemes where no external key is required as in [25], [26]. Of course, the entire disk could be encrypted before storing it on ENIGMA to achieve further security levels at the expense of a higher computational cost.

It must be noted that an attacker that succeeds in reconstructing a single disk sector still has a complex work ahead in order to obtain the entire logical file the stolen sector belongs to. Indeed, a mechanism should be set up to infer which other disk sectors compose the logical file and what is their offset in the file. Finally, the cluster storing each of them must be known to the attacker. This last problem can be made harder for an attacker by replacing the round-robin based cluster selection policy described in Sec. 4.1 by a random based one. Please refer to Appendix A of the supplement for the quantitative evaluation of the confidentiality achieved by ENIGMA. The results discussed there show that fragment dispersal and secrecy of the coding key provide computational security. Moreover, it is shown that dispersal is more effective for low values of x.

5.3 Data Integrity

In addition to the risks for confidentiality, data stored on virtual disks are exposed to integrity attacks, whereby a set of malicious nodes purposely corrupt some of the fragments they store. In particular, we assume that N_M out of S_C storage nodes of a cluster are malicious and deliberately modify the content of a fragment. In the attempt of eluding or delaying their identification, malicious storage nodes alter the content of a fragment on a probabilistic basis: when a fragment has to be transmitted to a requesting proxy its content is modified by flipping a coin whose weight is denoted as p_{poll}. We name this attacks as *pollution attack*, as data are corrupted by “polluting” them.

In this section we derive an analytical model that allows us to compute the ability of ENIGMA to resist to pollution attacks, quantified as the availability of sectors (i.e., the probability of correctly reconstruct them) in presence of purposely corrupted fragments.

We denote as $s_{S_C,N_M,n,x}(r)$ the probability that r fragments are placed on malicious storage nodes ($0 \leq r/x \leq N_M$). We observe that if r is an integer multiple of x then $s_{S_C,N_M,n,x}(r)$ follows a hyper-geometric distribution:

$$s_{S_C,N_M,n,x}(r) = \begin{cases} \frac{\binom{N_M}{r/x}\binom{S_C-N_M}{(n-r)/x}}{\binom{S_C}{n/x}} & \text{if } r \bmod x = 0 \\ 0 & \text{if } r \bmod x \neq 0. \end{cases}$$

We can still define the availability of a sector as the probability of retrieving enough fragments to allow decoding when malicious storage nodes play into action. This generalized quantity is computed as in Eq. 2 whose meaning is the following: assume that

- n_m fragments are placed on malicious storage nodes and that $n - n_m$ fragments are placed on honest storage nodes (this happens with probability $s_{S_C,N_m,n,x}(n_m)$);
- $0 \leq r_m \leq n_m$ fragments are retrieved from malicious storage nodes whose availability is $p\ (g_{n_m,x,p}(r_m))$;
- $0 \leq r_h \leq n - n_m$ fragments are retrieved from honest storage nodes whose availability is $p\ (g_{n-n_m,x,p}(r_h))$.

A clean sector is decoded by using d fragments if:

- the overhead is equal to $d - k$ when placing x fragments on each storage node: this happens with probability $\varepsilon_{n,k,x}(d)$, and
- if m out of d fragments that are hosted by malicious storage nodes are left unmodified: this happens with probability $\mathcal{H}(r_m, r_h, d, m)(1 - p_{poll})^m$ where $\mathcal{H}(r_m, r_h, d, m) = \frac{\binom{r_m}{m}\binom{r_h}{d-m}}{\binom{r_m+r_h}{d}}$ is the hyper-geometric probability distribution.

Clearly, all values for m and d must be considered and all the above reasoning must be iterated over all possible assignments of n_m fragments to N_M malicious storage nodes and over all possible values for r_m and r_h.

Accurate and efficient algorithms to spot and identify malicious storage nodes can be considered [27]. If we assume that it is possible to identify malicious storage nodes, we can envisage a recover mechanism for inconsistent sectors. To this end, we consider only the r_h fragments retrieved from honest storage nodes and try decoding from those. In this case we compute a recovery probability for inconsistent sectors as in Eq. 3.

$$c_{S_C,N_M,n,x,p}(k)=\sum_{n_m=0}^{n}\sum_{r_m=0}^{n_m}\sum_{r_h=0}^{n-n_m} s_{S_C,N_m,n,x}(n_m)g_{n_m,x,p}(r_m)g_{n-n_m,x,p}(r_h)\sum_{d=k}^{r_m+r_h}\varepsilon_{n,k,x}(d)\sum_{m=0}^{r_m}\frac{\binom{r_m}{m}\binom{r_h}{d-m}}{\binom{r_m+r_h}{d}}(1-p_{poll})^m \tag{2}$$

$$r_{S_C,N_M,n,x,p}(k)=\sum_{n_m=1}^{n}\frac{s_{S_C,N_m,n,x}(n_m)}{1-s_{S_C,N_m,n,x}(0)}\sum_{r_h=k}^{n-n_m} g_{n-n_m,x,p}(r_h)\sigma_{n,k,x}(r_h). \tag{3}$$

Please refer to Appendix A of the supplement for the quantitative evaluation of the the resistance of ENIGMA to pollution attacks for specific values of its design parameters. The results discussed there can be summarized by saying that small values of the grouping factor x worsen the effect of a pollution attack but ensure high recovery probability.

6 Performance Evaluation

To assess the usability of ENIGMA in practical settings we compared the performance it attains against that achieved by a *baseline system* representing a typical back-end for a BLCS system that uses whole-sector replication to ensure availability. Furthermore, we studied the impact on ENIGMA performance of its system parameters. Because of space constraints in this section we only discuss the comparison against the baseline system and we briefly comment on the impact on performance of its system parameters. For this study, we use a discrete-event simulator, that we developed for this purpose, able to simulate both ENIGMA and the baseline systems. In the rest of this section, we describe the simulation scenarios used for our experiments, and then we report the corresponding results. Appendix B in the supplement complements this section by discussing the simulation models used in the experiments, as well as by providing the complete discussion of the impact on performance of ENIGMA system parameters.

6.1 Simulation scenarios

The simulation scenarios considered in our experiments are obtained by instantiating the system parameters as well as the parameters of the simulation models described in the supplement. To ease readability, we list these parameters, together with their explanations and corresponding values, in Tab. 2.

In the experiments we use as workload a set of real-world disk traces [28], that can be downloaded from the *SNIA IOTTA Repository*, and that have been collected on various storage production servers at Microsoft Corporation [3]. These traces represent disk access patterns of various types, featuring a mix of read and write operations that involve sequences of consecutive sectors. The actual proportions of read/write in a given trace, as well as the number of consecutive sectors they target, vary across these traces. Space constraints prevent us from fully discussing them here (the interested reader may refer to [29] for a thorough discussion).

TABLE 2
Simulation parameters and values

Parameter	Meaning	Values
N_P	number of proxies	16
N_S	number of storage nodes	8192
N_C	number of clusters	$64, 128, 256$
S_C	size of each cluster	$128, 64, 32$
p	availability of storage nodes i	1
D_V	sector size	8192 bytes
k	original sector fragments	32
n	encoded sector fragments	$64, 128$
x	grouping factor	$1, 2, 4, 8, 16, 32, 64, 128$

6.2 Results

In our experiments we use the following metrics to quantify performance:

- the *mean response time* of individual disk requests;
- the *total number of I/O operations per second (IOPS)* completed by the whole storage infrastructure.

Because of space limitations, here we discuss only the results corresponding to the configuration using $N_C = 64$ clusters, and to a single trace [4]. The results for the other traces and system configurations are, however, very similar to the ones we report here.

6.2.1 Comparison with the baseline system

To determine whether ENIGMA offers performance comparable to that of a traditional back-end for BLCS systems, we carry out a set of experiments in which we compare its performance against those attained by the baseline system (please refer to Appendix B for the description of the corresponding simulation model).

Given that a critical factor affecting performance of the baseline system is the processing speed of the servers it uses, we consider various scenarios in which we progressively increase the number C of requests that each one of these servers can concurrently process; in particular, we run experiments for $C = 1, 10, 100, 1000$. In contrast, the results for the ENIGMA scenarios have been obtained by setting $C = 1$ on each cluster heads.

Finally, we consider a scenario in which baseline clients do not use a local cache (as in the *iSCSI* protocol),

3. In particular, we use the *MSN-CFS, RAD-AS, RAD-BE, DAP-DS, and DAP-PS* traces

4. In particular, we consider the requests targeting disk 0 of the MSN-CFS trace.

and one in which such a cache is instead present (as in the *Fiber Channel* protocol) and, in this case, we set its size to 10GB (a value higher than that used by most real storage systems). Because of space constraints, we show only the results for the baseline system using the cache, as this configuration results in the best performance for this system.

In Fig. 2 we compare the mean response time and the IOPS values obtained by the the baseline system for the various values of C against those attained by ENIGMA for different values of k, n and x and with a cache of 1 GB for each proxy.

As can be seen from Fig. 2, ENIGMA performs much better than the baseline system for $C = 1, 10$, and slightly better for $C = 100, 1000$ – both in terms of the mean response time and of the IOPS value – for those values of n and x that do not congest the downstream channel. Under congestion, however, ENIGMA still performs better than the baseline system for $C = 1, 10$ (although response times are very high).

The poor performance of the baseline system are due to the congestion of the input queue of storage servers, where disk operations wait a long amount of time before being served. Congestion is due to the fact that all the requests for a given disk target the same storage server (in contrast, in ENIGMA they are distributed across all the cluster heads). The usage of a very large cache in the baseline system is of little help, as its efficacy depends – as already discussed – on the amount of locality exhibited by the traces, a feature not offered by the traces considered in this work. We remark, however, that we use real-world traces collected on enterprise-level storage systems and, as such, can be considered representative of a large family of disk workloads.

As a final consideration, we point out that using 64 cluster heads (each one with $C = 1$) roughly corresponds (in terms of computing power) to setting $C = 64$ in the baseline system. However, even in this case the performance of the baseline system is much worse than those attained by ENIGMA.

Remark: ENIGMA achieves much better performance than an equivalent-cost (in terms of computing power) baseline system. To achieve similar performance, a baseline system must use a much larger amount of resources (thus exhibiting a larger cost).

6.2.2 *Impact of system parameters*

To assess how the system parameters of ENIGMA affect its performance, we perform various simulation experiments in which in turn we vary the value of n, the size of the proxy cache, and the proxy placement in the network.

Our results, that are fully discussed in Appendix B, can be summarized as follows:

- to achieve good performance, larger values of x should be used, and the larger the value of n, the larger must be the value of x;
- larger caches result in performance improvements only for larger values of x, while they provide limited benefit for smaller ones;
- the position of proxies on the network has a limited effect on performance.

7 Conclusions

In this paper we devised a suitable architecture for the back-end of BLCS systems that achieves adequate levels of access and transfer performance, availability, integrity, and confidentiality, for the data it stores. We exploited LT rateless codes and showed how beneficial they are to all system properties we considered. In particular:

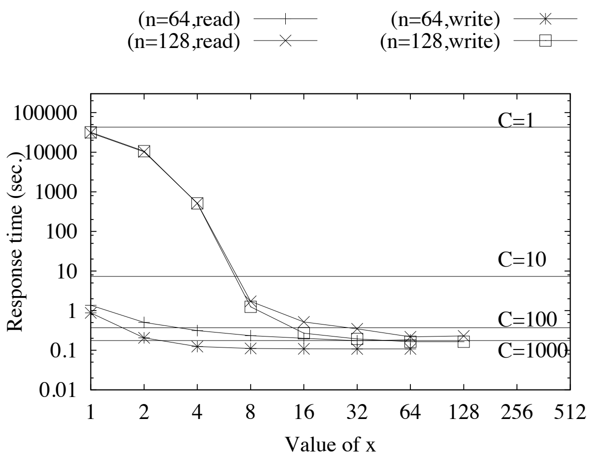
- the rateless property allows to blindly spread coded fragments to storage nodes in a cluster with the level of redundancy achieving the desired availability. Moreover, we devised a particular encoding strategy that for small block sizes ($k \leq 32$) guarantees zero decoding failure probability and improves availability for large values of the grouping factor x.
- confidentiality is obtained by keeping the coding key secret, i.e., we assume that proxies are trusted and cannot be compromised. Furthermore, the lower the values of x the higher the confidentiality.
- rateless codes allow for detection of polluted sectors and accurate and fast identification of malicious storage nodes [27]. Furthermore, we show that after identification high recovery probability can be achieved especially for low values of x.

As for the performance, we showed that much better performance than an equivalent-cost baseline system can be achieved even when caches are small and independently of the position of proxies in the network.

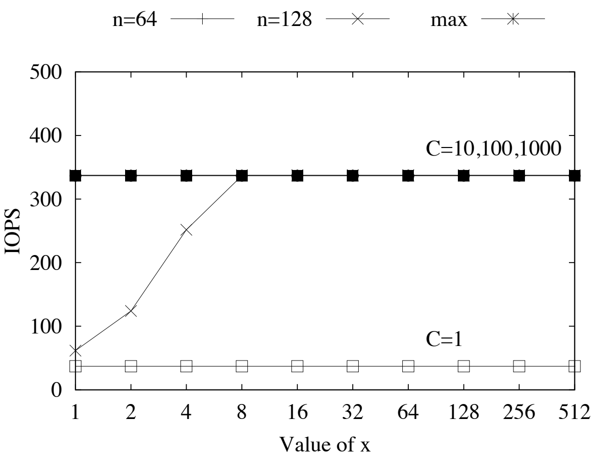
The future development foreseen for the current work are a thorough evaluation of the malicious storage nodes identification technique presented in [27] to assess accuracy, robustness, and reactivity. Furthermore, a prototype implementation on PlanetLAB is planned for the whole architecture.

References

[1] H. Dewan and R. Hansdah, “A survey of cloud storage facilities,” in *IEEE SERVICES*, jul 2011, pp. 224 –231.
[2] M. Bishop, *Introduction to Computer Security*. Addison-Wesley, 2005.
[3] M. Luby, “LT codes,” in *IEEE FOCS*, Nov. 2002, pp. 271–280.
[4] I. S. Reed and S. Gustave, “Polynomial codes over certain finite fields,” *Journal of the Society for Industrial & Applied Mathematics*, vol. 8, no. 2, pp. 300–304, 1960.
[5] A.G. Dimakis, K. Ramchandran, Y. Wu, and C. Suh, “A Survey on Network Codes for Distributed Storage,” *Proc. of IEEE*, vol. 99, no. 3, Mar. 2011.
[6] M. Zola, V. Bioglio, C. Anglano, R. Gaeta, M. Grangetto, and M. Sereno, “Enigma: Distributed virtual disks for cloud computing,” in *IEEE IPDPSW*, May 2011.
[7] “Amazon Elastic Compute Cloud (Amazon EC2).” [Online]. Available: http://aws.amazon.com/ec2
[8] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, “The Eucalyptus Open-source Cloud-computing System,” in *IEEE CCGRID*, Shangai, China, May 2009.



(a)



(b)

Fig. 2. Comparison with the baseline system: (a) mean response time, (b) IOPS.

[9] "OpenNebula: The Open Source Toolkit for Cloud Computing." [Online]. Available: http://www.opennebula.org

[10] X. Gao, M. Lowe, and M. Pierce, "Supporting Cloud Computing with the Virtual Block Store System," in *IEEE e-Science*, December 2009.

[11] X. Gao, Y. Ma, M. Pierce, M. Lowe, and G. Fox, "Building a Distributed Block Storage System for Cloud Infrastructure," in *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, Dec. 2010, pp. 312 –318.

[12] L. Zhou, Y.-C. Wang, J.-L. Zhang, J. Wan, and Y.-J. Ren, "Optimize Block-Level Cloud Storage System with Load-Balance Strategy," in *IEEE IPDPSW*, May 2012.

[13] "Encryption as a Service from Certes Networks." [Online]. Available: http://www.certesnetworks.com/products/encryption-as-a-service.html

[14] S. Kamara and K. Lauter, "Cryptographic cloud storage," in *FC*. Springer-Verlag, 2010.

[15] S. Zarandioon, D. Yao, and V. Ganapathy, "K2c: Cryptographic cloud storage with lazy revocation and anonymous access," in *Security and Privacy in Communication Networks*, ser. Lecture Notes of the Inst. for Computer Sciences, Social Informatics and Telecommunications Engineering. Springer Berlin Heidelberg, 2012, vol. 96, pp. 59–76.

[16] K. Bailey, "Addressing the Growth and Complexity of Information Security Concerns," International Data Corporation (IDC), Tech. Rep., Feb. 2013.

[17] N. Cao, S. Yu, Z. Yang, W. Lou, and Y. Hou, "LT codes-based secure and reliable cloud storage service," in *IEEE INFOCOM*, 2012, pp. 693–701.

[18] A. Shokrollahi, "Raptor codes," *IEEE Trans. on Information Theory*, vol. 52, no. 6, pp. 2551–2567, 2006.

[19] A. Dimakis, P. Godfrey, Y. Wu, M. Wainwright, and K. Ramchandran, "Network Coding for Distributed Storage Systems," *IEEE Trans. Inf. Theory*, vol. 56, no. 9, Sept. 2010.

[20] V. Bioglio, R. Gaeta, M. Grangetto, and M. Sereno, "On the fly gaussian elimination for lt codes," *IEEE Communication Letters*, vol. 13, pp. 953–955, 2009.

[21] A. Megasthenis and A. Dimakis, "Repairable Fountain Codes," in *IEEE ISIT*. IEEE, 2012.

[22] D. Leong, A. Dimakis, and T. Ho, "Distributed storage allocations," *IEEE Transactions on Information Theory*, vol. 58, no. 7, pp. 4733–4752, 2012.

[23] M. O. Rabin, "Efficient dispersal of information for security, load balancing, and fault tolerance," *J. ACM*, vol. 36, no. 2, pp. 335–348, 1989.

[24] M. W. Storer, K. M. Greenan, E. L. Miller, and K. Voruganti, "Potshards a secure, recoverable, long-term archival storage system," *Trans. Storage*, vol. 5, no. 2, pp. 1–35, 2009.

[25] H. Krawczyk, "Secret sharing made short," in *Proceedings of the 13th Annual International Cryptology Conference on Advances in Cryptology*, ser. CRYPTO 1993, 1994, pp. 136–146.

[26] J. Resch and J. S. Plank, "AONT-RS: Blending security and performance in dispersed storage systems," in *Proc. of USENIX FAST 11*, San Jose, USA, 2011.

[27] R. Gaeta and M. Grangetto, "Identification of malicious nodes in peer-to-peer streaming: A belief propagation based technique," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 10, pp. 1994–2003, 2013.

[28] "SNIA - Storage Networking Industry Association: IOTTA Repository Homes." [Online]. Available: http://iotta.snia.org/

[29] S. Kavalanekar, B. Worthington, Q. Zhang, and V. Sharda, "Characterization of storage workload traces from production windows servers," in *IEEE IISWC*, sept 2008, pp. 119 –128.

[30] L. N. Bairavasundaram, G. R. Goodson, S. Pasupathy, and J. Schindler, "An analysis of latent sector errors in disk drives," in *ACM SIGMETRICS*, 2007, pp. 289–300.

[31] V. Bioglio, M. Grangetto, R. Gaeta, and M. Sereno, "An optimal partial decoding algorithm for rateless codes," in *IEEE ISIT*, 2011, pp. 2731–2735.

[32] "The DiskSim Simulation Environmet (Version 4.0)." [Online]. Available: http://www.pdl.cmu.edu/DiskSim

[33] "Internet Delay Space Synthesizer." [Online]. Available: http://www.cs.rice.edu/~bozhang/ds2/

[34] B. Zhang, T. S. E. Ng, A. Nandi, R. H. Riedi, P. Druschel, and G. Wang, "Measurement-based analysis, modeling, and synthesis of the Internet delay space," *IEEE/ACM Trans. on Networking*, vol. 18, no. 1, pp. 229–242, Feb. 2010.

[35] S. Kavalanekar, B. Worthington, Q. Zhang, and V. Sharda, "Characterization of storage workload traces from production Windows Servers," in *IISWC*, October 2008, pp. 119–128.

Cosimo Anglano Cosimo Anglano received his Laurea Ph.D. degrees in Computer Science from the University of Torino, Italy, in 1990 and 1994, respectively. He is currently an associate professor of Computer Science at the University of Piemonte Orientale "A. Avogadro", Alessandria, Italy. His current research interests include resource management algorithms for cloud computing, distributed storage systems, and digital forensics.

Rossano Gaeta Rossano Gaeta received his Laurea and Ph.D. degrees in Computer Science from the University of Torino, Italy, in 1992 and 1997, respectively. He is currently Associate Professor at the Computer Science Department, University of Torino. He has been recipient of the Best Paper award at the 14-th IEEE/ACM International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2006) and at the 26th International Symposium on Computer Performance, Modeling, Measurements, and Evaluation (PERFORMANCE 2007). His current research interests include the design and evaluation of peer-to- peer computing systems and the analysis of compressive sensing and coding techniques in distributed applications.

Marco Grangetto M. Grangetto (S99-M03-SM09) received his Electrical Engineering degree and Ph.D. degree from the Politecnico di Torino, Italy, in 1999 and 2003, respectively. He is currently Associate Professor at the Computer Science Department, University of Torino. His research interests are in the fields of multimedia signal processing and networking. In particular, his expertise includes wavelets, image and video coding, data compression, video error concealment, error resilient video coding unequal error protection, and joint source channel coding. Prof. Grangetto was awarded the Premio Optime by Unione Industriale di Torino in September 2000, and a Fulbright grant in 2001 for a research period with the Department of Electrical and Computer Engineering, University of California at San Diego. He has participated in the ISO standardization activities on Part 11 of the JPEG 2000 standard. He has been a member of the Technical Program Committee for several international conferences, including the IEEE ICME, ICIP, ICASSP, and ISCAS.