Securing Dynamic Distributed Storage Systems against Eavesdropping and Adversarial Attacks

Sameer Pawar, Salim El Rouayheb, *Member, IEEE* and Kannan Ramchandran, *Fellow, IEEE*

***Abstract*—We address the problem of securing distributed storage systems against eavesdropping and adversarial attacks. An important aspect of these systems is node failures over time, necessitating, thus, a repair mechanism in order to maintain a desired high system reliability. In such dynamic settings, an important security problem is to safeguard the system from an intruder who may come at different time instances during the lifetime of the storage system to observe and possibly alter the data stored on some nodes. In this scenario, we give upper bounds on the maximum amount of information that can be stored safely on the system. For an important operating regime of the distributed storage system, which we call the *bandwidth-limited regime*, we show that our upper bounds are tight and provide explicit code constructions. Moreover, we provide a way to short list the malicious nodes and expurgate the system.**

***Index Terms*—Byzantine adversary, Distributed Storage, Network Codes, Secrecy.**

I. Introduction

Distributed storage systems (DSS) consist of a collection of n data storage nodes, typically individually unreliable, that are collectively used to reliably store data files over long periods of time. Applications of such systems are innumerable and include large data centers and peer-to-peer file storage systems such as OceanStore [1], Total Recall [2] and DHash++ [3] that use a large number of nodes spread widely across the Internet. To satisfy important requirements such as data reliability and load balancing, it is desirable for the system to be designed to enable a user, also referred to as a data collector, to download a file stored on the DSS by connecting to a smaller number k, $k < n$, nodes. An important design problem for such systems arises from the individual unreliability of the system nodes due to many reasons, such as disk failures (often due to the use of inexpensive "commodity" hardware) or peer "churning" in peer-to-peer storage systems. In order to maintain a high system reliability, the data is stored redundantly across the storage nodes. Moreover, the system is repaired every time a node fails by replacing it with a new node that connects to d other nodes and download data to replace the lost one.

Codes for protecting data from erasures have been well studied in classical channel coding theory, and can be used here to increase the reliability of distributed storage systems. Fig. 1 illustrates an example where a $(4, 2)$ *maximal distance separable* (MDS) code is used to store a file $\mathcal{F}$ of 4 symbols $(a_1, a_2, b_1, b_2) \in \mathbb{F}_5^4$ distributively on $n = 4$ different nodes, $v_1, \ldots, v_4$, each having a storage capacity of two symbols. The $(4, 2)$ MDS code ensures that a data collector connecting to any $k = 2$ storage nodes, out of $n = 4$, can reconstruct the whole file $\mathcal{F}$. However, what distinguishes the scenario here from the erasure channel counterpart is that, in the event of a node failure, the system needs to be repaired by replacing the failed node with a new one. A straightforward repair mechanism would be to add a replacement node that connects to $k = 2$ other nodes, downloads the whole file, reconstructs the lost part of the data and stores it. One drawback of this solution is the relatively high repair bandwidth, *i.e.*, the total amount of data downloaded by the new replacement node. For this straightforward repair scheme, the repair bandwidth is equal to the size of the file $\mathcal{F}$ which can be large in general. A more efficient repair scheme that requires less repair bandwidth is depicted in Fig. 1 where node v_1 fails and is replaced by node v_5. By making node v_5 connect to $d = 3$ nodes instead of $k = 2$, it is possible to decrease the total repair bandwidth from 4 to 3 symbols. Note that, in the proposed repair solution, v_5 does not store the exact data that was on v_1; the only required property is that the

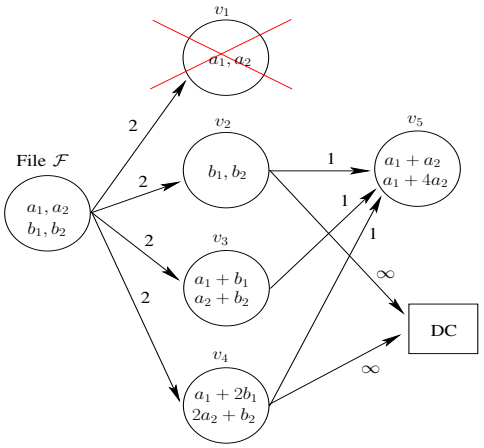

Fig. 1. An example of a distributed data storage system under repair. A file $\mathcal{F}$ of 4 symbols $(a_1, a_2, b_1, b_2) \in \mathbb{F}_5^4$ is stored on four nodes using a $(4, 2)$ MDS code. Node v_1 fails and is replaced by a new node v_5 that downloads $(b_1 + b_2), (a_1 + a_2 + b_1 + b_2)$ and $(a_1 + 4a_2 + 2b_1 + 2b_2)$ from nodes v_2, v_3 and v_4 respectively to compute and store $(a_1 + a_2, a_1 + 4a_2)$. Nodes $v_2, \ldots, v_5$ form a new $(4, 2)$ MDS code. The edges in the graph are labeled by their capacities. The figure also depicts a data collector connecting to nodes v_2 and v_4 to recover the stored file.

This research was funded by an NSF grant (CCF-0964018), a DTRA grant (HDTRA1-09-1-0032), and in part by an AFOSR grant (FA9550-09-1-0120).

Sameer Pawar is with the Wireless Foundation, Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA 94704 USA (e-mail: spawar@eecs.berkeley.edu).

Salim El Rouayheb is with the Wireless Foundation, Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA 94704 USA (e-mail: salim@eecs.berkeley.edu).

K. Ramchandran is with the Wireless Foundation, Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA 94704 USA (e-mail: kannanr@eecs.berkeley.edu).

data stored on all the surviving nodes v_2, v_3, v_4 and v_5 form a $(4, 2)$ MDS code. The above important observations were the basis of the original work of [4] where the authors showed that there exists a fundamental tradeoff between the storage capacity at each node and the repair bandwidth. They also introduced and constructed *regenerating codes* as a new class of codes that generalizes classical erasure codes and permits the operation of a DSS at any operational point on the optimal tradeoff curve.

When a distributed data storage system is formed using nodes widely spread across the Internet, e.g., peer-to-peer systems, individual nodes may not be secure and may be thus susceptible to an intruder that can eavesdrop on the nodes and possibly modify their data, e.g., viruses, botnet, etc. In this work, we address the issue of securing dynamic distributed storage systems, with nodes continually leaving and joining the system, against such intruders. The dynamic behavior of the system can jeopardize the data by making the intruder more powerful. For instance, while eavesdropping on a new node during the repair process, the intruder can observe not only its stored content but also all its downloaded data. Moreover, it allows an adversary to introduce errors on nodes beyond his/her control by sending erroneous messages when contacted for repair.

In our analysis, we focus on three different types of intruders: (i) a *passive eavesdropper* who can eavesdrop on ℓ nodes in the system, (ii) an *active omniscient adversary* who has complete knowledge of the data stored in the system and can maliciously modify the data on any b nodes in the system, and (iii) an *active limited-knowledge adversary* who can eavesdrop on any ℓ nodes and can maliciously corrupt the data on any b nodes among the ℓ observed ones. In the last case, the intruder's knowledge about the stored data in the system is limited to what can be inferred from the nodes he/she is observing.

We define the *secrecy* and *resiliency capacities* of a distributed storage system as the maximum amount of information that it can store safely, respectively, in the presence of an eavesdropper or a malicious adversary. For these intruder scenarios, we derive general upper bounds on the secrecy and resiliency capacity of the system. Motivated by system considerations, we define an important operation regime that we call the *bandwidth-limited* regime where there is a fixed allowed budget for the repair bandwidth with no constraints on the node storage capacity. This regime is of increasing importance due to the asymmetry in the cost of bandwidth vs. storage. For the bandwidth-limited regime, we show that our upper bounds are tight and provide explicit constructions of capacity-achieving codes.

The work in this paper is related to the recent work in the literature on secure network coding for networks with restricted wiretapping sets [5] and networks comprising traitor nodes [6]. The problem of studying such networks is known to be much harder in general than models considering (unrestricted) compromised edges instead of nodes. For instance, the work of [5] implies that finding the secrecy capacity of networks with wiretapped nodes is an NP-hard problem. Moreover, non-linear coding at intermediate network nodes may be necessary for securing networks against malicious nodes as shown in [6]. The contribution of this paper resides, at a high level, in showing that the networks representing distributed storage systems have structural symmetry that makes the security problem more tractable than in general networks. We leverage this fact to derive the exact expressions of the secrecy and resiliency capacities of these systems in the important bandwidth-limited regime. Moreover, we present capacity-achieving codes that are linear. These codes are characterized by a separation property: the file to be stored is first encoded for security then stored in the system without any modification to the internal operation of the system nodes. An additional interesting property of our proposed codes is that, in the active adversary case, they permit the identification of a small list of suspected nodes guaranteed to contain the malicious ones, permitting thus the expurgation of the system.

The rest of this paper is organized as follows. In Section II, we discuss related work on distributed storage systems and secure network coding. In Section III, we describe the flow graph model for distributed storage systems and elaborate on the intruder model. We provide a brief summary of our main results in Section IV. In Section V, we derive an upper bound on the secrecy capacity of the system and provide an achievable scheme for the bandwidth-limited regime. We provide a similar analysis for the omniscient and limited-knowledge adversary cases respectively in Section VI and Section VII, where we find upper bounds on the resiliency capacity and construct capacity achieving codes for the bandwidth-limited regime. We conclude the paper in Section VIII and discuss some related open problems.

II. Related Work

The pioneering work of Dimakis et al. in [4], [7], [8], demonstrated the fundamental trade-off between repair bandwidth and storage cost in a distributed storage system, where nodes fail over time and are repaired to maintain a desired system reliability. They also introduced *regenerating codes* as codes that are more efficient than classical erasure codes for distributed storage applications. In many scenarios of interest, the data is required to exist in the system always in a systematic form. This has motivated the study of *exact regenerating codes* [9], [10], [11], [12] that achieve this goal by repairing a failed node with an exact copy of the lost data. The construction of exact regenerating codes in [9] turns out to be instrumental in achieving the secrecy and resiliency capacity of a DSS in the bandwidth-limited regime.

In [7], the construction of regenerating codes was linked to finding network codes for a suitable network. Network coding was introduced in the seminal paper of [13] and extends the classical routing approach by allowing the intermediate nodes in the network to encode their incoming packets as opposed to just copying and forwarding it. The literature on network coding is now rich in interesting results which can be found in references [14] and [15], that provide a comprehensive overview of this area.

In this paper, we are interested in securing distributed storage systems under repair dynamics, which is a special

case of the more general problem of achieving security in dynamical systems. A node-based intruder model is natural in this setting and is related to the recent work of [16] on securing distributed storage systems in the presence of a trusted verifier and that of Kosut et al. in [6] on protecting data in networks with traitor nodes. An intruder model that can observe and/or change the data on links, as opposed to nodes, has been extensively studied in the network coding literature. Cai and Yeung introduced in [17], [18] the problem of designing secure network codes in the presence of an eavesdropper, which was further studied in [19], [20], [21], [5]. A *Byzantine* adversary that can maliciously introduce errors on the network links was investigated in [22], [23], [24], [25], [26]. The problem of error correction in networks was also studied by Cai and Yeung in [27], [28] from a classical coding theory perspective. A different approach for correcting errors in networks was proposed by Koetter and Kschischang in [29], where communication is established by transmitting subspaces instead of vectors through the network. The use of maximum rank-metric codes for error control under this model was investigated in [30].

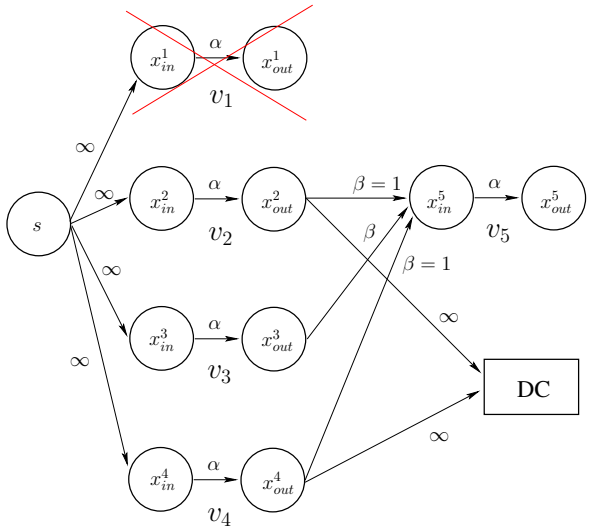

Fig. 2. The flow graph model of the DSS $\mathcal{D}(4,2,3)$ of Fig. 1 when node v_1 fails and is replaced by node v_5. Each storage node v_i is represented by two nodes x^i_{in} and x^i_{out} connected by an edge (x^i_{in}, x^i_{out}) of capacity α representing the node storage constraint. A data collector DC connecting to nodes v_2 and v_4 is also depicted.

III. Model

A. Distributed Storage System

A distributed storage system (DSS) is a dynamic network of storage nodes. These nodes include a source node that has an incompressible data file $\mathcal{F}$ of R symbols, or units, each belonging to a finite field $\mathbb{F}$. The source node is connected to n storage nodes $v_1, \ldots, v_n$, each having a storage capacity of α symbols, which may be utilized to save coded parts of the file $\mathcal{F}$. The storage nodes are individually unreliable and may fail over time. To guarantee a certain desired level of reliability, we assume that the DSS is required to always have n active, *i.e.*, non-failed, storage nodes that are simultaneously in service. Therefore, when a storage node fails, it is replaced by a new node with the same storage capacity α. The DSS should be designed in such a way as to allow any legitimate user or data collector, that contacts any k out of the n active storage nodes available at any given time, to be able to reconstruct the original file $\mathcal{F}$. We term this condition as the *reconstruction property* of distributed storage systems.

We assume that nodes fail one at a time [1], and we denote by v_{n+i} the new replacement node added to the system to repair the i-th failure. The new replacement node connects then to some d nodes, $d \geq k$, chosen, possibly randomly, out of the remaining active $n-1$ nodes and downloads γ units of data in total from them, which are then possibly compressed (if $\alpha < \gamma$) and stored on the node. The data stored on the replacement node can be different than the one that was stored on the failed node, as long as the reconstruction property of the DSS is retained. The process of replenishing redundancy to maintain the reliability of a DSS is referred to as the *"regeneration"* or *"repair"* process, and we call γ, the total amount of data (in symbols) downloaded for repair, the *repair bandwidth* of the system.

Due to load balancing and "fairness" requirements in the system, the repair process is typically *symmetric* where the new replacement node downloads equal amount of data, $\beta = \gamma/d$ units, from each of the node participating in the repair process. We will adopt the symmetric repair model throughout this paper. A distributed storage system $\mathcal{D}$ is thus characterized as $\mathcal{D}(n,k,d)$, where $k \leq d \leq n-1$. For instance, the DSS depicted in Fig. 1 corresponds to $\mathcal{D}(4,2,3)$ operating at the point $(\alpha, \gamma) = (2,3)$.

B. Flow Graph Representation

We adopt the same model as in [4] where the distributed storage system is represented by an information flow graph $\mathcal{G}$. The graph $\mathcal{G}$ is a directed acyclic graph with capacity constrained edges. It consists of three kinds of nodes: a single source node s, input storage nodes x^i_{in} and output storage nodes x^i_{out}, and data collectors DC_j for $i, j \in \{1, 2, \ldots\}$. The source node s holds an information source S having the file $\mathcal{F}$ as a special realization. Each storage node v_i in the DSS is represented by two nodes x^i_{in} and x^i_{out} in $\mathcal{G}$. To account for the storage capacity of v_i, these two nodes are joined by a directed edge (x^i_{in}, x^i_{out}) of capacity α (see Fig. 2).

The repair process that is initiated every time a failure occurs, causes the DSS, and consequently the flow graph, to be dynamic and evolving with time. At any given time, each node in the graph is either active or inactive depending on whether it has failed or not. The graph $\mathcal{G}$ starts with only the source node s and the nodes $x^1_{in}, \ldots, x^n_{in}$ connected respectively to the nodes $x^1_{out}, \ldots, x^n_{out}$. Initially, only the source node s is active and is connected to the storage input nodes $x^1_{in}, \ldots, x^n_{in}$ by outgoing edges of infinite capacity. From this point onwards,

[1] Multiple nodes failing simultaneously is a rare event. When this occurs, the DSS implements an "emergency" recovery process that employs a reserved set of trusted nodes, guaranteed not to be compromised. The trusted nodes then replace the failed ones by acting as data collectors and downloading data from k active nodes. The trusted nodes then consecutively leave the system, thus triggering multiple rounds of the repair process.

the source node s becomes and remains inactive, and the n input and output storage nodes become active. When a node v_i fails in a DSS, the corresponding nodes x_{in}^i and x_{out}^i become inactive in $\mathcal{G}$. If a replacement node v_j joins the DSS in the process of repairing a failure and connects to d active nodes $v_{i_1}, \ldots, v_{i_d}$, the corresponding nodes x_{in}^j and x_{out}^j with the edge (x_{in}^j, x_{out}^j) are added to the flow graph $\mathcal{G}$, and node x_{in}^j is connected to the nodes $x_{out}^{i_1}, \ldots, x_{out}^{i_d}$ by incoming edges of capacity $\beta = \gamma/d$ units each. A data collector is represented by a node connected to k active storage output nodes through infinite capacity links enabling it to download all their stored data and reconstruct the file $\mathcal{F}$. The graph $\mathcal{G}$ constitutes a multicast network with the data collectors as destinations. An underlying assumption here is that the flow graph corresponding to a distributed storage system depends on the sequence of failed nodes. As an example, we depict in Fig. 2 the flow graph corresponding to the DSS $\mathcal{D}(4, 2, 3)$ of the previous section (see Fig. 1) when node v_1 fails.

Let $\mathcal{V}$ be the set of nodes in the flow graph $\mathcal{G}$. A cut $C(V, \overline{V})$ in the flow graph separating the source s from a data collector DC_i is a partition of the node set of $\mathcal{G}$ into two subsets $V \subset \mathcal{V}$ and $\overline{V} = \mathcal{V} \setminus V$, such that $s \in V$ and $\text{DC}_i \in \overline{V}$. We say that an edge (n_1, n_2) belongs to a cut $C(V, \overline{V})$ if $n_1 \in V$ and $n_2 \in \overline{V}$. The *value* of a cut is the sum of the capacities of the edges belonging to it.

C. Intruder Model

We assume the presence of an illegitimate intruder in the DSS who can eavesdrop on some of the storage nodes, and possibly alter the stored data on some of them in order to sabotage the system. We characterize the power of an intruder by two parameters ℓ and b, where ℓ denotes the number of nodes that the intruder can eavesdrop on, and b denotes the number of nodes it can control by maliciously corrupting its data. We distinguish among three categories of intruders: a *passive eavesdropper* "Eve", an *active omniscient adversary* "Calvin", and an *active limited-knowledge adversary* "Charlie". We always assume that all the data collectors and intruders have the complete knowledge of the storage and the repair scheme implemented in the system.

a) Passive Eavesdropper: We assume that the eavesdropper Eve can access up to ℓ, $\ell < k$, nodes of her choice among all the storage nodes, $v_1, v_2, \ldots$, possibly at different time instances as the system evolves. Eve is passive and can only read the data on the observed ℓ nodes without modifying it, *i.e.*, $b = 0$. In the flow graph model, Eve is an eavesdropper that can access a fixed number ℓ of nodes chosen from the storage input nodes $x_{in}^1, x_{in}^2, \ldots$. Notice that while a data collector observes the output storage nodes, *i.e.*, the data stored on the nodes it connects to, Eve, has access to the input storage nodes, and thus can observe, in addition to the stored data, all the messages incoming to these nodes. As a result, Eve can choose some of the compromised ℓ nodes to be among the initial n storage nodes, and/or, if she deems it more profitable, she can wait for certain failures to occur and then eavesdrop on the replacement nodes by observing its downloaded data.

b) Active Omniscient Adversary: The active adversary Calvin is omniscient [24], *i.e.*, he knows the file $\mathcal{F}$ and the data stored on all the nodes. Moreover, Calvin can control b nodes in total, where $2b < k$, that can include some of the original nodes $v_1, \ldots, v_n$, and/or some replacement nodes $v_{n+1}, \ldots$. Calvin can maliciously alter the data stored on the nodes under his control. It can also send erroneous outgoing messages when contacted for repair or reconstruction. In the flow graph, this corresponds to controlling a set of b input nodes $\{x_{in}^{i_1}, x_{in}^{i_2}, \ldots, x_{in}^{i_b}\}$ and the corresponding output nodes $\{x_{out}^{i_1}, x_{out}^{i_2}, \ldots, x_{out}^{i_b}\}$.

c) Active Limited-knowledge Adversary: The active adversary Charlie is not *omniscient* but has *limited knowledge* about the data stored in the system. In particular, he has a limited eavesdropping capability ℓ not sufficient enough to know all the stored data. In addition, Charlie can control b nodes of his choice and maliciously corrupt their data. In distributed storage systems, an intruder controlling a node will also observe its data. Therefore, we assume that $b \leq \ell$, and that these b nodes are a subset of the ℓ eavesdropped nodes. In the flow graph, this corresponds to eavesdropping on some ℓ input nodes $\{x_{in}^{i_1}, \ldots, x_{in}^{i_\ell}\}$ and controlling a subset of size b of these nodes and the corresponding output nodes. A similar model was studied in [23], [24], [25] where the authors consider a limited-knowledge adversary that can eavesdrop and control *edges* rather than *nodes* in multicast networks.

IV. Results

The primary goal of this work is to secure distributed storage systems with repair dynamics in the presence of different types of intruders: passive eavesdropper, active omniscient adversary and active limited-knowledge adversary. We address the following issues:

- In the case of a passive eavesdropper, we study the *secrecy capacity* C_s of the DSS, *i.e.*, the maximum amount of data that can be stored on the DSS and delivered to a legitimate data collector without revealing any information about the data to the intruder.
- In the case of an active adversary, we study the *resiliency capacity* C_r of the DSS, *i.e.*, the maximum amount of data that can be stored on the DSS and reliably made available to a legitimate data collector.

For a DSS with symmetric repair, we provide upper bounds on the *secrecy* capacity and *resiliency* capacity. These bounds are maximized for the choice of repair degree $d = n - 1$. In this case, we provide explicit coding schemes that can achieve these bounds in the bandwidth-limited regime. Our results are summarized in Table I. We also show that for the active adversary controlling b nodes, our capacity achieving schemes can identify a list, of size at most $2b$ nodes, that is guaranteed to contain the malicious nodes. Thus, the system can be expurgated of these corrupt nodes, and thereby its resiliency to active adversaries is rejuvenated.

The upper bounds in Table I are based on cut arguments over the information flow graph representing the DSS [4]. Note that when there is no intruder, *i.e.*, $\ell = b = 0$, all the upper bounds in the second column of the Table I collapse to the DSS

Adversary Model	Upper bound $\gamma = d\beta$	Bandwidth limited regime (Γ) $d = n-1, d\beta = \Gamma$
Passive eavesdropper ($\ell < k$, $b = 0$)	$C_s(\alpha,\gamma) \leq \sum_{i=\ell+1}^{k} \min\{(d-i+1)\beta, \alpha\}$	$C_s^{BL}(\Gamma) = \sum_{i=\ell+1}^{k}(n-i)\beta$
Active omniscient adversary ($\ell = k$, $2b < k$)	$C_r(\alpha,\gamma) \leq \sum_{i=2b+1}^{k} \min\{(d-i+1)\beta, \alpha\}$	$C_r^{BL}(\Gamma) = \sum_{i=2b+1}^{k}(n-i)\beta$
Active limited-knowledge adversary(ℓ, $b \leq \ell$)	$C_r(\alpha,\gamma) \leq \sum_{i=b+1}^{k} \min\{(d-i+1)\beta, \alpha\}$	$C_r^{BL}(\Gamma) = \sum_{i=b+1}^{k}(n-i)\beta$

TABLE I

SUMMARY OF OUR CAPACITY RESULTS FOR A DSS $\mathcal{D}(n,k,d)$, WITH α UNITS OF STORAGE CAPACITY AT EACH NODE AND $\gamma = d\beta$ REPAIR BANDWIDTH. AN ADVERSARY IS CHARACTERIZED BY TWO PARAMETERS: ℓ, THE NUMBER OF NODES IT CAN EAVESDROP ON, AND b, THE NUMBER OF NODES IT CAN CONTROL. C_s AND C_r DENOTE THE SECRECY CAPACITY AND RESILIENCY CAPACITY, RESPECTIVELY. Γ IS THE UPPER LIMIT ON THE REPAIR BANDWIDTH FOR THE BANDWIDTH-LIMITED REGIME. NOTE THAT IF THE CONDITIONS ON ℓ, b SPECIFIED IN THE FIRST COLUMN ARE NOT SATISFIED, THEN C_s, C_r ARE EQUAL TO ZERO

capacity $M = \sum_{i=1}^{k} \min\{(d-i+1)\beta, \alpha\}$ which was derived in the original work of [4]. The upper bound on the secrecy capacity C_s, for the case of a passive eavesdropper can be explained intuitively by recognizing that when the DSS knows the identity of the ℓ compromised nodes it can discard them and avoid using them for storage. Hence, in the expression of the upper bound on C_s, we see a loss of ℓ terms in the summation as compared to the capacity with no intruder.

The upper bound on the resiliency capacity C_r, for the case of an active omniscient adversary, is similar to the one derived in [6] and can be regarded as a network version of the Singleton bound: a redundancy of $2b$ nodes is needed in order to correct the adversarial errors on b nodes. Whereas, a feasible strategy for the limited-knowledge adversary is to delete the data stored on the b nodes it controls rendering them useless resulting in the corresponding upper bound. Rigorous proofs of these results will be provided in the coming sections.

To get more insight into the above results for the bandwidth-limited case, we consider an asymptotic regime for the DSS where the number of nodes goes to infinity whereas the parameters k, ℓ and b are kept constant. We compute the ratios C_s^{BL}/M and C_r^{BL}/M, where M is the capacity of the DSS in the absence of any intruder. This ratio for the secrecy capacity is,

$$\frac{C_s^{BL}(\Gamma)}{M} = \frac{\sum_{\ell+1}^{k}(n-i)\beta}{\sum_{1}^{k}(n-i)\beta} \approx 1 - \frac{\ell}{k}, \tag{1}$$

as $n \to \infty$. Similarly, for the resiliency capacities, we have for omniscient adversary,

$$\frac{C_r^{BL}(\Gamma)}{M} \approx 1 - \frac{2b}{k}. \tag{2}$$

And for limited-knowledge adversary,

$$\frac{C_r^{BL}(\Gamma)}{M} \approx 1 - \frac{b}{k}. \tag{3}$$

Note that these asymptotic ratios are reminiscent of the capacity of the classical wiretap channel [31] in the case of a passive eavesdropper (1), the Singleton bound [32] in the case of omniscient adversary (2), and the capacity of the erasure channel [33] for the case of limited-knowledge adversary (3).

V. PASSIVE EAVESDROPPER

In this section, we consider a distributed storage system $\mathcal{D}(n,k,d)$ in the presence of a passive intruder "Eve". As described in Section III, Eve can eavesdrop on any $\ell < k$ storage nodes[2] of her choice in order to learn information about the stored file. However, Eve cannot modify the data on these nodes. We assume that Eve has complete knowledge of the storage and repair schemes implemented in the DSS. Next, we define the *secrecy capacity* of a DSS as the maximum amount of data that can be stored on a DSS under a *perfect secrecy* requirement, *i.e.*, without revealing any information about it to the eavesdropper.

A. Secrecy Capacity

Let S be a random variable uniformly distributed over $\mathbb{F}_q^R$ representing the incompressible data file of size R symbols at the source node, which is to be stored on the DSS. Thus, we have $H(S) = R$ (in base $\log_q$). Let $V_{in} := \{x_{in}^1, x_{in}^2, \dots\}$ and $V_{out} := \{x_{out}^1, x_{out}^2, \dots\}$ be the sets of input and output storage nodes in the flow graph, respectively. For each storage node v_i, let D_i and C_i be the random variables representing its downloaded messages and stored content respectively. Thus, C_i represents the data observed by a data collector DC when connecting to node v_i. If v_i is compromised while joining the DSS, Eve will observe all its downloaded data D_i, with $H(D_i) \leq \gamma$, and not only what it stores.

Let V_{out}^a be the collection of all subsets of V_{out} of cardinality k consisting of the nodes that are simultaneously active, *i.e.*, not failed, at a certain instant in time. For any subset B of V_{out}, define $C_B := \{C_i : x_{out}^i \in B\}$. Similarly for any subset E of V_{in}, define $D_E := \{D_i : x_{in}^i \in E\}$. The reconstruction property at the data collector can be written as

$$H(S|C_B) = 0 \quad \forall B \in V_{out}^a, \tag{4}$$

and the perfect secrecy condition implies

$$H(S|D_E) = H(S) \quad \forall E \subset V_{in} \text{ and } |E| \leq \ell. \tag{5}$$

[2]When Eve observes $\ell \geq k$ the secrecy capacity of the system is trivially equal to zero since Eve can implement the data collector's scheme to recover all the stored data.

Given a DSS $\mathcal{D}(n,k,d)$ with ℓ compromised nodes, its secrecy capacity, denoted by $C_s(\alpha,\gamma)$, is then defined to be the maximum amount of data that can be stored in this system such that the reconstruction property in (4) and the perfect secrecy condition in (5) are simultaneously satisfied for all possible data collectors and eavesdroppers, *i.e.*,

$$C_s(\alpha,\gamma) := \sup_{\substack{H(S|C_B)=0 \quad \forall B \\ H(S|D_E)=H(S) \quad \forall E}} H(S), \tag{6}$$

where $B \in V^a_{out}$, $E \subset V_{in}$ and $|E| \leq \ell$.

B. Special Cases

Before we proceed to the general problem of determining the secrecy capacity of a DSS, we analyze two special cases that shed light on the general problem.

1) Static Systems: A static version of the problem studied here corresponds to a DSS with ideal storage nodes that do not fail, and hence there is no need for repair in the system. The flow graph of this system constitutes then a well-known multicast network studied in network coding theory called the combination network [15, Chap. 4]. Therefore, the static storage problem can be regarded as a special case of wiretap networks [18], [20], or equivalently, as the erasure-erasure wiretap-II channel studied in [34]. The secrecy capacity for such systems is equal to $(k-\ell)\alpha$, and can be achieved using either the nested MDS codes of [34] or the coset codes of [20], [31].

Even though the above proposed solution is optimal for the static case, it can have a very poor security performance when applied directly to dynamic storage systems experiencing failures and repairs. For instance, consider the straightforward way of repairing a failed node by downloading the whole file and regenerating the lost data. In this case, if Eve observes the new replacement node while it is downloading the whole file, she will be able to reconstruct the entire original data. Hence, no secrecy scheme will be able to hide any part of the data from Eve, and the secrecy rate would be zero.

The case of static systems highlights the new dimension that the repair process brings into the secrecy picture of distributed storage systems. The dynamic nature of the DSS renders it intrinsically different from the static counterpart making the repair process a key factor that should be carefully designed in order not to jeopardize the whole stored data.

2) Systems Using Random Network Coding: Using the flow graph model, the authors of [4] showed that *random linear network codes* over a large finite field can achieve any point (α,γ) on the optimal storage-repair bandwidth tradeoff curve with a high probability. Consider an example of a random linear network code used in a compromised DSS $\mathcal{D}(4,3,3)$ which stores a file of size $R=6$ symbols with $\beta=1$, *i.e.*, $\gamma = d\beta = 3$, and $\alpha = 3$. From [4], it can be shown using the max-flow min-cut theorem that the maximum file size that can be stored on this DSS is equal to 6 symbols. In this case, each of the initial nodes $v_1,\ldots,v_4$ store 3 independently generated random linear combinations of the 6 information symbols. Assume now that node v_4 fails (see Fig. 3) and is replaced by a new node v_5 that connects to v_1, v_2, v_3 and downloads from each $\beta = 1$ random linear combination of their stored data. Now suppose that node v_5 fails after some time and is replaced by node v_6 in a similar fashion. If $\ell = 2$ and Eve had accessed nodes v_5 and v_6 while they were being repaired, it would observe 6 random linear equations of the data symbols. Since the underlying field is typically of large size, the 6 linear equations observed by Eve are linearly independent with high probability. Hence, she will be able to reconstruct the whole file, and the secrecy rate here is equal to 0. Later in Example 3 we present a scheme that achieves a secrecy rate of 1 unit for this DSS.

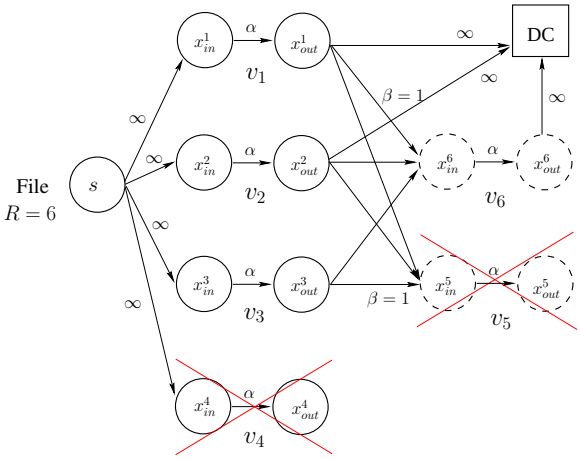

Fig. 3. The DSS $\mathcal{D}(4,3,3)$ with $(\alpha,\gamma)=(3,3)$, *i.e.*, $\beta=1$. Eve can observe $\ell=2$ nodes. Node v_4 fails and is replaced by node v_5, which fails in turn after some time and is replaced by node v_6. Nodes v_5 and v_6 are compromised and shown with broken boundaries. If random network coding is used and Eve observes nodes v_5 and v_6 during repair, it will be able to decode all the stored data with a high probability.

While random network codes are appealing for use in distributed storage systems due to their decentralized nature and low complexity, the above analysis shows that this may not always be the case for achieving security. This is also in contrast with the case of multicast networks where an intruder can observe a fixed number of edges instead of nodes [18], wherein, random network coding performs as good as any deterministic secure code [21].

C. Results on Passive Eavesdropper

We present here our two main results for the compromised DSS with *passive eavesdropper*:

Theorem 1: [Secrecy Capacity Upper Bound] For a distributed storage system $\mathcal{D}(n,k,d)$, with $\ell < k$ compromised nodes, the secrecy capacity is upper bounded by

$$C_s(\alpha,\gamma) \leq \sum_{i=\ell+1}^{k} \min\{(d-i+1)\beta, \alpha\}, \tag{7}$$

where $\beta = \gamma/d$.

In the bandwidth-limited regime, we have a constraint on the repair bandwidth $\gamma \leq \Gamma$, while no constraint is imposed on the node storage capacity α. The secrecy capacity in this

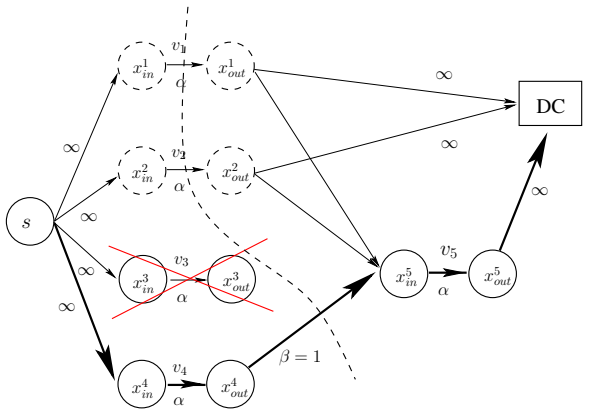

Fig. 4. The flow graph of the DSS $\mathcal{D}(4,3,3)$ with $(\alpha,\gamma)=(3,3), \beta=1$ and $\ell = 2$. Node v_3 fails and is replaced by node v_5. Nodes v_1, v_2 are compromised to Eve and are shown with broken boundaries. A data collector DC connects to nodes v_1, v_2, v_5 to retrieve the data file. The data collector can get at most one unit of information securely on the path $(s, x^4_{in}, x^4_{out}, x^5_{in}, x^5_{out}, \text{DC})$ which is not observed by Eve.

regime is thus defined as

$$C_s^{BL}(\Gamma) := \sup_{\substack{\gamma \le \Gamma \\ \alpha \ge 0}} C_s(\alpha,\gamma) \tag{8}$$

$$\le \sup_{\gamma \le \Gamma} \sum_{i=\ell+1}^{k} (d-i+1)\beta. \tag{9}$$

The last inequality follows from Theorem 1 by setting $\alpha = \Gamma$. When the parameter d is a system design choice, the maximum in the above optimization is attained at $d^* = n-1$. In Section V-D, we demonstrate a scheme that achieves this upper bound, thereby establishing the following theorem.

Theorem 2: [Secrecy Capacity: Bandwidth-Limited Regime] For a distributed data storage system $\mathcal{D}(n,k,d)$ with $d = n-1$ and $\ell < k$ compromised nodes, the secrecy capacity in the bandwidth-limited regime is given by

$$C_s^{BL}(\Gamma) = \sum_{i=\ell+1}^{k} (n-i)\beta,$$

where $\beta = \frac{\Gamma}{n-1}$ and can be achieved for a node storage capacity $\alpha = \Gamma$.

Before we proceed to prove the above theorems, we consider an example that gives insights into the proof techniques.

Example 3: Consider again the DSS $\mathcal{D}(4,3,3)$ operating at $\alpha = 3, \beta = 1$ and $\ell = 2$ of Section V-B2. We show first that the upper bound on the secrecy capacity of this system is 1 as given by Theorem 1, and then provide a scheme that achieves it.

To obtain the upper bound on the secrecy capacity, consider the flow graph of this DSS shown in Fig. 4 where nodes v_1 and v_2 are compromised and observed by Eve. Suppose that node v_3 fails and is replaced by v_5 that downloads $\beta = 1$ unit of information from each of the $d = 3$ nodes v_1, v_2, v_4. We focus now on a data collector that connects to the three nodes v_1, v_2 and v_5 to reconstruct the source file. Even if the source node s and the data collector knew the location of the eavesdropper, it can get at most one unit of secure information by ignoring all the information received from the compromised nodes. The data can only be conveyed securely through the path $(s, x^4_{in}, x^4_{out}, x^5_{in}, x^5_{out}, \text{DC})$, that has a "bottleneck" edge (x^4_{out}, x^5_{in}) with capacity $\beta = 1$ unit. Since our analysis is based on a worst case scenario, this gives an upper bound of 1 unit on the secrecy capacity. This bound can be reinterpreted as taking the minimum value of a cut separating the source s from any data collector in the flow graph after deletion of any two nodes. This argument can be generalized to any DSS $\mathcal{D}(n,k,d)$ by finding an upper bound on the value of the min-cut in the flow graph after deleting ℓ nodes. Thus, we obtain the upper bound of Theorem 1 whose detailed proof is provided in Appendix A.

Before we provide a coding scheme that achieves the previous upper bound, we define the *nested MDS codes* [34] which will be an important building block in our code construction.

Definition 4 (Nested MDS Codes): An (n,k) MDS code with generator matrix G is called nested if there exists a positive integer $k_0 < k$ such that $G = \begin{bmatrix} G_1 \\ G_2 \end{bmatrix}$, with G_1, of dimensions $(k_0 \times n)$, itself is a generator matrix of an (n,k_0) MDS code.

Our proposed capacity-achieving code is depicted in Fig. 5 and consists of the concatenation of an outer nested MDS code with a special inner repetition code that was introduced in [9] for constructing exact regeneration codes. Let $S \in \mathbb{F}_q$ denote the information symbol that is to be securely stored on the system and $\mathcal{K} = [K_1 \ \ldots \ K_5]$ be a vector of independent random keys each uniformly distributed over $\mathbb{F}_q$. The MDS coset code is chosen to be a nested MDS code [34] with its generator matrix given by $G := \begin{bmatrix} G_K \\ G_S \end{bmatrix}$, where

$$G_K = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \text{ and}$$

$$G_S = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Note that the matrix $G := \begin{bmatrix} G_K \\ G_S \end{bmatrix}$ a generator of a $(6,6)$ MDS code and the sub-matrix G_K is a generator of an $(6,5)$ MDS code ($k_0 = 5$). Hence, the code generated by G is a nested MDS code. Set, $Z = S + \sum_{i=1}^{5} K_i$, then the codeword X given by

$$X = \begin{bmatrix} \mathcal{K} & S \end{bmatrix} \begin{bmatrix} G_K \\ G_S \end{bmatrix}, \tag{10}$$

can be written as $X = \begin{bmatrix} Z & K_1 & \ldots & K_5 \end{bmatrix}$. The encoded symbols $Z, K_1, \ldots, K_5$ are then stored on the nodes $v_1, \ldots, v_4$ as shown in Fig. 5, following the special repetition code of Rashmi et al [9], which we henceforth refer to as *RSKR-repetition code*.

In the RSKR-repetition code used here, nodes $v_1, \ldots, v_4$ store respectively $\{Z, K_1, K_2\}$, $\{Z, K_3, K_4\}$, $\{K_1, K_3, K_5\}$ and $\{K_2, K_4, K_5\}$. Since $d = 3$, in the case of a failure

the new replacement node contacts all the 3 remaining active nodes in the system and recovers an exact copy of the lost data. For example, when node v_1 fails the new replacement node connects to nodes v_2, v_3 and v_4 and downloads the symbols Z, K_1 and K_2 from each, respectively. It can also be checked that a data collector connecting to any 3 nodes observes all the symbols $Z, K_1, \ldots, K_5$ and hence can decode the information symbol S as $S = Z - \sum_{i=1}^{5} K_i$. However, an eavesdropper accessing any two nodes will observe some subset of 5 symbols out of 6, and therefore cannot obtain any information about S.

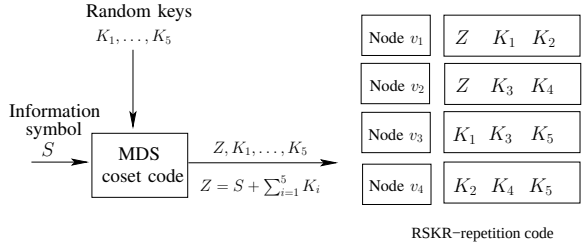

Fig. 5. A schematic representation of the optimal code for the DSS $\mathcal{D}(4,3,3)$, operating at $(\alpha,\gamma) = (3,3)$ with $\ell = 2$, that achieves the secrecy capacity of 1 unit. The information symbol S and 5 independent random keys are mixed appropriately using an MDS coset code. The encoded symbols are then stored on the DSS using the RSKR-repetition code. An eavesdropper observing any $\ell = 2$ nodes cannot get any information about the stored symbol S.

In the following section, we provide a generalization of the code in this example, and show that it achieves the secrecy capacity of DSS for $d = n-1$ in the bandwidth-limited regime, thus proving Theorem 2.

D. Secrecy Capacity in the Bandwidth-Limited Regime

The special cases studied in Section V-B pointed out that the main difficulty in determining the secrecy capacity of distributed storage systems is due to its dynamic nature. We will demonstrate that in the bandwidth-limited regime for $d = n-1$, with a careful choice of code, it is possible to transform the problem of secrecy over a dynamic DSS into a static problem of secrecy over a point to point channel equivalent to the erasure-erasure wiretap channel-II in [34]. Then, we show that using nested MDS codes at the source one can achieve the secrecy capacity of the equivalent wiretap channel.

Our approach builds on the results of [9] where the authors constructed a family of exact regenerating codes for the DSS $\mathcal{D}(n,k,d)$ with $d = n-1, \alpha = d\beta$. The "exact" property of these codes allows any repair node to reconstruct and store an identical copy of the data lost upon a failure. The code construction in [9] consists of the concatenation of an MDS code with the RSKR-repetition code. This construction is instrumental for obtaining codes that can achieve the secrecy capacity by carefully choosing the outer code to be a nested MDS coset code as was done in Example 3.

For simplicity, we will explain the code for $\beta = 1$, *i.e.*, $\Gamma = n-1$. For any larger values of Γ, and in turn of β, the file can be split into chunks, each of which can be separately encoded using the construction corresponding to $\beta = 1$. Since the DSS

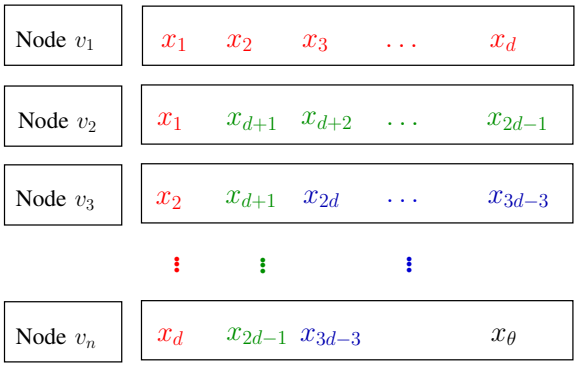

Fig. 6. The structure of the RSKR-repetition code of Rashmi et al [9] for n storage nodes, $\alpha = d = n-1, \beta = 1$ and $\theta = \frac{n(n-1)}{2}$. The RSKR-repetition code stores 2 copies of each coded symbol, *i.e.,* the total number of stored symbols is $nd = 2\theta$.

is operating in the bandwidth-limited regime with no constraint on the node storage capacity, we choose $\alpha = \Gamma$. From [4], we know that for a DSS $\mathcal{D}(n,k,d = n-1)$ with $\alpha = n-1, \beta = 1$ the capacity in the absence of an intruder ($\ell = 0$) is $M = \sum_{i=1}^{k}(n-i)$. Let $R := \sum_{i=\ell+1}^{k}(n-i)$ be the maximum number of information that we could store securely on the DSS, and $\theta := \frac{n(n-1)}{2}$. Let $S = (s_1, \ldots, s_R) \in \mathbb{F}_q^R$ denote the information file and $\mathcal{K} = (K_1, \ldots, K_{M-R}) \in \mathbb{F}_q^{M-R}$ denote $M-R$ independent random keys each uniformly distributed over $\mathbb{F}_q$. Then, the proposed code consists of an outer (θ, M) nested MDS code (see (10)) which takes S and $\mathcal{K}$ as an input and outputs $X = (x_1, \ldots, x_\theta)$, as,

$$X \;=\; \begin{bmatrix} \mathcal{K} & S \end{bmatrix} \begin{bmatrix} G_K \\ G_S \end{bmatrix},$$

where, $G := \begin{bmatrix} G_K \\ G_S \end{bmatrix}$ is a generator matrix of a (θ, M) MDS code such that G_K itself is a generator matrix of a $(\theta, M-R)$ MDS code. This outer (θ, M) nested MDS code is then followed by an inner RSKR-repetition code which stores the codeword X on the DSS following the pattern depicted in Fig. 6.

The RSKR-repetition codes were introduced in [9] as a method for constructing exact regenerating codes for a distributed storage system. These codes consist of "filling" the storage nodes $v_1, \ldots, v_n$ successively, by repeating "vertically" (*i.e*, across all the nodes) the data stored "horizontally" (*i.e.*, on a single storage node), as shown in Fig. 6. This procedure can be described using an auxiliary complete graph over n vertices $u_1, \ldots, u_n$ that consists of θ edges. Suppose the edges are indexed by the coded symbols $x_1, \ldots, x_\theta$. The code then consists of storing on node v_i the indices of the edges adjacent to vertex u_i in the complete graph. As a result, the RSKR-repetition code has a special property that every coded symbol x_i is stored on exactly two storage nodes, and any pair of two storage nodes have exactly one coded symbol in common. This property along with the fact that the repair degree $d = n-1$, enables the exact repair of any failed node in the DSS as it was explained in Example 3.

The use of the RSKR-repetition code transforms the dy-

namic storage system into a static point-to-point channel as explained below. Notice first that since $\Gamma = \alpha = n-1$, all the data downloaded during the repair process is stored on the new replacement node without any further compression[3]. Thus, accessing a node during repair, *i.e.*, observing its downloaded data, is equivalent to accessing it after repair, *i.e.*, only observing its stored data. Second, the RSKR-repetition code restore the replacement node with an exact copy of the lost data. Therefore, even though there are failures and repairs, the data storage system looks exactly the same at any point of time: any data collector downloads M symbols out of $x_1, \ldots, x_\theta$ by contacting k nodes, and any eavesdropper can observe $\mu = \sum_{i=1}^{\ell}(d-i+1) = M - R$ symbols. Thus, the system becomes similar to the erasure-erasure wiretap channel-II of parameters (θ, M, μ)[4]. Therefore, since the outer code is a nested MDS code, from [34] we know that it can achieve the secrecy capacity of the erasure-erasure wiretap channel which is equal to $M - \mu$. Hence for the DSS, our codes achieve the secrecy rate of

$$M - (M - R) = R = \sum_{i=\ell+1}^{k} (n-i).$$

This rate corresponds to $\beta = 1$. For the general case when $\beta = \Gamma/(n-1)$, the total secrecy rate achieved is,

$$\sum_{i=\ell+1}^{k} (n-i)\beta,$$

thus completing the proof of Theorem 2.

VI. Active Omniscient Adversary

In this section we study distributed storage systems in the presence of an active adversary "Calvin" that can control up to b nodes. Calvin can choose to control any b nodes among all the storage nodes, $v_1, v_2, \ldots$, and possibly at different time instances as the system evolves in time due to failures and repairs. Moreover, Calvin is assumed to be omniscient ($l = k$), so he knows the source file $\mathcal{F}$. Moreover, since he has complete knowledge of the storage and repair schemes, he knows the content stored on each node in the system. Under this setting, we define the *resiliency capacity* of a DSS as the maximum amount of data that can be stored on the DSS and delivered reliably to any data collector that contacts any k nodes in the system.

Example 5: Consider again our example of the DSS $\mathcal{D}(4,3,3)$ with $\alpha = \gamma = 3$. Assume that there is an omniscient active adversary Calvin that can control one storage node, *i.e.*, $b = 1$, and can modify its stored data and/or its messages outgoing to data collectors and repair nodes.

A first approach for finding a scheme to reliably store data on this DSS would be to use the results in the network coding literature [24], [27], [28], [29] on the capacity of multicast networks in the presence of an adversary that can control t edges of unit capacity each. It is shown there that the resiliency capacity of these networks is equal to $\Omega - 2t$, where Ω is the capacity of the multicast network in the absence of the adversary. This resiliency capacity can be achieved by overlaying an error-correction code such as a Maximum Rank Distance (MRD) code [21] on top of the network at the source. This approach turns out to be not very useful here. In fact, the capacity in the absence of Calvin is 6 (see [4]), and $b = 1$ corresponds to $t = \alpha = 3$. Hence, the above approach will achieve a storage rate of $6 - 2t = 0$.

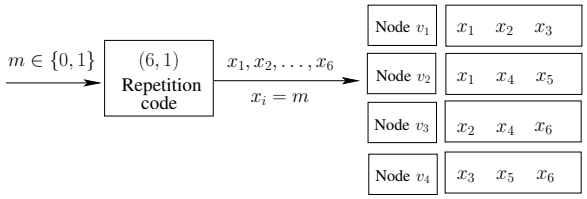

Fig. 7. A coding scheme for storing 1 bit reliably on the DSS $\mathcal{D}(4,3,3)$ with $\alpha = 3$ bits and $\beta = 1$, in the presence of an omniscient adversary Calvin who controls $b = 1$ node.

We now give a coding scheme that can reliably store 1 bit of information for the DSS. Later, we show that this is also the best that can be done, *i.e.*, the resiliency capacity of this DSS is equal to 1 unit. The proposed code is formed by concatenating a $(6,1)$ repetition code with an RSKR-repetition code as shown in Fig 7. The repair process is that of the RSKR-repetition codes described in Section V-D. When a node fails, the replacement node recovers the lost bits by downloading the bits with same indices from the remaining three active nodes.

Any data collector contacting three nodes will observe 9 bits. In the static case, when no failure or repair occur, only 3 bits (the ones stored on the compromised node) among the 9 bits observed by the data collector may be erroneous. In that case, the DC can perform a majority decoding to recover the information bit. However, in the dynamic model, the DC can receive up to 5 erroneous bits. To show how this may occur, assume that the DSS is storing the all-zero codeword, *i.e.*, $x_i = 0$ for $i = 1, \ldots, 6$, in Fig. 7, corresponding to the message $m = 0$. Suppose that node v_1 is the one that is compromised and controlled by the adversary Calvin as shown in Fig. 8. Assume that Calvin changes all the 3 stored bits (x_1, x_2, x_3) on node v_1, from $(0,0,0)$ to $(1,1,1)$ and also sends the erroneous bit "1" whenever v_1 is contacted for repair. Now suppose that node v_2 fails and it is replaced by node v_5 which, based on the RSKR-repetition structure, downloads bits $x_1 = 1, x_4 = 0$ and $x_5 = 0$ from nodes v_1, v_3 and v_4 respectively. Suppose also that, after some period of time, node v_3 fails and is replaced by node v_6 which downloads bits $x_2 = 1, x_4 = 0$ and $x_6 = 0$ from nodes v_1, v_4 and v_5 respectively. An important point to note here is that our repair scheme is fixed and is based on the RSKR-repetition structure irrespective of the possible errors in the bits downloaded during the repair process. As a result a data collector that contacts nodes v_1, v_5 and v_6 observes the data as shown in the table in Fig. 8 which includes 5 errors.

[3]This corresponds to the *Minimum Bandwidth Regenerating* (MBR) codes described in [4].

[4]In the erasure-erasure wiretap channel-II of parameters (θ, M, μ), the transmitter sends θ symbols through an erasure channel to a legitimate receiver that receives M symbols. The eavesdropper can observe any μ symbols out of the transmitted M [34].

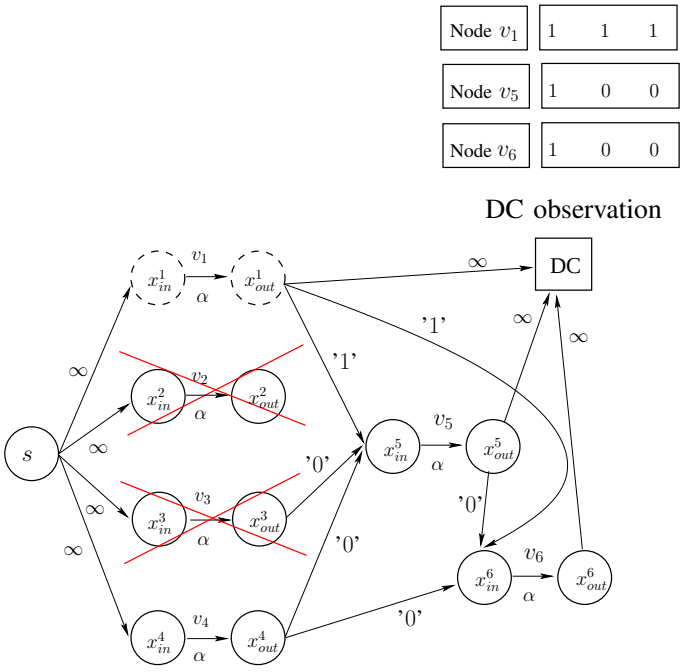

Fig. 8. Node v_1 with broken boundary is compromised and controlled by an omniscient adversary Calvin. Nodes v_2 and v_3 fail, and are replaced by nodes v_5, v_6 respectively. The all-zero codeword corresponding to message $m = 0$ is stored on the DSS. The Data collector DC connecting to nodes v_1, v_5 and v_6 observes a total number of 9 bits out of which 5 bits are erroneous and equal to "1" as shown in the table above.

In a worst case scenario, Calvin will be able to corrupt all the bits in the DSS having the same indices as the bits stored on the nodes it controls (here the bits with labels x_1, x_2 and x_3). Therefore, Calvin can introduce at most 5 erroneous bits on a collection of $k = 3$ nodes which may be observed by a data collector. In this case, a majority decoder, or equivalently a minimum Hamming distance decoder, will not be able to decode to the correct message.

To overcome this problem, we exploit the fact that Calvin controls only one node, so he can introduce errors only in specific patterns, to design a special decoder that will always decode to the correct message m irrespective of Calvin's adversarial strategy. In fact, for any possible choice of the compromised node, one of the following four sets $T_1 = \{x_4, x_5, x_6\}, T_2 = \{x_2, x_3, x_6\}, T_3 = \{x_1, x_3, x_5\}$ and $T_4 = \{x_1, x_2, x_4\}$ is a *trusted set* that only contains symbols that were not altered by Calvin. For example, when Calvin controls v_1, the trusted set is T_1. The proposed decoder operates in the following way. First, it finds a set $T^* \in \{T_1, \ldots, T_4\}$ whose elements all agree to either 0 or 1. Then, it declares accordingly that message $m = 0$ or $m = 1$ was stored. This decoder will always decode to the correct message since each set T_i intersects with every other set $T_j, j \neq i$, in exactly one symbol and one of them is a trusted set. Therefore each set T_i contains at least one symbol which is unaltered by Calvin. Thus, if all the symbols in T_i agree, they will agree to the correct message.

A. Results on Omniscient Adversary

In [6], the resiliency capacity of unicast networks with a single compromised node was analyzed and a cut-set upper bound was derived. In the following, Theorem 6 generalizes the bound in [6] for the case of distributed storage systems, where $b \geq 1$ nodes are controlled by an omniscient adversary.

Theorem 6: [Resiliency Capacity Upper Bound] Consider a distributed storage system DSS $\mathcal{D}(n, k, d)$. If an omniscient adversary controls any $b \geq 1$ nodes, with $2b < k$, the resiliency capacity $C_r(\alpha, \gamma)$ is upper bounded as,

$$C_r(\alpha, \gamma) \leq \sum_{i=2b+1}^{k} \min\{(d - i + 1)\beta, \alpha\}, \qquad (11)$$

where $\beta = \gamma/d$. If $2b \geq k$, then $C_r(\alpha, \gamma) = 0$.

This bound is a network version of the Singleton bound and is obtained by computing the value of certain cuts in the flow graph of the DSS after the deletion of $2b$ nodes. The detailed proof of the above theorem is given in Appendix B.

The resiliency capacity in the bandwidth-limited regime is defined as

$$C_r^{BL}(\Gamma) := \sup_{\substack{\gamma \leq \Gamma \\ \alpha \geq 0}} C_r(\alpha, \gamma),$$

where Γ is the upper limit on the total repair bandwidth. We again note that if the parameter d is a system design choice, the upper bound of Eq. (11) in the bandwidth-limited regime is maximized for $d = n - 1$. In the following section we exhibit a scheme that achieves this upper bound. This result is summarized in Theorem 7.

Theorem 7: Consider a distributed storage system $\mathcal{D}(n, k, d = n - 1)$ operating in the bandwidth-limited regime. If an omniscient adversary controls b nodes, with $2b < k$, the resiliency capacity of the DSS is given by

$$C_r^{BL}(\Gamma) = \sum_{i=2b+1}^{k} (n - i)\beta, \qquad (12)$$

where $\beta = \frac{\Gamma}{n-1}$ and can be achieved for a node storage capacity $\alpha = \Gamma$. If $2b \geq k$, then $C_r^{BL}(\Gamma) = 0$.

B. Resiliency Capacity in the Bandwidth-Limited Regime

Similar to the proof of Theorem 2, it suffices to show the achievability for $\beta = 1$, *i.e.*, $\Gamma = n - 1$. In this case, our capacity achieving code uses a node storage capacity $\alpha = n-1$ symbols.

The code has a similar structure to the scheme used in Section V for the case of a passive adversary and is a generalization of the code used in Example 5. The $(6, 1)$ repetition code in the example is replaced by an (θ, R) MDS code where $R := C_r(n - 1) = \sum_{i=2b+1}^{k}(n - i)$ and $\theta = \frac{n(n-1)}{2}$. In the second layer, the output of the MDS code is stored on the DSS following the RSKR-repetition structure as in Fig 6. As explained in Example 5, node failures are repaired using the RSKR-repetition structure (also see Section V for additional details) irrespective of the possible errors introduced by Calvin. Notice that the MDS code used here has a rate

lower then the one used in the passive adversary case in Section V-D to allow for correcting the errors introduced by the adversary.

A data collector accessing any k nodes will observe a total of $\alpha k = (n-1)k$ symbols, out of which $M = \sum_{i=1}^{k}(n-i)$ symbols have distinct indices, and $\frac{k(k-1)}{2}$ symbols are repeated due to the RSKR-repetition code. The adversary can corrupt identically the two copies of each symbol stored on the b controlled nodes. Therefore, the data collector focuses on M symbols with distinct indices out of $(n-1)k$ and uses them for decoding. These M symbols with distinct indices form a codeword of an (M, R) MDS code, say $\mathcal{X}$, which are possibly corrupted by the errors introduced by the adversary. The minimum distance of the MDS code $\mathcal{X}$ is,

$$d_{\min}(\mathcal{X}) = M - R + 1 = \sum_{i=1}^{2b}(n-i) + 1. \tag{13}$$

The adversary that controls b nodes can introduce up to $t = \sum_{i=1}^{b}(n-i)$ errors in the set of M symbols with distinct indices. A simple manipulation shows that $t > \lfloor \frac{d_{\min}(\mathcal{X})-1}{2} \rfloor$. Therefore, a classical minimum distance decoder for $\mathcal{X}$ will not be able to recover the original file. Thus, the minimum distance decoder fails for this specific adversarial strategy where Calvin corrupts the repeated symbols identically and cannot be used for a general adversarial strategy.

Next, we present a novel decoder that can correct errors beyond the classical upper bound of $\lfloor \frac{d_{\min}(\mathcal{X})-1}{2} \rfloor$ in the DSS. The main idea is to take advantage of the special structure of the error patterns that can be introduced by the adversary.

First, we introduce two definitions that will be useful in describing the decoding algorithm and that will serve as a generalization of the concept of trusted set in the previous example.

Definition 8: **Puncturing a vector:** Consider a vector $\vec{v} \in \mathbb{F}^N$ for some field $\mathbb{F}$. Let $I \subset \{1, 2, \ldots, N\}, |I| = p$, be a given set. Then *puncturing* vector $\vec{v}$ with pattern I corresponds to deleting the entries in $\vec{v}$ indexed by the elements in I to obtain a vector $\vec{v}_I \in \mathbb{F}^{N-p}$.

Definition 9: **Puncturing a Code:** Consider a code $\mathcal{C}$ in $\mathbb{F}^N$. Let $I \subset \{1, 2, \ldots, N\}, |I| = p$, be a given set. The *punctured code* $\mathcal{C}_I$ is obtained by *puncturing* all the codewords of $\mathcal{C}$ with pattern I, *i.e.*,

$$\mathcal{C}_I := \{\vec{x}_I | \vec{x} \in \mathcal{C}\}.$$

Proposition 10: If $\mathcal{C}$ is an MDS code with parameters (n, k) then for any given fixed pattern $I \subset \{1, 2, \ldots, n\}, |I| = p < (n-k+1)$, the punctured code $\mathcal{C}_I$ is also an MDS code with parameters $(n-p, k)$.

Decoding Algorithm: Let $B, |B| \leq b$, denote the set of storage nodes controlled by the adversary. Because of the exact repair property of the RSKR-repetition codes, it is sufficient to focus on the case when $B \subset \{v_1, \ldots, v_n\}$ with $|B| = b$. For each such set B, we define $I_B \subset \{1, 2, \ldots, \theta\}$ to be the set of the indices of the symbols stored on the nodes in B. For instance, in Example 5, if $B = \{v_1\}$, $I_B = \{1, 2, 3\}$.

The decoding algorithm proceeds in the following way:

1) The data collector connecting to k nodes selects any M symbols with distinct indices, out of the $(n-1)k$ observed symbols, as its input $Y \in \mathbb{F}_q^M$ for decoding. In Example 5, Fig. 8, the DC connecting to nodes v_1, v_5, v_6 observes vector $(y_1, y_2, y_3, y_1, y_4, y_5, y_2, y_4, y_6)$. After removing the repeated symbols, we get $Y = (y_1, y_2, y_3, y_4, y_5, y_6)$. Note for a fixed DC, Y is a codeword of an (M, R) MDS code which we call $\mathcal{X}$. Y includes possible errors introduced by the adversary. The code $\mathcal{X}$ itself is a punctured code of the outer (θ, R) MDS code.
2) For each $B \subset \{v_1, \ldots, v_n\}, |B| = b$, find I_B.
3) Puncture Y and the code $\mathcal{X}$ with pattern I_B to obtain the observed word Y_{I_B} and punctured code $\mathcal{X}_{I_B}$. Note that due to the RSKR-repetition structure, the size of such puncturing pattern is
$$|I_B| = \sum_{i=1}^{b}(n-i)$$
which is less than the minimum distance of the MDS code $\mathcal{X}$ (see (13)). Hence, by Proposition 10 $\mathcal{X}_{I_B}$ is an MDS code.
4) Let $H_{\mathcal{X}_{I_B}}$ be the parity check matrix of the punctured code $\mathcal{X}_{I_B}$. Compute the syndrome of the observed word Y_{I_B} as
$$\vec{\sigma}_{I_B} = H_{\mathcal{X}_{I_B}} Y_{I_B}^T.$$
5) If $\vec{\sigma}_{I_B} = 0$, then Y_{I_B} is a codeword of $\mathcal{X}_{I_B}$. Assume it to be a trusted codeword and decode to message using the code $\mathcal{X}_{I_B}$.

Proof of Correctness: We now prove the correctness of the above decoding algorithm by showing that it will always correct the errors introduced by the adversary and output the correct message. Notice first that the syndrome $\vec{\sigma}_{I_B}$ will always be equal to zero whenever $B = B^*$, the actual set of nodes controlled by the adversary (which is not known to the data collector). Therefore, the above decoding algorithm will always give an output. Next, we show that this output always corresponds to the correct message stored on the DSS. Denote by X the true codeword in $\mathcal{X}$, that would have been observed by the DC in the absence of Calvin. Let B^* be the set of the b traitor nodes. Then, the proposed decoding algorithm fails iff there exists some other set $B \neq B^*$, and some other codeword $X' \in \mathcal{X}$, s.t. $X' \neq X$, for which $Y_{I_B} = X'_{I_B} \in \mathcal{X}_{I_B}$. This implies that

$$X_{I_{B^* \cup I_B}} = X'_{I_B \cup I_{B^*}}. \tag{14}$$

But, from the RSKR-repetition code structure we know

$$|I_{B^*} \cup I_B| \quad \leq \quad \sum_{i=1}^{2b}(n-i). \tag{15}$$

Equations (14) and (15) imply that $d_{\min}(\mathcal{X}) \leq \sum_{i=1}^{2b}(n-i)$ which contradicts equation (13).

Remark 11 (Decoder complexity): The complexity of the proposed decoder is exponential in the number b of malicious nodes. Therefore, it is not practical for systems with large

values of b. However, this decoder can be regarded as a proof technique for the achievability of the resiliency capacity C_r^{BL} of Theorem 7.

Remark 12: [Expurgation of malicious nodes] As shown above, the proposed decoder always decodes to the correct message, and thus, can identify the indices of any erroneous symbols. The data collector can then report this set of indices to a central authority (tracker) in the system. This authority will combine all the sets it receives, and knowing the RSKR-repetition structure (see Fig. 6), it forms a list of suspected nodes that will surely include the malicious nodes that are sending corrupted data to the data collectors. Since there are at most b malicious nodes and each symbol x_i is stored on exactly two nodes, the size of the list will be at most $2b$. The system is then purged by discarding the nodes in this list.

VII. Active Limited-Knowledge Adversary

In this section, we consider the case of a non-omniscient active adversary with limited eavesdropping and controlling capabilities. We assume the adversary can eavesdrop on ℓ nodes and control some subset of $b \leq \ell$ nodes out of these ℓ nodes. The adversary's knowledge about the stored file is *limited* to what it can deduce from the observed nodes. Moreover, we assume that the adversary knows the coding and decoding strategies at every node in the system. Clearly when $\ell \geq k$, the adversary becomes omniscient. We are interested here in the limited-knowledge scenario that does not degenerate into the omniscient model studied in the previous section. For this case, we demonstrate that the resiliency capacity of the DSS exceeds that of the omniscient case, and can be achieved by storing a small *hash* on the nodes in addition to the data. Our approach is similar to that of [23], [24], [25], where the authors consider a limited-knowledge adversary that can eavesdrop and control *edges* rather than *nodes* in multicast networks.

Example 13: Consider a DSS $\mathcal{D}(5,3,4)$ with $\alpha = \gamma = 4$ with an adversary Charlie that can eavesdrop on and control one node, *i.e.*, $b = \ell = 1$. In the omniscient case with $b = 1$, the resiliency capacity of this system as given by Theorem 7 is equal to 2. Here, we show that the limitation on Charlie's knowledge can be leveraged to increase the resiliency capacity to 5.

First, we show that the resiliency capacity for this DSS is upper bounded by 5. To that end, consider the case when node v_1 is observed and controlled by Charlie. Moreover, assume that nodes v_2 and v_3 fail successively and are replaced by nodes v_6 and v_7 as shown in Fig. 9. Consider now a data collector DC that connects to nodes v_1, v_6, v_7 and wants to reconstruct the stored file. One possible attack that Charlie can perform, is to erase all the data stored on node v_1, *i.e.*, always change it to a fixed value irrespective of the stored file. This renders node v_1 useless and the system performs as if node v_1 was removed which reduces the value of the cut $C(V, \bar{V})$ (see Fig. 9) between the source s and data collector DC to 5.

We now exhibit a code that uses a simple "correlation" hash scheme to achieve the above upper bound with high probability.

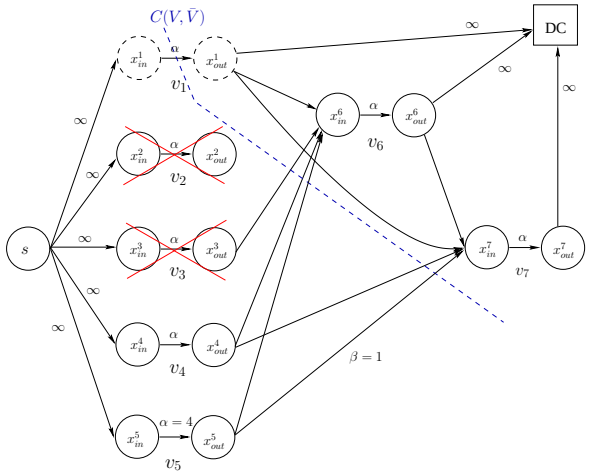

Fig. 9. The limited-knowledge adversary Charlie eavesdrops and controls node v_1, shown with the broken boundary. If Charlie erases the data stored on node v_1, the value of the cut $C(V, \bar{V})$, with $\bar{V} = \{x_{out}^1, x_{in}^6, x_{out}^6, x_{in}^7, x_{out}^7, \text{DC}\}$, between the source node s and a data collector DC accessing nodes v_6, v_7, v_8 becomes equal to 5.

a) Code Construction: The code consists of an outer $(10, 5)$ MDS code over $\mathbb{F}_{q^v}$, followed by the RSKR-repetition code enabling the exact repair of the nodes in the case of failures. Furthermore, each data packet $\mathbf{x_i} \in \mathbb{F}_{q^v}$ is appended with a hash vector $\mathbf{h_i} = (h_{i,1}, \ldots, h_{i,10}) \in \mathbb{F}_q^{10}$ computed as,

$$h_{i,j} \quad = \quad \mathbf{x_i} \mathbf{x_j}^T,$$

for $j = 1, 2, \ldots, 10$, where with abuse of notation, $\mathbf{x_i}$ also denotes the vector $(x_{i,1}, \ldots, x_{i,v})$ in $\mathbb{F}_q^v$ representing the corresponding element of $\mathbb{F}_{q^v}$. The schematic form of the code is shown in Table II below.

For simplicity, we assume in this example that the hash values stored on the nodes are made secure from Charlie who can neither observe, nor corrupt them. Later in Appendix C, we explain how this can be achieved in the general case with a negligible sacrifice in the system capacity. Note that even though Charlie cannot directly observe the hash table, he can generate some of the hash values using the observed data packets on $\ell = 1$ eavesdropped nodes, since he knows the coding scheme. Charlie can use these computed hash values to carefully introduce errors in the data symbols such that it is still consistent with these hash values.

Node	data $\in \mathbb{F}_{q^v}$	hash $\in \mathbb{F}_q^{10}$
v_1	$\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}, \mathbf{x_4}$	$\mathbf{h_1}, \mathbf{h_2}, \mathbf{h_3}, \mathbf{h_4}$
v_2	$\mathbf{x_1}, \mathbf{x_5}, \mathbf{x_6}, \mathbf{x_7}$	$\mathbf{h_1}, \mathbf{h_5}, \mathbf{h_6}, \mathbf{h_7}$
v_3	$\mathbf{x_2}, \mathbf{x_5}, \mathbf{x_8}, \mathbf{x_9}$	$\mathbf{h_2}, \mathbf{h_5}, \mathbf{h_8}, \mathbf{h_9}$
v_4	$\mathbf{x_3}, \mathbf{x_6}, \mathbf{x_8}, \mathbf{x_{10}}$	$\mathbf{h_3}, \mathbf{h_6}, \mathbf{h_8}, \mathbf{h_{10}}$
v_5	$\mathbf{x_4}, \mathbf{x_7}, \mathbf{x_9}, \mathbf{x_{10}}$	$\mathbf{h_4}, \mathbf{h_7}, \mathbf{h_9}, \mathbf{h_{10}}$

TABLE II
The schematic form of the code stored on the DSS $\mathcal{D}(5,3,4)$, along with the secure hash table that is not accessible to the adversary Charlie.

b) Decoding logic: A data collector contacting 3 nodes observes 12 symbols in total. In a worst case scenario, Charlie can corrupt 6 out of these 12 symbols. This can happen, for instance, when Charlie eavesdrops and controls node v_1,

and maliciously changes its stored data from $\mathbf{x_i}$ to $\mathbf{y_i} = \mathbf{x_i} + \mathbf{e_i}, \mathbf{e_i} \neq 0, i = 1, \ldots, 4$. Then, v_2, v_3 fail successively (as shown in Fig. 9) and Charlie sends the erroneous symbols $\mathbf{y_1}$ and $\mathbf{y_2}$, respectively, to nodes v_5 and v_6 during the repair process. In this scenario, a data collector, unaware of Charlie's actual node location, accessing nodes v_1, v_6 and v_7 will have among its observation 6 corrupted symbols, namely those having indices $1, \ldots, 4$ as shown in Table III, where the symbol $\mathbf{y_i}$ denotes the possibly corrupted version of $\mathbf{x_i}, i = 1, \ldots, 9$. Here, we have $\mathbf{y_i} = \mathbf{x_i}, i = 5, \ldots, 9$. The table also shows the hash vectors observed by the same data collector.

Node	data $\in \mathbb{F}_{q^v}$	hash $\in \mathbb{F}_q^{10}$
v_1	$\mathbf{y_1}, \mathbf{y_2}, \mathbf{y_3}, \mathbf{y_4}$	$\mathbf{h_1}, \mathbf{h_2}, \mathbf{h_3}, \mathbf{h_4}$
v_6	$\mathbf{y_1}, \mathbf{y_5}, \mathbf{y_6}, \mathbf{y_7}$	$\mathbf{h_1}, \mathbf{h_5}, \mathbf{h_6}, \mathbf{h_7}$
v_7	$\mathbf{y_2}, \mathbf{y_5}, \mathbf{y_8}, \mathbf{y_9}$	$\mathbf{h_2}, \mathbf{h_6}, \mathbf{h_8}, \mathbf{h_9}$

TABLE III
THE DATA SYMBOLS AND HASH VALUES OBSERVED BY THE DATA COLLECTOR CONTACTING NODES v_1, v_6, v_7, WHEN NODE v_1 IS CONTROLLED BY CHARLIE.

Among the 12 stored symbols $\mathbf{x_i}$ observed by the data collector and their hashes $\mathbf{h_i}$, each of the 3 symbols with indices $1, 2, 5$ and the corresponding hash vectors h_1, h_2, h_5 are repeated twice. Since the adversary can change both copies of each repeated data symbol identically, our decoder focuses only on a set of $M = 9$ symbols of distinct indices and the corresponding hash vectors for decoding. Note that the corresponding 9 symbols $(\mathbf{x_1}, \ldots, \mathbf{x_9})$ form a codeword of a $(9, 5)$ MDS code that we refer to as $\mathcal{X}$.

Let H denote the 9×9 hash matrix observed by the data collector, obtained as

$$H = \begin{bmatrix} \mathbf{h_1} \\ \mathbf{h_2} \\ \vdots \\ \mathbf{h_9} \end{bmatrix},$$

where the i^{th} row $\mathbf{h_i} \in \mathbb{F}_q^{10}$ corresponds to the hash vector of the symbol $\mathbf{y_i}, i = 1, \ldots, 9$. The data collector then computes its own 9×9 hash matrix $\hat{H}$ from the 9 observed symbols $\mathbf{y_i}$ as

$$\hat{H}_{ij} = \mathbf{y_i} \mathbf{y_j}^T, \qquad 1 \leq i, j \leq 9.$$

Then, it compares the entries in $\hat{H}$ with the corresponding entries in H to generate a 9×9 comparison table. Table IV is an example of such a comparison table where a "✓" in position (i, j) indicates that the computed hash and the observed hash match, *i.e.*, $\hat{H}_{ij} = H_{ij}$, whereas "×" indicates that $\hat{H}_{ij} \neq H_{ij}$ due to the errors introduced by the adversary.

The decoder selects a *trusted set* of 5 symbols from $\{\mathbf{y_1}, \ldots, \mathbf{y_9}\}$ that index a 5×5 sub-table of the comparison table where all the entries are "✓", *e.g.*, symbols $\mathbf{y_5}, \mathbf{y_6}, \mathbf{y_7}, \mathbf{y_8}, \mathbf{y_9}$ in Table IV. It then sets the remaining 4 symbols as erasures and proceeds to decode using a minimum distance decoder for the $(9, 5)$ MDS code $\mathcal{X}$, that can correct up to 4 erasures. There always exists at least one set of 5 symbols that generates a consistent hash table, *e.g.*, $T = \{\mathbf{y_5}, \mathbf{y_6}, \mathbf{y_7}, \mathbf{y_8}, \mathbf{y_9}\}$ when Charlie controls node v_1. Hence, the proposed decoding will eventually stop and output a decoding decision. Next, we analyze the probability of selecting a trusted set that results in an error in decoding.

Data Symbol	$\mathbf{y_1}$	$\mathbf{y_2}$	$\mathbf{y_3}$	$\mathbf{y_4}$	$\mathbf{y_5}$	$\mathbf{y_6}$	$\mathbf{y_7}$	$\mathbf{y_8}$	$\mathbf{y_9}$
$\mathbf{y_1}$	✓	✓	✓	✓	×	×	×	×	×
$\mathbf{y_2}$	✓	✓	✓	✓	×	×	×	×	×
$\mathbf{y_3}$	✓	✓	✓	✓	×	×	×	×	×
$\mathbf{y_4}$	✓	✓	✓	✓	×	×	×	×	×
$\mathbf{y_5}$	×	×	×	×	✓	✓	✓	✓	✓
$\mathbf{y_6}$	×	×	×	×	✓	✓	✓	✓	✓
$\mathbf{y_7}$	×	×	×	×	✓	✓	✓	✓	✓
$\mathbf{y_8}$	×	×	×	×	✓	✓	✓	✓	✓
$\mathbf{y_9}$	×	×	×	×	✓	✓	✓	✓	✓

TABLE IV
EXAMPLE OF THE COMPARISON TABLE OF THE HASH MATRICES H AND $\hat{H}$. NOTE THAT SINCE CHARLIE OBSERVES THE DATA SYMBOLS $\{\mathbf{x_1}, \ldots, \mathbf{x_4}\}$, HE CAN INTRODUCE ERRORS SUCH THAT THE HASH VALUES OF $\{\mathbf{y_1} \ldots, \mathbf{y_4}\}$ ARE CONSISTENT.

c) Error Analysis: Let $E = \{\mathbf{x_1}, \ldots, \mathbf{x_4}\}$ denote the set of data symbols observed by Charlie by eavesdropping on $\ell = 1$ node (v_1 in this case). The above proposed decoder may result in an error only if the chosen trusted set T contains at least one erroneous symbol, say $\mathbf{y_1}$. Therefore, we can write $\mathbf{y_1} = \mathbf{x_1} + \mathbf{e_1}$ for some error $\mathbf{e_1} \neq 0 \in \mathbb{F}_{q^v}$. Any chosen trusted set T is also guaranteed to contain at least one error-free symbol that is not observed by Charlie, say $\mathbf{y_5} = \mathbf{x_5} \notin E$. To see this, note that the cardinality of the trusted set T is 5, and by eavesdropping and controlling any one node Charlie can observe and introduce errors in a maximum of 4 symbols with distinct indices to any data collector observation. For the set T, containing $\mathbf{y_1}, \mathbf{y_5}$ along with 3 other symbols, to be a trusted set, it has to generate a consistent hash table of size 5×5. Therefore, Charlie has to pick the error $\mathbf{e_1}$ to satisfy $\mathbf{x_5} \mathbf{e_1}^T = 0$.

The observation $E = \{\mathbf{x_1}, \ldots, \mathbf{x_4}\}$ of Charlie is independent of $\mathbf{x_5}$ due to the MDS property of the outer code. Therefore, for any choice of $\mathbf{e_1}$ that Charlie makes, there are q^v equally likely choices of $\mathbf{x_5}$, out of which q^{v-1} are orthogonal to the chosen $\mathbf{e_1}$. Hence, the consistency condition of hash $\hat{H}_{5,1} = H_{5,1}$ is satisfied with probability,

$$P_r(\mathbf{x_5} \mathbf{e_1}^T = 0 | E, \mathbf{e_1}) = \frac{1}{q}.$$

Note that if Charlie could observe the complete hash table, then $\mathbf{x_5}$ is no more independent of Charlie's observation. For example, if Charlie observes the hash value $H_{2,5} = \mathbf{x_2} \mathbf{x_5}^T$, then for a given value of $\mathbf{x_2}$ and $H_{2,5}$, there are only q^{v-1} equally likely choices for $\mathbf{x_5}$. In which case Charlie can always choose $\mathbf{e_1}$ to belong to the space orthogonal to $v - 1$ dimensional space of possible choices of $\mathbf{x_5}$, thus, deceiving the proposed decoder. Therefore, it is crucial to keep the hash values secure from Charlie.

It can be verified that the above reasoning easily carries to any choice of $b = 1$ node controlled by Charlie. Therefore, the probability of error is upper bounded by $1/q$ which vanishes with increasing the field size q.

d) Rate Analysis: We encode 5 information symbols in $\mathbb{F}_{q^v}$ to form the coded symbols $\mathbf{x_i}, i = 1, \ldots, 10$. For these 10 symbols we construct a hash table of size 10×10 with elements in $\mathbb{F}_q$. Hence the total overhead of the hash table

is $\frac{100}{5v} = \mathcal{O}(\frac{1}{v})$ per information symbol. Thus, the rate of our code is $5 - \mathcal{O}(\frac{1}{v})$ which approaches 5 with an increasing block length v.

A. Results on Active Limited-Knowledge Adversary

Below we summarize our two main results on the resiliency capacity in the case of a limited-knowledge adversary.

Theorem 14: For a DSS $\mathcal{D}(n,k,d)$ with an adversary that can eavesdrop on any $\ell < k$ nodes and control a subset of size b of these ℓ nodes ($b \leq \ell$), the following upper bound holds on the resiliency capacity,

$$C_r(\alpha,\gamma) \leq \sum_{i=b+1}^{k} \min\{(d-i+1)\beta, \alpha\} \tag{16}$$

where $d\beta = \gamma$.

Proof: (sketch) Consider a case when nodes $v_1,\ldots,v_k$ fail successively and are replaced by nodes $v_{n+1},\ldots,v_{n+k}$ as shown in Fig. 10. Also consider a data collector DC that contacts these k nodes $\{v_{n+1},\ldots,v_{n+k}\}$ to retrieve the source file. If the adversary Charlie controls the b nodes $\{v_{n+1},\ldots,v_{n+b}\}$, one possible adversarial strategy that Charlie can use is to erase all the data stored on these b nodes, *i.e.*, always change it to a fixed value irrespective of the file stored on the DSS. This renders the b controlled nodes useless, resulting in the upper bound stated in the theorem. ∎

Let $R := \sum_{i=b+1}^{k} \min\{(d-i+1)\beta, \alpha\}$ and $\mathcal{E} := \sum_{i=1}^{\ell} \min\{(d-i+1)\beta, \alpha\}$. Our second results states that if the eavesdropping capability ℓ of the adversary Charlie is limited, in particular ℓ is such that $\mathcal{E} < R$, the upper bound in Theorem 14 can be achieved for $d = n-1$ in the bandwidth-limited regime.

Theorem 15: Consider a DSS $\mathcal{D}(n,k,d=n-1)$ operating in the bandwidth-limited regime in the presence of an adversary that can eavesdrop on ℓ nodes and controls a subset of size b of these ℓ nodes ($b \leq \ell$). Then, if the adversary is limited-knowledge, *i.e.*, ℓ is such that $\mathcal{E} < R$, the resiliency capacity of the system is,

$$C_r^{BL}(\Gamma) = \sum_{i=b+1}^{k} (n-i)\beta, \tag{17}$$

where $\beta = \Gamma/(n-1)$.

The condition $\mathcal{E} < R$ in Theorem 15 says that the eavesdropping capability of the adversary is insufficient to determine the message stored on the DSS, *i.e.,* the adversary is not omniscient. This limitation in the adversary's knowledge enables every data collector to identify the erroneous symbols introduced by the adversary and discard them, thus, resulting in erasures rather than errors. In this case also, identifying the erroneous symbols helps in the expurgation of the system and discarding the malicious nodes, as pointed out in Remark 12.

The proof of Theorem 15 is detailed in Appendix C and is composed of two parts. In the first part, we assume that the hash table is secure from the adversary and generalize the reasoning of Example 13 to show how the hash table can be used to identify, with high probability, the erroneous symbols introduced by Charlie and thus decode correctly. In the second part, we demonstrate an efficient scheme to store the hash table securely and reliably with a negligible sacrifice in the system capacity.

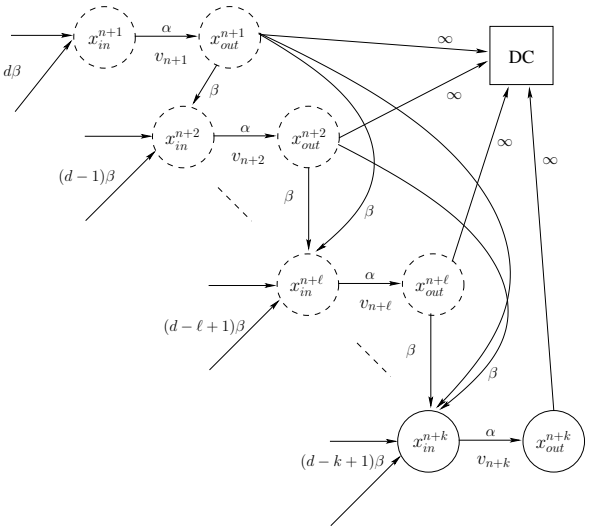

Fig. 10. Part of the information flow graph corresponding to a DSS $\mathcal{D}(n,k,d)$, when nodes $v_1,\ldots,v_k$ fail successively and are replaced by nodes $v_{n+1},\ldots,v_{n+k}$. A data collector contacts these k nodes and wants to reconstruct the stored file. Nodes $v_{n+1},\ldots,v_{n+\ell}$ shown with broken boundaries are compromised by Eve while they were being repaired.

VIII. Conclusion

In this paper we have considered the problem of securing a distributed storage system under *repair dynamics* against eavesdropping and adversarial attacks. We proposed a new dynamical model for the intrusion, wherein the adversary intrudes the system at different time instances in order to exploit the system repair dynamics to its own benefit. For the general model of an adversary that can eavesdrop and/or maliciously change the data on some nodes in the system, we investigate the problem of determining the *secrecy capacity* and *resiliency capacity* of the system. We provide upper bounds on the *secrecy and resiliency* capacity and show their achievability in the *bandwidth-limited regime*. General expressions of these capacities in addition to efficient decoding algorithms remain an open problem.

Appendix

A. Proof of Theorem 1

Consider a DSS $\mathcal{D}(n,k,d)$ with $\ell < k$, operating at point (α,γ) with $d\beta = \gamma$. Assume that nodes $v_1, v_2, \ldots, v_k$ have failed successively and were replaced during the repair process by the nodes $v_{n+1}, v_{n+2}, \ldots, v_{n+k}$ respectively as shown in the corresponding information flow graph $\mathcal{G}$ in Fig. 10. Now suppose that Eve accesses the ℓ input nodes in the set $E = \{x_{in}^{n+1}, x_{in}^{n+2}, \ldots, x_{in}^{n+\ell}\} \subset V_{in}$ while they were being repaired. Consider also a data collector DC that downloads data from the k output nodes in $B = \{x_{out}^{n+1}, x_{out}^{n+2}, \ldots, x_{out}^{n+k}\} \in V_{out}^{a}$. The reconstruction property of Eq. (4) implies $H(S|C_B) = 0$ and the perfect secrecy

condition in Eq. (5) implies $H(S|D_E) = H(S)$. We can therefore write

$$\begin{aligned} H(S) &= H(S|D_E) - H(S|C_B) \\ &\overset{(1)}{\leq} H(S|C_E) - H(S|C_B) \\ &\overset{(2)}{=} H(S|C_E) - H(S|C_E, C_{B\setminus E}) \\ &= I(S, C_{B\setminus E}|C_E) \\ &\leq H(C_{B\setminus E}|C_E) \\ &= \sum_{i=\ell+1}^{k} H(C_{n+i}|C_{n+1}, \ldots, C_{n+i-1}) \\ &\overset{(3)}{\leq} \sum_{i=\ell+1}^{k} \min\{(d-i+1)\beta, \alpha\} \end{aligned}$$

Inequality (1) follows from the Markov chain $S \rightarrow D_E \rightarrow C_E$ i.e., the stored data C_E is dependent on S only through the downloaded data D_E, (2) from $C_{B\setminus E} := \{C_{n+\ell+1}, \ldots, C_{n+k}\}$, (3) follows from the fact that each node can store at most α units, and for each replacement node we have $H(C_i) \leq H(D_i) \leq d\beta$, also from the topology of the network (see Fig. 10) where each node x_{in}^{n+i} is connected to each of the nodes $x_{out}^{n+1}, \ldots, x_{out}^{n+i-1}$ by an edge of capacity β. The upper bound of Theorem 1 then follows directly from the definition of Eq. (6).

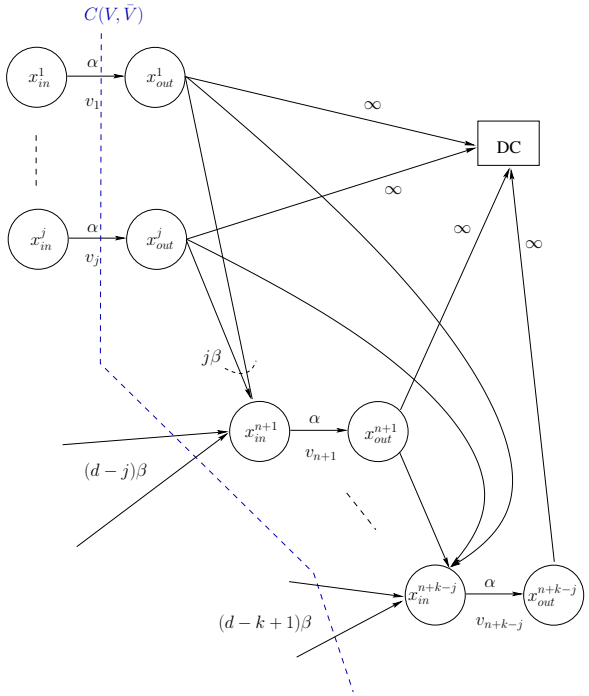

Fig. 11. Part of the information flow graph corresponding to a DSS (n, k, d) when nodes $v_{j+1}, \ldots, v_k$ fail successively and are replaced by nodes $v_{n+1}, \ldots, v_{n+k-j}$. A data collector connects to nodes $v_1, \ldots, v_j, v_{n+1}, \ldots, v_{n+k-j}$ to retrieve the file.

B. Proof of Theorem 6

Consider a DSS $\mathcal{D}(n, k, d)$ operating at point (α, γ) with $d\beta = \gamma$, in the presence of an omniscient adversary that can control b nodes, with $2b < k$. Assume that nodes $v_{j+1}, v_{j+2}, \ldots, v_k$, for some j, $2b < j < k$, have failed consecutively and were replaced by nodes $v_{n+1}, v_{n+2}, \ldots, v_{n+(k-j)}$, respectively. The information flow graph $\mathcal{G}$ of the DSS corresponding to this sequence of node failures and repairs is shown in Fig. 11. Consider a data collector (Fig. 11) that observes the stored data on the k nodes $v_1, \ldots, v_j, v_{n+1}, \ldots, v_{n+k-j}$. Consider also the cut $C(V, \bar{V})$ with $\bar{V} = \{x_{out}^1, \ldots, x_{out}^j, x_{in}^{n+1}, \ldots, x_{in}^{n+k-j}, x_{out}^{n+1}, \ldots, x_{out}^{n+k-j}, \text{DC}\}$ that separates the source node s from the data collector DC. We group the edges belonging to this cut into 3 disjoint sets as follows:

1) E_1: the set of edges outgoing from nodes $x_{in}^p, p = 1, \ldots, b$.
2) E_2: the set of edges outgoing from nodes $x_{in}^p, p = b+1, \ldots, 2b$.
3) E_3: the set of edges outgoing from nodes $x_{in}^p, p = 2b+1, \ldots, j$, in addition to the edges belonging to the cut $C(V, \bar{V})$ that are incoming to the nodes $x_{in}^q, q = n+1, \ldots, n+k-j$.

Let $X_{E_i}(m), i = 1, 2, 3$, be the symbols transmitted on the edges in set E_i corresponding to the stored message m. We claim that in the presence of an adversary controlling any b nodes and for any two distinct messages $m_1 \neq m_2$ the following condition is necessary for the DC to not make a decoding error:

$$X_{E_3}(m_1) \neq X_{E_3}(m_2).$$

Suppose that there exist two distinct messages $m_1 \neq m_2$ satisfying $X_{E_3}(m_1) = X_{E_3}(m_2)$. Now, if the symbols carried on the edges belonging to the cut $C(V, \bar{V})$ are $X_{E_1}(m_1), X_{E_2}(m_2)$ and $X_{E_3}(m_1) = X_{E_3}(m_2)$. Then, assuming all the messages to be equally likely, the data collector will make a decoding error with probability at least $1/2$. This is true since it will not be able to distinguish between the following two cases:

- The true message is m_2 and the nodes $x_{in}^1, \ldots, x_{in}^b$ are controlled by the adversary Calvin who changed the transmitted symbols on the edges in the set E_1, from $X_{E_1}(m_2)$ to $X_{E_1}(m_1)$.
- The true message is m_1 and the nodes $x_{in}^{b+1}, \ldots, x_{in}^{2b}$ are controlled by the adversary Calvin who changed the transmitted symbols on the edges in the set E_2, from $X_{E_2}(m_1)$ to $X_{E_2}(m_2)$.

Thus, the capacity of the DSS is upper bounded by the total capacity of the edges in the set E_3, *i.e.*,

$$C_r(\alpha, \gamma) \leq \sum_{i=2b+1}^{j} \alpha + \sum_{i=j+1}^{k} (d-i+1)\beta, \qquad j = 2b+1, \ldots, k-1.$$

The same analysis, as above, can be applied for $j = 2b$ resulting in,

$$C_r(\alpha, \gamma) \leq \sum_{i=2b+1}^{k} (d-i+1)\beta.$$

And also for $j = k$, which gives,

$$C_r(\alpha, \gamma) \leq \sum_{i=2b+1}^{k} \alpha.$$

The bound in Theorem 6 then follows by taking the minimum of all the above upper bounds obtained for $j = 2b, \ldots, k$. It can be easily seen that the above argument extends to the case of $2b \geq k$ for which the set E_3 is empty and $C_r(\alpha, \gamma) = 0$. □

C. Proof of Theorem 15

Consider a DSS $\mathcal{D}(n, k, d)$, with $d = n-1$, operating in the bandwidth-limited regime, in the presence of an adversary that can eavesdrop on ℓ nodes and control a subset of them of size b, $b \leq \ell$. As in the earlier proofs, we show the achievability for $\beta = 1$, *i.e.*, $\Gamma = n - 1$. Any larger values of β or Γ can be achieved by repeatedly applying the proposed scheme. Since there is no constraint on the node storage capacity α in bandwidth-limited regime, we choose $\alpha = n - 1$. Let $\theta := \frac{n(n-1)}{2}$, $M := \sum_{i=1}^{k}(n-i)$, $R := \sum_{i=b+1}^{k}(n-i)$ and $\mathcal{E} := \sum_{i=1}^{\ell}(n-i)$.

Our proof consists of two parts: 1) We assume that the hash table can be stored securely and reliably, and show an achievable scheme that can attain the resiliency capacity. 2) We present an efficient method to reliably and securely store the hash table in the presence of a limited-knowledge adversary Charlie.

C.1 Resiliency Capacity in the Limited-knowledge Case for the Bandwidth-Limited Regime

Code Construction: The code that we propose here is a generalization of the one used in Example 13 of Section VII. It consists of an outer (θ, R) MDS code whose output $X = (\mathbf{x_1}, \ldots, \mathbf{x}_\theta) \in \mathbb{F}_{q^v}^{\theta}$ is stored on the n storage nodes using an inner RSKR-repetition code that enables exact repair in case of any node failure. As shown in Table V, each data packet $\mathbf{x_i} \in \mathbb{F}_{q^v}, i = 1, \ldots, \theta$, is further appended with a hash vector $\mathbf{h_i} = (h_{i,1}, \ldots, h_{i,\theta}) \in \mathbb{F}_q^{\theta}$. The values of these hashes are computed as follows,

$$h_{i,j} \quad = \quad \mathbf{x_i}\mathbf{x_j}^T,$$

for $j = 1, 2, \ldots, \theta$, where with abuse of notation $\mathbf{x_i}$ also denotes the vector in $\mathbb{F}_q^v$ representing the corresponding element of $\mathbb{F}_{q^v}$. We assume for now that the hash values stored on the nodes are secure from Charlie who can neither observe nor corrupt them (as shown in the next section). Although Charlie cannot directly observe the hash table, he can compute some of the hash values using the observed data packets on ℓ eavesdropped nodes and possibly introduce errors that are consistent with these hash values.

Decoding Logic: A data collector accessing any k nodes will observe a total of $(n-1)k$ symbols and the corresponding hash vectors, where $\binom{k}{2}$ indices are repeated twice. As noted earlier, since the adversary can corrupt both of the stored symbols with same indices identically, the decoder focuses only on a set of $M = \sum_{i=1}^{k}(n-i)$ symbols with distinct indices along with their hash vectors to make a decoding decision. These M symbols form a codeword of an (M, R) MDS code $\mathcal{X}$ possibly corrupted by errors introduced by the adversary.

Node	data packet $\in \mathbb{F}_{q^v}$				hash $\in \mathbb{F}_q^{\theta}$			
v_1	$\mathbf{x_1}$	$\mathbf{x_2}$	$\ldots$	$\mathbf{x_{n-1}}$	$\mathbf{h_1}$	$\mathbf{h_2}$	$\ldots$	$\mathbf{h_{n-1}}$
v_2	$\mathbf{x_1}$	$\mathbf{x_n}$	$\ldots$	$\mathbf{x_{2n-3}}$	$\mathbf{h_1}$	$\mathbf{h_n}$	$\ldots$	$\mathbf{h_{2n-3}}$
v_3	$\mathbf{x_2}$	$\mathbf{x_n}$	$\ldots$	$\mathbf{x_{3n-6}}$	$\mathbf{h_2}$	$\mathbf{h_n}$	$\ldots$	$\mathbf{h_{3n-6}}$
$\vdots$	$\ldots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
v_n	$\mathbf{x_{n-1}}$	$\mathbf{x_{2n-3}}$	$\ldots$	$\mathbf{x}_\theta$	$\mathbf{h_{n-1}}$	$\mathbf{h_{2n-3}}$	$\ldots$	$\mathbf{h}_\theta$

TABLE V
SCHEMATIC FORM OF THE CODE STORED ON THE DSS $(n, k, d = n-1)$, ALONG WITH THE HASH TABLE THAT IS NOT ACCESSIBLE TO THE ADVERSARY CHARLIE.

Recall that Charlie can eavesdrop on a total of ℓ nodes and control some subset $b \leq \ell$ of these eavesdropped nodes in the system. Let $\mathbf{y_i}, i = 1, \ldots, \theta$, denote the possibly corrupted version of the original data symbols $\mathbf{x_i}$. We have $\mathbf{y_i} = \mathbf{x_i} + \mathbf{e_i}$, where $\mathbf{e_i}$ is the error introduced by Charlie on the symbols stored on the nodes he controls, and for rest of symbols $\mathbf{e_i} = 0$. Without loss of generality, we suppose that the data collector observes nodes $v_1, \ldots, v_k$, *i.e.*, data symbols $\mathbf{y_i}$ and hash values $\mathbf{h_i}, i \in \{1, 2, \ldots, M\}$. The data collector observes the hash values with no errors since the hash table is assumed to be secure and reliable against the adversary. Let H denote the observed $M \times \theta$ hash matrix having the vectors $\mathbf{h_i} \in \mathbb{F}_q^{\theta}, i = 1, \ldots, M$ as rows. The data collector then computes its own $M \times M$ hash matrix $\hat{H}$ as

$$\hat{H}_{ij} = \mathbf{y_i}\mathbf{y_j}^T, \qquad 1 \leq i, j \leq M$$

from the observed M data packets and compares it with the corresponding entries in H. It generates an $M \times M$ comparison table similar to Table IV in Example 13. In this table a "✓" in the i-th row and j-th column indicates that the computed hash and the observed hash match, *i.e.*, $\hat{H}_{ij} = H_{ij}$, whereas "×" indicates that $\hat{H}_{ij} \neq H_{ij}$ due to the errors introduced by the adversary.

The decoder then selects a set of R symbols, among $(\mathbf{y_1}, \ldots, \mathbf{y_M})$, that index an $R \times R$ sub-table of the comparison table with all its entries equal to "✓", and declares it as a *trusted set* with no errors. Then, it sets the rest of the $M - R$ observed symbols as erased and proceeds to decode the obtained vector as a codeword of an (M, R) MDS code $\mathcal{X}$ with $M - R$ erasures. Since Charlie can control only b nodes there always exists at least one set of size $M - \sum_{i=1}^{b}(n-i) = R$ symbols that generates a consistent hash sub-table of size $R \times R$ with "✓". Hence, the proposed decoder is guaranteed to stop. Next, we compute the probability that the above decoder decodes to an incorrect message.

Error Analysis: The proposed decoder may result in an error in decoding only if the chosen trusted set of R observed symbols contains at least one erroneous symbol, say $\mathbf{y_j} = \mathbf{x_j} + \mathbf{e_j}$,

$\mathbf{e_j} \neq 0$. Also, since $b \leq \ell$, we have,

$$\sum_{i=1}^{b}(n-i) \leq \sum_{i=1}^{\ell}(n-i) < R, \tag{18}$$

where the last inequality follows from our assumption (see Theorem 15) that the eavesdropping capability $\mathcal{E}$ is strictly less than the desired storage rate R. From equation (18), it is clear that the chosen trusted set contains at least one error-free symbol that is not observed by Charlie, say $\mathbf{y_i} = \mathbf{x_i} \notin E$. For this set to be a trusted set, it has to generate a consistent hash table of size $R \times R$. In particular $\hat{H}_{ij} = H_{ij}$, *i.e.*, $\mathbf{x_i}\mathbf{e_j}^T = 0$.

Next, we compute the probability of such event. Let E be the set of symbols in the codeword X that are observed by Charlie. Since X is the output of a (θ, R) MDS code and $|E| < R$, any symbol $\mathbf{x_i}$ of X that does not belong to E is uniformly distributed in $\mathbb{F}_{q^v}$ conditioned on E, *i.e.*,

$$Pr(\mathbf{x_i} = x_i|E) = \frac{1}{q^v}, \qquad x_i \in \mathbb{F}_{q^v}. \tag{19}$$

Therefore, for any choice of $\mathbf{e_j}$ that Charlie makes based on his observation E, there are q^v equally likely choices of $\mathbf{x_i}$ out of which q^{v-1} are orthogonal to the chosen $\mathbf{e_j}$. Hence, the consistency condition of hash $\hat{H}_{i,j} = H_{i,j}$ is satisfied with probability,

$$P_r(\mathbf{x_i}\mathbf{e_j}^T = 0|E, \mathbf{e_j}) = \frac{1}{q},$$

which goes to zero with increasing field size q.

Note that if Charlie could observe the complete hash table, $\mathbf{x_i}$ would no more be independent of Charlie's observation. Then, as shown in Example 13, Charlie can always choose $\mathbf{e_j}$ to belong to the orthogonal space of all possible choices of $\mathbf{x_i}$, thus deceiving the proposed decoder. Therefore, it is crucial to keep the hash values secure from Charlie.

Rate Analysis: We encode R information symbols in $\mathbb{F}_{q^v}$ using a (θ, R) MDS code to form a codeword $(\mathbf{x_1}, \ldots, \mathbf{x}_\theta)$. For these symbols we construct a hash table of size $\theta \times \theta$ with symbols in $\mathbb{F}_q$. Hence the total overhead of the hash table is $\frac{\theta^2}{Rv} = \mathcal{O}(\frac{1}{v})$ per information symbol which goes to zero with increasing block length v. Hence, asymptotically in block length v, these codes achieve the capacity of Theorem 15.

C.2 Reliable and Secure Storage of the Hash Table

The scheme described here for storing the hash table securely and reliably is along the parallel lines of the scheme proposed [25][5] in the context of securing multicast networks. It aims at storing 1 bit of information securely and reliably. The scheme can then be repeated to store the complete hash table which, as shown in the previous section, is of constant size and independent of the block length v of the information symbols. The total overhead incurred by this scheme can be then made arbitrarily small by increasing v.

[5]The scheme of [25] is matrix-based and is designed for networks where intermediate nodes perform random network coding. Our scheme here can be regarded as a simple vector version of the one in [25]. This simplification is possible due to the special structure of the networks (information flow graphs) representing distributed storage systems in conjunction with the RSKR-repetition codes that limit coding in these networks to the source.

Code Construction: Let $G = \begin{pmatrix} G_K \\ G_S \end{pmatrix}$ be a generator matrix of a (θ, M) nested MDS code over the finite field $\mathbb{F}_q$ (symbols in the hash table also belong to the same field). The matrix G_K in itself is a generator matrix of a $(\theta, \mathcal{E})$ MDS code over $\mathbb{F}_q$. If the bit to be stored is "1" then choose a vector S randomly and uniformly from $\mathbb{F}_q^{M-\mathcal{E}}$, otherwise, set $S = 0 \in \mathbb{F}_q^{M-\mathcal{E}}$. Let $\mathcal{K} = (K_1 \ldots, K_\mathcal{E})$ denote $\mathcal{E}$ random keys mutually independent and each uniformly distributed over $\mathbb{F}_q$. Now, we form the vector $X \in \mathbb{F}_q^\theta$ to be stored on the DSS as part of the hash table by "mixing" S with the random keys using the nested MDS code as,

$$X \quad = \quad \mathcal{K}G_K + SG_S.$$

This encoded vector $X \in \mathbb{F}_q^\theta$ is then stored on the (n, k, d) DSS using the RSKR-repetition code as shown in Fig. 6. The RSKR-repetition structure allows the exact repair of a node in case of failure as explained in Section V.

Security Analysis: The coding scheme used here is same as the one in Section V-D that discusses passive adversary and hence the vector S, which is of the appropriate rate $M - \mathcal{E}$, is perfectly secure from Charlie eavesdropping on ℓ nodes. The perfect secrecy of S implies the perfect secrecy of the hash bit.

Next we describe a decoding algorithm that the data collector uses to decode the stored bit with high probability of success even in the presence of errors introduced by Charlie controlling b nodes.

Decoding Logic: We denote by $\mathbb{D}$ the decoder used by the data collector to recover the stored bit belonging to the hash table. $\mathbb{D}$ implements the same decoding steps as the decoder of Section VI-B, of omniscient adversary, except for the decision rule that determines the output. The input to $\mathbb{D}$ is the data observed by the data collector accessing k nodes which is formed of $k\alpha = k(n-1)$ symbols, among which $\binom{k}{2}$ pairs have the same indices. The decoder executes the following steps:

1) $\mathbb{D}$ selects any set of M symbols having distinct indices among the observed $k\alpha$ symbols. These symbols are grouped in a vector $Y \in \mathbb{F}_q^M$ which can be written as
$$Y \quad = \quad \mathcal{K}\bar{G}_K + S\bar{G}_S + \mathbf{e},$$
where $\bar{G}_K$ and $\bar{G}_S$ are submatrices of G_K and G_S of size $\mathcal{E} \times M$ and $(M - \mathcal{E}) \times M$, respectively. The vector $\mathbf{e} \in \mathbb{F}_q^M$, with up to $\sum_{i=1}^{b}(n-i)$ non-zero terms, is the error vector that accounts for the errors introduced by the adversary.
2) Let $B, |B| = b$, denote the set of storage nodes controlled by the adversary. Again, due to the exact repair property of the RSKR-repetition code it is sufficient to consider $B \subset \{v_1, \ldots, v_n\}$ with $|B| = b$. For each such set B, let $I_B \subset \{1, 2, \ldots, \theta\}$ denote the set of indices of the symbols stored on the nodes in B.
3) For each possible $B \subset \{v_1, v_2, \ldots, v_n\}$, $|B| = b$, $\mathbb{D}$ punctures Y with pattern I_B to obtain Y_{I_B} as
$$Y_{I_B} \quad = \quad \mathcal{K}\bar{G}_{K_{I_B}} + S\bar{G}_{S_{I_B}} + \mathbf{e}_{I_B},$$

where $\bar{G}_{K_{I_B}}$ and $\bar{G}_{S_{I_B}}$ are the submatrices of $\bar{G}_K$ and $\bar{G}_S$ obtained by deleting the columns corresponding to the punctured elements of Y, and $\mathbf{e}_{I_B}$ is the punctured error vector.

4) $\mathbb{D}$ checks whether Y_{I_B} is a valid codeword of the code generated by the matrix $\bar{G}_{K_{I_B}}$ by checking whether the corresponding syndrome is zero.
5) The decoder $\mathbb{D}$ repeats steps 3) and 4) for each of the $\binom{n}{b}$ sets B until the syndrome obtained in step 4) is zero. In this case, $\mathbb{D}$ declares that bit "0" was stored. Otherwise, if for all possible values of B no zero syndrome is obtained, $\mathbb{D}$ declares that "1" was stored.

Error Analysis: We do the error analysis of the above decoding logic considering two different cases based on the value of the stored hash bit.

- *Hash bit '0':* We will show that when the stored information bit is '0', the decoder $\mathbb{D}$ makes no error. In fact, this case corresponds to $S = 0$ and, thus, $Y = \mathcal{K}\bar{G}_K + \mathbf{e}$. Let B^* be the actual set of nodes controlled by Charlie. Then, there is at least one set $B = B^*$ for which $Y_{I_B^*} = \mathcal{K}\bar{G}_{K_{I_B^*}}$, since $\mathbf{e}_{I_{B^*}} = 0$. As a result, the decoder always outputs "0".
- *Hash bit '1':* Information bit '1' corresponds to

$$Y \quad = \quad \mathcal{K}\bar{G}_K + S\bar{G}_S + \mathbf{e},$$

where $(\mathcal{K}, S)$ is a uniformly random vector in $\mathbb{F}_q^M$ and $\mathbf{e} \in \mathbb{F}_q^M$ is the error vector introduced by Charlie. Note that the matrix G is a generator matrix of a (θ, M) MDS code, hence the $M \times M$ sub-matrix $\bar{G} := \begin{pmatrix} \bar{G}_K \\ \bar{G}_S \end{pmatrix}$ is invertible. Thus, we can write

$$Y \quad = \quad (\mathcal{K} + \mathbf{e}_K)\bar{G}_K + (S + \mathbf{e}_S)\bar{G}_S, \qquad (20)$$

where $\mathbf{e}_K, \mathbf{e}_S$ are the coefficients of the error vector $\mathbf{e}$ in terms of the basis corresponding to the rows of $\bar{G}_K, \bar{G}_S$. We have already shown in the security analysis above, that S is perfectly secure from Charlie's observation. Hence $S + \mathbf{e}_S$ is a uniformly random vector in $\mathbb{F}_q^{M-\mathcal{E}}$. Consider any set $B \subset \{v_1, \ldots, v_n\}$ of cardinality $|B| = b$ with index set I_B. Then, $|I_B| = \sum_{i=1}^{b}(n-i)$, hence the matrix $\bar{G}_{I_B}$ obtained by deleting the columns of $\bar{G}$ corresponding to the indices I_B has $R = M - |I_B|$ or more columns. Now, the matrix $\bar{G}_{K_{I_B}}$ is a generator of an $(M, \mathcal{E})$ MDS code and $\mathcal{E} < R$ (Theorem 15). Hence, the rank of $\bar{G}_{K_{I_B}}$ is $\mathcal{E}$. This, along with the fact that $\bar{G}$ is an invertible matrix, implies that the rank of matrix $\bar{G}_{S_{I_B}}$ is $R - \mathcal{E}$ or more. The probability, that the syndrome computed in the step 4) of the proposed decoding logic for this set B is equal to zero, is equal to the probability of the event that a uniformly random vector $(S + \mathbf{e}_S)$ lies in the space orthogonal to the span of columns of $\bar{G}_{S_{I_B}}$. This probability is upper bounded by $1/q^{R-\mathcal{E}}$.
Now applying the union bound to all $\binom{n}{b}$ choices of the set B that the decoder attempts, the probability of error can be upper bounded by,

$$\lim_{q\to\infty} \frac{\binom{n}{b}}{q^{R-\mathcal{E}}} \to 0$$

which goes to zero with increasing the field size q.

Rate Analysis: In the code proposed above to store the hash values securely and reliably we need θ symbols in $\mathbb{F}_q$ for each 1 bit of hash information. Also, in the previous section we showed that the total size of the hash table of interest is θ^2 symbols in $\mathbb{F}_q$. Thus, the total overhead of the proposed code to store the hash table is $\theta^3 \log q$ symbols of $\mathbb{F}_q$, that is independent of the block length v of information packets.

Thus, we have shown how the hash table described in Table V can be stored on the DSS with a negligible overhead and is guaranteed with a high probability to be secret and resilient to the adversary provided that field size q and block length v are large enough.

REFERENCES

[1] S. Rhea, C. Wells, P. Eaton, D. Geels, B. Zhao, H. Weatherspoon, and J. Kubiatowicz, "Maintenance-free global data storage," *IEEE Internet Computing*, pp. 40–49, 2001.

[2] R. Bhagwan, K. Tati, Y.-C. Cheng, S. Savage, and G. M. Voelker, "Total recall: System support for automated availability management," in *Proc. NSDI*, 2004.

[3] F. Dabek, J. Li, E. Sit, J. Robertson, M. Kaashoek, and R. Morris, "Designing a DHT for low latency and high throughput," in *Proc. NSDI*, 2004.

[4] A. Dimakis, P. Godfrey, Y. Wu, M. Wainright, and K. Ramchandran, "Network coding for distributed storage systems," *IEEE Transactions on Information Theory*, vol. 56, pp. 4539–4551, Sep. 2010.

[5] T. Cui, T. Ho, and J. Kliewer, "On secure network coding over networks with unequal link capacities and restricted wiretapping sets," in *IEEE Internat. Symp. Inform. Th. (ISIT)*, 2010.

[6] O. Kosut, L. Tong, and D. Tse, "Nonlinear network coding is necessary to combat general byzantine attacks," in *Proc. of 47th Annual Allerton Conf. on Comm., Control, and Computing*, Oct. 2009.

[7] A. G. Dimakis, P. B. Godfrey, M. J. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," in *IEEE Internat. Conf. on Comp. Comm. (INFOCOM)*, 2007.

[8] Y. Wu, A. G. Dimakis, and K. Ramchandran, "Deterministic regenerating codes for distributed storage," in *Proc. of 45th Annual Allerton Conf. on Comm., Control, and Computing*, 2007.

[9] K. Rashmi, N. B. Shah, P. V. Kumar, and K. Ramchandran, "Exact regenerating codes for distributed storage," in *Proc. of 47th Annual Allerton Conf. on Comm., Control, and Computing*, 2009.

[10] Y. Wu and A. G. Dimakis, "Reducing repair traffic for erasure coding-based storage via interference alignment," in *IEEE Internat. Symp. Inform. Th. (ISIT)*, 2009.

[11] C. Suh and K. Ramchandran, "Exact regeneration codes for distributed storage repair using interference alignment," in *IEEE Internat. Symp. Inform. Th. (ISIT)*, 2010.

[12] N. B. Shah, K. V. Rashmi, P. V. Kumar, and K. Ramchandran, "Explicit codes minimizing repair bandwidth for distributed storage," in *Proceedings of IEEE Information Theory Workshop (ITW'10)*, 2010.

[13] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network Information Flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, 2000.

[14] C. Fragouli and E. Soljanin, *Network Coding Fundamentals (Foundations and Trends in Networking)*. Now Publishers Inc, 2007.

[15] R. Yeung, S.-Y. Li, and N. Cai, *Network Coding Theory (Foundations and Trends in Communications and Information Theory)*. Now Publishers Inc, 2006.

[16] T. K. Dikaliotis, A. G. Dimakis, and T. Ho, "Security in distributed storage systems by communicating a logarithmic number of bits," in *IEEE Internat. Symp. Inform. Th. (ISIT)*, 2010.

[17] N. Cai and R. W. Yeung, "Secure network coding," in *IEEE Internat. Symp. Inform. Th. (ISIT)*, 2002.

[18] N. Cai and R. W. Yeung, "Secure Network Coding on a Wiretap Network," *IEEE Transactions on Information Theory*, vol. 57, pp. 424–435, 2011.

[19] J. Feldman, T. Malkin, C. Stein, and R. A. Servedio, "On the capacity of secure network coding," in *Proc. of 42nd Annual Allerton Conf. on Comm., Control, and Computing*, 2004.

Notation	Explanation
$\mathcal{G}$	Information flow graph of a distributed storage system.
$\mathcal{V}$	Set of nodes in the information flow graph.
$C(V, V)$	Cut partitioning the set of nodes $\mathcal{V}$ in a graph into two sets $V \subset \mathcal{V}$ and $V = \mathcal{V} \setminus V$.
S	Random variable representing an incompressible source file.
n	Total number of active nodes in a distributed storage system.
k	Number of nodes a data collector connects to in order to retrieve the source file.
d	Number of nodes a new replacement node connects to during the repair process.
α	Storage capacity at each storage node in a distributed storage system.
β	Amount of data downloaded from every node participating in the repair process.
γ	The total amount of data downloaded during the repair process i.e., repair bandwidth.
Γ	Upper limit on the repair bandwidth in the bandwidth-limited regime.
D_i	All the data\messages downloaded on the replacement node v_i during the repair process.
C_i	Data stored on the node v_i.
R	Desired or achieved storage rate.
M	Capacity of the distributed storage system in the absence of an adversary.
$\mathbf{x_i}$	Data symbol or packet stored on a distributed storage system.
$\mathbf{y_i}$	Data symbol or packet, possibly corrupted by an adversary, observed by a data collector.
ℓ	Number of nodes an adversary can eavesdrop on in a distributed storage system.
b	Number of nodes an active adversary can maliciously control.
E	A set of symbols\nodes observed by an adversary by eavesdropping on ℓ nodes.
C_s	Secrecy capacity of a distributed storage system.
C_r	Resiliency capacity of a distributed storage system.

TABLE VI
TABLE OF IMPORTANT NOTATIONS

[20] S. El Rouayheb and E. Soljanin, "On wiretap networks II," in *IEEE Internat. Symp. Inform. Th. (ISIT)*, 2007.

[21] D. Silva and F. R. Kschischang, "Security for wiretap networks via rank-metric codes," in *IEEE Internat. Symp. Inform. Th. (ISIT)*, 2008.

[22] T. Ho, B. Leong, R. Koetter, M. Medard, M. Effros, and D. Karger, "Byzantine modification detection in multicast networks using randomized network coding," in *IEEE Internat. Symp. Inform. Th. (ISIT)*, pp. 616–624, 2004.

[23] S. Jaggi and M. Langberg, "Resilient network codes in the presence of eavesdropping byzantine adversaries," in *IEEE Internat. Symp. Inform. Th. (ISIT)*, pp. 541–545, 2007.

[24] Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi, M. Medard, and M. Effros, "Resilient network coding in the presence of byzantine adversaries," in *IEEE Transactions on Information Theory (special issue on information-theoretic security)*, pp. 2596–2603, 2008.

[25] H. Yao, D. Silva, S. Jaggi, and M. Langberg, "Network codes resilient to jamming and eavesdropping," in *IEEE Internat. on Network Coding (NetCod'10)*, 2010.

[26] S. Ki, T. Ho, M. Effros, and S. Avestimehr, "New results on network error correction: capacities and upper bounds," in *Information Theory and Applications Workshop (ITA'10)*, 2010.

[27] R. W. Yeung and N. Cai, "Network error correction, part I: Basic concepts and upper bounds," in *Commun. Inf. Syst, vol. 6*, pp. 19–36, 2006.

[28] R. W. Yeung and N. Cai, "Network error correction, part II: Lower bounds," in *Commun. Inf. Syst, vol. 6*, pp. 37–54, 2006.

[29] R. Koetter and F. Kschischang, "Coding for errors and erasures in random network coding," in *IEEE Transactions on Information Theory*, pp. 3579–3591, 2008.

[30] D. Silva, F. R. Kschischang, and R. Koetter, "A rank-metric approach to error control in random network coding," in *IEEE Transactions on Information Theory*, 2008.

[31] L. H. Ozarow and A. D. Wyner, "Wire-tap channel-II," in *AT&T Bell lab tech. journal vol. 63, no. 10*, 1984.

[32] R. E. Blahut, *Algebraic Codes for Data Transmission*. Cambridge University Press, 2002.

[33] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. Wiley-Interscience, 2006.

[34] S. Arunkumar and S. W. Mclaughlin, "MDS codes on erasure-erasure wire-tap channel," in *arXiv:0902.3286v1*, 2009.

Sameer Pawar received the M.S. degree in electrical engineering from Indian Institute of Science (IISc), Bangalore, India, in 2005. Since 2007, he has been with the Department of Electrical Engineering and Computer Science in the University of California at Berkeley. Prior to that, he had been with the Communications Department, Infineon Technologies India. His research interests include information theory and Coding theory for Storage and communication systems. He is recipient of Gold Medal for the Best Masters thesis in Electrical Division in IISc.

Salim El Rouayheb (S'07M'09) received the Diploma degree in electrical engineering from the Lebanese University, Faculty of Engineering, Roumieh, Lebanon, in 2002, and the M.S. degree in computer and communications engineering from the American University of Beirut, Lebanon, in 2004. He received the Ph.D. degree in electrical engineering from Texas A&M University, College Station, in 2009. He is currently a Postdoctoral Research Fellow with the Electrical Engineering and Computer Science Department, University of California, Berkeley. His research interests lie in the broad area of communications with a focus on reliable an secure distributed information systems and on the algorithmic and information-theoretic aspects of networking.

Kannan Ramchandran is a Professor of Electrical Engineering and Computer Science at the University of California at Berkeley, where he has been since 1999. Prior to that, he was with the University of Illinois at Urbana-Champaign from 1993 to 1999, and was at AT&T Bell Laboratories from 1984 to 1990. His current research interests include distributed signal processing algorithms for wireless sensor and ad hoc networks, multimedia and peer-to-peer networking, multi-user information and communication theory, and wavelets and multi-resolution signal and image processing. Prof. Ramchandran is a Fellow of the IEEE. His research awards include the Elaihu Jury award for the best doctoral thesis in the systems area at Columbia University, the NSF CAREER award, the ONR and ARO Young Investigator Awards, two Best Paper awards from the IEEE Signal Processing Society, a Hank Magnuski Scholar award for excellence in junior faculty at the University of Illinois, and an Okawa Foundation Prize for excellence in research at Berkeley. He is a Fellow of the IEEE. He has published extensively in his field, holds 8 patents, serves as an active consultant to industry, and has held various editorial and Technical Program Committee positions.