Securing Dynamic Distributed Storage Systems from Malicious Nodes

Sameer Pawar, Salim El Rouayheb, Kannan Ramchandran
Dept. of Electrical Engineering and Computer Sciences
University of California, Berkeley
{spawar, salim, kannanr}@eecs.berkeley.edu

***Abstract*—We address the problem of securing distributed storage systems against adversarial node 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 a malicious adversary who may come at different time instances during the lifetime of the storage system to corrupt the data stored on some nodes. We provide upper bounds on the maximum amount of information that can be stored safely on the system in the presence of the adversary. For an important operating regime, which we call the *bandwidth-limited regime*, we show that our upper bounds are tight and provide explicit linear code constructions. Moreover, we provide a way to shortlist the malicious nodes and expurgate the system.**

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 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] 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, $k < n$, of nodes. An important design problem for such systems arises from the individual unreliability of the constituent 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 downloads data to replace the lost one.

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 become victim of a malicious adversary that can corrupt 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 malicious attacks. The dynamic behavior of the system can jeopardize the data by making the intruder more powerful. For instance, consider the DSS of Fig. 1. Here, an adversary that can control one node (say node v_2 as depicted in Fig. 1) not only corrupts the data stored on that node, but also sends erroneous data when contacted by a new node joining the system. Thus, due to the dynamics of nodes leaving (failure) and new nodes joining (repair) the system, a malicious adversary controlling a single node may be able to "poison" the whole system.

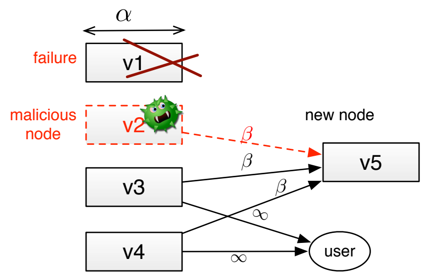


Fig. 1. A distributed storage system (DSS) with $(n, k, d) = (4, 2, 3)$. The DSS has $n = 4$ nodes, $v_1, \ldots, v_4$, each can store α units of information. A user should be able to recover his file by contacting any $k = 2$ nodes. The figure depicts the instance where node v_1 fails and the system is repaired by replacing it with the new node v_5. Node v_5 contacts $d = 3$ surviving nodes and downloads β units of data from each. Suppose there is a malicious adversary that controls node v_2. Then, the adversary, in addition to his ability to corrupt the data stored on v_2, can inject erroneous data into the system when contacted for repair, for instance, by node v_5.

We are interested in determining the fundamental information-theoretic limits on the amount of information that can be stored securely on these systems, in addition to constructing coding schemes that can achieve these limits. We focus on an *omniscient adversary* model who has complete knowledge of the data in the system but can only control a fraction of the nodes. Recent work in the literature by Kosut et al. [2] showed that studying networks with malicious nodes is a hard problem and provided instances where non-linear coding at intermediate network nodes may be necessary for achieving security. The contribution of the current 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 a general upper bound on the secure capacity of these systems and show its achievability in an important regime that we call the *bandwidth-limited*

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).

regime. Moreover, we present linear codes that can achieve capacity in this regime. 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 they permit the identification of a small list of suspected nodes guaranteed to contain the malicious ones, permitting thus the expurgation of the system.

Related work: Dimakis et al. in [3], [4] demonstrated a fundamental trade-off between repair bandwidth and storage capacity in distributed storage systems. They also introduced "regenerating codes" that have minimum repair bandwidth overhead. Constructions of *exact* regenerating codes that allow the recovery of an exact replica of the lost data were studied in [5], [6], [7], [8]. Achieving data privacy against eavesdropping in a DSS was studied in [9]. The security of storage systems with the help of a trusted verifier was investigated in [10]. Kosut et al. in [2] studied the capacity of general unicast networks with malicious nodes. A *Byzantine* adversary that can maliciously introduce errors on *links* instead of *nodes* is a well studied model in the network coding literature [11], [12].

Organization: The rest of this paper is organized as follows. In Section II, we discuss distributed storage systems and the adversary model. We provide a brief summary of our main results in Section III and explain them through example in Section IV. In Section V, we provide a capacity achieving construction for an important operating regime known as *bandwidth-limited regime* and conclude the paper in Section VI.

II. Model

Distributed Storage Systems: We consider a distributed storage system (DSS) formed of n storage nodes $v_1, \dots, v_n$, each having a storage capacity of α symbols. The storage nodes are individually unreliable and may fail over time. To guarantee a desired level of reliability, the system is repaired when a failure occurs by replacing the failed node with a new node of the same storage capacity α. The DSS should allow any legitimate user or data collector to reconstruct the file by contacting any k out of the n active storage nodes. We term this condition as the *reconstruction property* of distributed storage systems.

Repair Process: 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 to some d nodes, $d \geq k$, chosen, possibly randomly, out of the remaining active $n-1$ nodes and downloads γ symbols in total from them, which is then possibly compressed (if $\alpha < \gamma$) and stored on the node. Thus, we denote a DSS by the triplet (n, k, d), e.g., Fig. 1 depicts a DSS with parameters $(n, k, d) = (4, 2, 3)$. Note that 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. We refer to γ, the total amount of data (in symbols) downloaded for repair, as the *repair bandwidth* of the system. For load balancing requirements, we assume that the repair process is *symmetric* where the new replacement node downloads equal amount of data, $\beta = \gamma/d$ symbols, from each of the contacted nodes.

Adversary Model: We assume the presence of an active adversary Calvin who can control a certain number of nodes in the DSS. Calvin is assumed to be omniscient [11], *i.e.*, he knows the stored file and the data stored on the individual nodes. Moreover, Calvin can control b nodes in total, where $2b < k$, that can include some of the original nodes $v_1, \dots, v_n$, and/or some replacement nodes $v_{n+1}, v_{n+2}, \dots$[2]. Calvin can maliciously alter the data stored on the nodes under his control. He can also send erroneous outgoing messages when contacted for repair or reconstruction.

III. Results Summary

We define the *resiliency capacity* C_r of the DSS, as the maximum amount of data that can be stored on the DSS and *reliably* delivered to a legitimate data collector. Our first result gives a general upper bound on the resiliency capacity.

Theorem 1: [Upper Bound] For an (n, k, d) DSS with an omniscient adversary controlling b nodes, with $0 < 2b < k$, the resiliency capacity $C_r(\alpha, \gamma)$ is upper bounded as,

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

with $\beta = \gamma/d$. For $k \leq 2b, C_r(\alpha, \gamma) = 0$.

The above upper bound can be interpreted as a network version of the Singleton bound where $2b$ nodes among the k nodes observed by the user are rendered obsolete. The value of the summand in the bound can be less then α due to possible correlation between the data stored on each node. The detailed proof of Theorem 1 can be found in [13].

In many scenarios, the repair bandwidth becomes the bottleneck in the system rather than the node storage capacity. Therefore, we identify an important operational regime for the DSS which we call the *bandwidth-limited regime* [9]. In this regime there is an imposed upper limit Γ on the repair bandwidth, *i.e.*, $\gamma \leq \Gamma$, while the storage capacity per node can be taken to be as large as desired[3]. The resiliency capacity in the bandwidth-limited regime is defined as

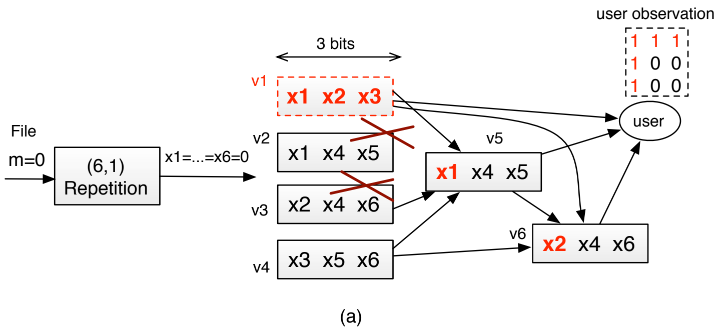
$$C_r^{BL}(\Gamma) := \sup_{\gamma \leq \Gamma, \alpha \geq 0} C_r(\alpha, \gamma).$$

Note that if the parameter d is a system design choice, the upper bound of (1) 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 2.

[1]Multiple nodes failing simultaneously is a rare event. When this occurs, the DSS implements an "emergency" repair process where the replacement nodes act as data collectors and download data from k active nodes.

[2]A new node replacing a failed node that was compromised may or may not be compromised itself. The only constraint that our model supposes is that at any given time Calvin can control at most b nodes.

[3]In the sequel, we show that no gain in the system resiliency capacity can be achieved by taking the storage capacity per node α to be larger than Γ.



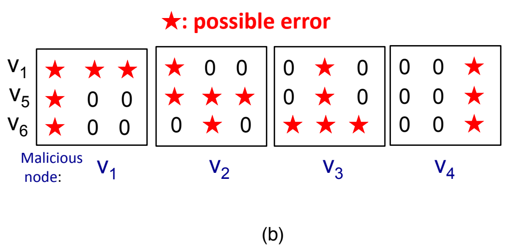


Fig. 2. Example of a capacity achieving code for a DSS with $(n, k, d) = (4, 3, 3)$ and $b = 1$ node controlled by the adversary Calvin. This code is used to reliably store 1 bit of information designated by m. (a) Assume that node v_1 is the compromised node and that nodes v_2 and v_3 fail and are replaced by nodes v_5 and v_6, respectively. Consider a user contacting nodes v_1, v_5, v_6. Although Calvin can directly control 3 bits, he can induce 5 errors in this user observation due to the repair process. Therefore, a simple majority decoding rule will fail here. (b) The four error patterns that can be introduced by Calvin in the user observation. Each pattern contains 5 possible error locations and corresponds to a distinct possibility of compromised node. These patterns can be leveraged to decode correctly the message.

Theorem 2: For an (n, k, d) DSS, with $d = n-1$, operating in the bandwidth-limited regime with an omniscient adversary controlling b nodes, $2b < k$, the resiliency capacity of the DSS is given by

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

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

Our proposed capacity achieving codes are linear and characterized by a desired 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. Moreover, these codes permit the identification of a small "suspect" list guaranteed to contain the malicious nodes which can be then discarded from the system.

In the next section, we explain the above results using a simple example and prove Theorem 2.

IV. Example

Consider the example of a DSS with $(n, k, d) = (4, 3, 3)$ and $\alpha = \gamma = 3$ bits. 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. Note that, although Calvin controls a single node here, he can corrupt the data on other nodes in the system by sending corrupted data to the new nodes joining the system during the repair process.

By Theorem 1, we know that the resiliency capacity of this system is upper bounded by 1 bit. We now provide a code that can reliably store 1 bit of information on the DSS, thereby showing that $C_r = 1$. This code, depicted in Fig 2(a), consists of the concatenation of an outer $(6, 1)$ repetition code followed by a special repetition code that was introduced in [5] by Rashmi et al. and which we refer to as the RSKR-repetition code (see Fig. 3). To repair the system, the replacement node recovers the lost bits by downloading from the remaining three active nodes the bits with same indices.

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 data collector 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 $i = 1, \dots, 6$, in Fig 2(a), corresponding to the message $m = 0$. Suppose node v_1 is compromised and is controlled by the adversary Calvin. 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 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. 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 2(a) which includes 5 errors. An important point to note here is that the repair scheme is based on the RSKR-repetition code and is fixed irrespective of the possible errors in the bits downloaded during repair.

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 one of the four patterns depicted in Fig. 2(b). To decode, the user will go over each of the possible patterns and use it as a "mask" to puncture its data, *i.e.*, it will delete the bits in the 'star' positions. If all the bits with distinct indices in the

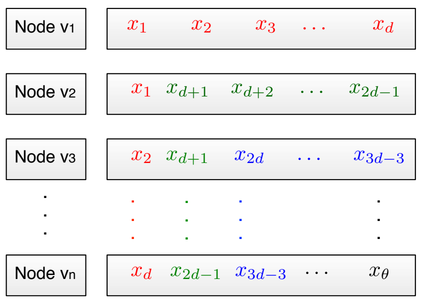

Node v1
x_1 x_2 x_3 $\cdots$ x_d
Node v2
x_1 x_{d+1} x_{d+2} $\cdots$ x_{2d-1}
Node v3
x_2 x_{d+1} x_{2d} $\cdots$ x_{3d-3}
Node vn
x_d x_{2d-1} x_{3d-3} $\cdots$ x_θ

Fig. 3. The structure of the RSKR-repetition code of Rashmi et al [5] 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 $nd = 2\theta$.

remaining bits agree, the decoder will output a result. Note that there is always a pattern where all he unpunctured bits will agree.

To prove that this decoder will never make an error, we have to show that whenever the unpunctured bits agree, they must agree on the correct value. 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 code operates in the following way. First, it finds a set $T^* \in \{T_1, \dots, T_4\}$ whose all elements agree to either 0 or 1. Then, it declares that message $m = 0$ or $m = 1$ was stored accordingly. 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. Hence with the proposed code and "masking" decoder, we can store 1 bit of data securely. In the next section, we generalize this construction to obtain capacity achieving codes for the bandwidth-limited regime.

V. Capacity-Achieving Codes

We now give the proof of Theorem 2 by explicitly constructing codes that can achieve the resiliency capacity in the bandwidth-limited regime for systems with $d = n-1$. It suffices to show the achievability for the normalized case of $\beta = 1$, *i.e.,* $\Gamma = n-1$. In this case, our capacity achieving code uses a node storage capacity $\alpha = n-1$ symbols.

The code is a generalization of the code used in the previous example. The $(6, 1)$ repetition code in the example is replaced by a (θ, R) MDS code where $R := C_r^{BL}(n-1) = \sum_{i=2b+1}^{k}(n-i)$ and $\theta = \frac{n(n-1)}{2}$. In the second layer, the output of the MDS code is stored on the DSS using the general RSKR-repetition code depicted in Fig 3. Note that node failures are repaired using the RSKR-repetition code irrespective of the possible errors introduced by Calvin.

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 twice due to the structure of the RSKR-repetition code. The adversary can corrupt both copies of the repeated symbols stored on the b compromised nodes. Therefore, the data collector focuses on M symbols with distinct indices out of the $(n-1)k$ total observed symbols 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 with the errors introduced by the adversary. The minimum distance of the MDS code $\mathcal{X}$ is,

$$d_{\min}(\mathcal{X}) = M - R + 1 = \sum_{i=1}^{2b}(n-i) + 1. \qquad (3)$$

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.

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 3: **Puncturing a vector:** Consider a vector $\vec{v} \in \mathbb{F}^N$ for some field $\mathbb{F}$. Let $I \subset \{1, 2, \dots, N\}, |I| = p$, 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 4: **Puncturing a Code:** Consider a code $\mathcal{C}$ in $\mathbb{F}^N$. Let $I \subset \{1, 2, \dots, N\}, |I| = p$, 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.,* $\mathcal{C}_I := \{\vec{x}_I | \vec{x} \in \mathcal{C}\}$.

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 $B \subset \{v_1, \dots, v_n\}$ with $|B| = b$. For each such set B, we define $I_B \subset \{1, 2, \dots, \theta\}$ to be the set of the indices of the symbols stored on the nodes in B. For instance, in Example IV, $B = \{v_1\}$, $I_B = \{1, 2, 3\}$.

The decoding algorithm proceeds in the following way:

Step 1: The data collector connecting to k nodes observes a total of $(n-1)k$ symbols. It then selects any M symbols with distinct indices $Y \in \mathbb{F}_q^M$, and uses it for decoding. In Fig 2(a), 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 corrupted 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.

Step 2: For each $B \subset \{v_1, \dots, v_n\}, |B| = b$, find I_B.

Step 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 (3)). Hence, $\mathcal{X}_{I_B}$ is an MDS code.

Step 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 $\vec{\sigma}_{I_B} = H_{\mathcal{X}_{I_B}} Y_{I_B}^T$.

Step 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 a decoder for the code $\mathcal{X}_{I_B}$.

Proof of Correctness: Notice 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 (not known to the user). 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 malicious 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 $X_{I_{B^*} \cup I_B} = X'_{I_B \cup I_{B^*}}$. But, from the RSKR-repetition code structure we know that $|I_{B^*} \cup I_B| \leq \sum_{i=1}^{2b}(n-i)$. This implies that $d_{\min}(\mathcal{X}) \leq \sum_{i=1}^{2b}(n-i)$ which contradicts equation (3).

Remark 5 (Decoder complexity): The complexity of the proposed decoder is exponential in the number b of malicious nodes. Therefore, it may not be practical for protecting systems against powerful adversaries, *i.e.*, 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 2.

Remark 6 (Expurgation of malicious nodes): As shown above, the proposed decoder always decodes to the correct message, and thus, can identify the indices of the erroneous symbols. The data collector can then report this set of indices to a central authority (tracker) in the system. This authority can further combine such information from multiple data collectors, and knowing the RSKR-repetition structure (see Fig. 3), it forms a list of suspected nodes that will surely include the malicious nodes. 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.

VI. Conclusion

We have investigated the fundamental performance limits of distributed storage systems under *repair dynamics* in the presence of malicious nodes. We presented a general upper bound on the resiliency capacity of the system and constructed codes that can achieve this bound in the bandwidth-limited regime. In general, it is known that intermediate nodes in the network may have to perform non-linear operations to achieve the secure capacity [2]. In contrast, our capacity-achieving codes are linear and enjoy a separation-based structure consisting of file precoding without changing the behavior of the system nodes.

This work is a first step towards understanding the security of distributed storage systems and dynamic information systems in general. Many related questions remain open:

1) The proposed decoder for our capacity-achieving code has a decoding complexity that grows exponentially in the number of malicious nodes. Whether there is a exists a polynomial-time decoder is still an open question. Moreover, can the decoding complexity be reduced by allowing the replacement nodes in the system to perform "smarter" operations than simply storing their incoming data?

2) Our capacity result and codes for the bandwidth-limited regime hold for the case of $d = n-1$, *i.e.*, when all the nodes in the system have to be contacted for repair, which can be prohibitive in large-scale systems. Recently, exact regenerating codes were constructed in [14] for the cases of $d < n-1$. Can these codes be used as inner component codes for achieving security for $d < n-1$. Unfortunately, our constructions here cannot be readily generalized to that case.

3) Our current approach to securing a dynamic system consists of transforming it into static system using exact codes. However, this is not always possible [8]. In this case, is the upper bound of Theorem 1 tight, and what codes would achieve it?

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

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

[4] A. G. Dimakis, P. B. Godfrey, M. J. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," in *INFOCOM'07*, 2007.

[5] K. Rashmi, N. B. Shah, P. V. Kumar, and K. Ramchandran, "Exact regenerating codes for distributed storage," in *Allerton Conference on Control, Computing, and Communication, Urbana-Champaign, IL*, 2009.

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

[7] C. Suh and K. Ramchandran, "Exact regeneration codes for distributed storage repair using interference alignment," in *Proc. IEEE Intl Symp. on Information Theory (ISIT)*, 2010.

[8] N. B. Shah, K. V. Rashmi, P. V. Kumar, and K. Ramchandran, "Explicit codes minimizing repair bandwidth for distributed storage," in *ITW*, 2010.

[9] S. A. Pawar, S. El Rouayheb, and K. Ramchandran, "On secure distributed data storage under repair dynamics," in *IEEE Internat. Symp. Inform. Th. (ISIT'10)*, 2010.

[10] 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'10)*, 2010.

[11] Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi, M. Medard, and M. Effros, "Resilient network coding in the presence of byzantine adversaries," pp. 2596–2603, 2008.

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

[13] S. A. Pawar, S. El Rouayheb, and K. Ramchandran, "Securing dynamic distributed storage systems against eavesdropping and adversarial attacks," in *arXiv:1009.2556v2*, 2011.

[14] K. V. Rashmi, N. B. Shah, and P. V. Kumar, "Optimal exact-regenerating codes for distributed storage at the msr and mbr points via a product-matrix construction," in *arxiv:1005.4178*, 2010.