ISIT'2013 1569720187

Polytope Codes for Distributed Storage in the Presence of an Active Omniscient Adversary

Oliver Kosut
School of Electrical, Computer & Energy Eng.
Arizona State University, Tempe, AZ
Email: okosut@asu.edu

***Abstract*—Distributed storage systems are studied in the presence of an active omniscient adversary. The adversary is able to control several storage nodes in the system and alter their behavior. A Polytope code is proposed to handle such an adversary, and it is used to prove a lower bound on the overall storage capacity. Polytope codes have been shown to outperform linear codes over a finite field in defeating active adversaries. In a Polytope code, linear operations are performed over the integers rather than a finite field. This allows examinations of cross-covariances as a sort of parity check, which can improve error detection and correction without sacrificing asymptotic rate.**

I. Introduction

Distributed storage systems (DSS) are becoming increasingly important for reliably storing large amounts of data. They are used in numerous cloud services, and across peer-to-peer networks. DSSs counteract individually unreliable components by maintaining redundant storage nodes, so that if any one node fails, an equivalent replacement can be constructed. In a DSS spread across a wide geographic area—whether because it is split between multiple data centers or it is stored across a peer-to-peer network—communication between storage nodes must occur over the internet. As such, there is potential for the system to be compromised by a malicious intruder. This paper addresses a particular type of intruder: an active adversary that covertly takes control of a subset of storage nodes in a DSS, and causes these compromised nodes to deviate from their specified actions. We develop codes to satisfy all the usual constraints on DSSs—replacing failed nodes with equivalent ones, and maintaining file integrity so that the original data can be recovered at any point—even when some nodes are compromised by this adversary.

The principle result in the study of distributed storage codes is that DSS problems can be cast as *network coding* problems. This connection was first made in the pioneering work of [1], leading to a long line of work using network coding ideas to design DSS codes (see survey [2]). Meanwhile, active adversarial attacks on network coding were studied in [3], [4], which examine a network in which a fixed number of links are subject to adversarial errors. In [5], DSSs were secured against adversaries with an interactive scheme in which random hashes of stored data are sent to an independent verifier. The present paper most closely follows [6], which studied the same problem with more limited interaction between nodes (in particular, the communication graph is acyclic). Several adversarial models were considered in [6], including a passive eavesdropper and the omniscient active adversary considered here. For the latter, a simple cut-set upper bound on the storage capacity of the DSS was found. They also proposed a linear code that achieves this bound in the so-called *bandwidth limited regime*: that is, when the storage capacity of each individual node is large compared to the bandwidth of the communication links between nodes. This problem was also studied in [7], which investigated selfish as well as polluting Byzantine attacks, including when multiple nodes are regenerated once. In [8], errors in DSSs are handled by varying the amount of data downloaded when decoding. [9] provided an approach based on rank-metric codes.

Our approach is to use *Polytope codes* to address the same problem. Polytope codes were originally introduced in [10], and then substantially expanded upon in [11], [12]. Polytope codes are in effect linear codes that operate over the integers rather than a finite field. The advantage of this is that cross-covariances can be calculated between received data packets. The covariances can be compared against their anticipated value as a sort of "parity check". If this check fails—i.e. the covariance does not match—the adversary must have corrupted one of the incoming packets. Covariance checks rely on operations being performed over the integers rather than a finite field, since in a finite field covariances do not have the same meaning. The specifics of a Polytope code for the DSS problem are explained in Sec. IV. The codes described in this paper differ slightly from those in [10], [11], [12] in that they use covariances as summary statistics for sequences instead of types, but the principles are the same.

It was shown in [10] that when the adversary controls a fixed number of *nodes* in a network (rather than a fixed number of links as in [3], [4]), Polytope codes can outperform classical linear codes on a finite field. A similar result was found in [13], which studied an adversary that can control links, but links with unequal capacity, and found that finite field linear codes are insufficient to achieve capacity. In the DSS problem, it is much more natural to think of the adversary controlling nodes rather than links. This is especially true because a single adversarial storage node may control many links; indeed, that node may transmit an unbounded number of messages if it continues to be contacted when new nodes are formed. Hence, the number of links controlled by the adversary may be very large, even though the number of corrupted nodes is small. This makes Polytope codes naturally suited to this problem.

We propose a specific Polytope code for DSSs, and use it to prove a lower bound on storage capacity. This code operates across a wide variety of parameters, including well outside the bandwidth limited regime of [6]. In some cases our lower bound matches the upper bound of [6], including cases where our code achieves the same rates as the achievable scheme of [6] but requiring substantially less node capacity.

The paper is organized as follows. Section II describes the problem. Section III states several known bounds on storage capacity for a DSS, as well as our main result lower bounding storage capacity. In Section IV we prove our main result by describing our proposed Polytope code for DSSs. We conclude in Section V.

II. Problem Description

A. Distribution Storage System

A DSS is a collection of storage nodes, each holding a portion of a single data file. We assume each node has capacity α, meaning it can store up to αm bits. (All results will be in the asymptotic limit as $m \to \infty$.) At any given time, there are n active storage nodes, but individual nodes are unreliable and may fail. When one node fails, a new node is created to replace it. The new node contacts d existing nodes and downloads messages from each one, from which it constructs data to store. The communication links used to transmit these messages each have capacity $\beta \leq \alpha$, again meaning they can carry βm bits. The key property that must be maintained is that at any time in this evolution, a data collector (DC) may contact any $k \leq d$ existing nodes, download their contents, and perfectly reconstruct the original file. The specific evolution of the system, such as which nodes fail, which nodes are contacted when a new node is formed, and when the DC collector downloads data to reconstruct the file, is arbitrary and unknown *a priori*. Note that we are considering *functional repair* rather than *exact repair* or *exact repair of systematic parts* (see [2]).

B. Adversary Model

We assume the presence of an *active omniscient adversary*. This adversary is able to take control of a subset of the storage nodes, and alter any message sent from any of those nodes. This includes messages sent when constructing a new node, as well as data downloaded to a DC. Once a code is fixed, all honest (non-adversarial) nodes behave according to this code, but adversarial nodes may deviate from the code by replacing outgoing transmissions with arbitrary messages. The adversary is omniscient in the sense that it knows the complete stored file, as well as every aspect of the code used by the honest nodes. We further assume that honest nodes cannot generate random numbers unknown to the adversary. These assumptions err on the side of pessimism so as to maintain performance even if they do not hold. The adversary may control up to b nodes at any given time. That is, as nodes fail and are replaced, the adversary might continue taking control of new nodes, but at no moment does it control more than b nodes. This is a slightly more pessimistic assumption than in [6], in which the adversary could control a total of b nodes over the entire evolution of the system, whether or not they existed simultaneously.

We say a rate R is *achievable* for a DSS problem with parameters $(\alpha, \beta, n, k, d, b)$ if for some m there exists a code such that a file consisting of mR bits can always be reconstructed, no matter the evolution of the system or the adversary actions. The *storage capacity* C is the supremum of all achievable rates. In the sequel, we make the simplifying assumption that $k = d = n - 1$. Moreover, we assume α and β are integers. Note that if α and β are scaled equally, by definition the storage capacity scales by the same amount.

C. Flow-Graph Representation

We adopt the now-standard flow-graph representation of the DSS problem that was used in [1]. We construct a directed acyclic graph $\mathcal{G}$ with capacity constrained links. The graph $\mathcal{G}$ has three types of nodes: a source node s, input and output nodes x^i_{in} and $\mathsf{x}^i_{\text{out}}$ for each storage node i, and data collector nodes DC_j. Storage nodes and data collectors are indexed by integers i, j. For each storage node i, there is a link $(\mathsf{x}^i_{\text{in}}, \mathsf{x}^i_{\text{out}})$ with capacity α representing its limited storage capability. When a new node j is formed and node i is among the nodes it downloads data from, there is a link $(\mathsf{x}^i_{\text{out}}, \mathsf{x}^j_{\text{in}})$ with capacity β representing the communication link. There are infinite capacity links from the source s to each of the initial n nodes representing the initial upload of the data file to the storage nodes. Finally, there are infinite capacity links to each data collector DC_j from each of the k nodes it contacts representing the downloading of data to a DC.

III. Bounds on Storage Capacity

We first state the upper and lower bounds on storage capacity found in [6]. With a combination of a cut-set bound and the Singleton bound, [6, Theorem 6] states that

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

Making use of a linear code, [6, Theorem 7] states that when $d = n - 1$ and $\alpha \geq (n-1)\beta$,

$$C \geq \sum_{i=2b+1}^{k} (n-i)\beta. \tag{2}$$

Note that (2) matches (1) when $\alpha \geq (n-1)\beta$, i.e. in the bandwidth limited regime.

The following theorem is our main achievability result. The proof appears in Section IV.

Theorem 1. *The storage capacity C is lower bounded by*

$$\min\left\{ \sum_{i=z_b+1}^{k} \min\{(d-i+1)\beta, \alpha\}, (d-b)\beta, (k-b)\alpha \right\}. \tag{3}$$

where $z_b := (\lfloor \frac{b}{2} \rfloor + 1)(\lceil \frac{b}{2} \rceil + 1)$.

Observe that for $b \leq 3$, $z_b = 2b$. Hence for this range of b our achievable result matches the upper bound as long as the

right hand side of (1) does not exceed $\min\{(d-b)\beta, (k-b)\alpha\}$. In the simple case that $d = k = n-1$, this determines the exact capacity for all α, β when $b = 1, n \leq 5$ or $b = 2, n \leq 7$ or $b = 3, n \leq 10$. Moreover, in some cases Theorem 1 achieves the same rate as (2) but with smaller α. For example, when $n = 5, b = 1$, and again $d = k = n - 1$, (2) achieves rate 3β (the capacity) only for $\alpha = 4\beta$, but (3) achieves the same rate for $\alpha = 2\beta$.

IV. Achievable Scheme

Our achievable scheme uses a Polytope code. Polytope codes are similar to classical finite field linear codes in that new packets are formed by taking linear combinations of previous packets. They differ in that in a Polytope code, linear operations are performed over the integers rather than a finite field. In the network coding literature, the term *non-linear* refers to codes in which operations are not linear over a finite field; thus Polytope codes are in this sense non-linear, even though most operations are linear over the integers. In a Polytope code, the linear constraints that result from these operations specify a hyperplane over the integers, but to ensure that there are only a finite number of messages, we place bounds on this hyperplane (hence "Polytope": a bounded hyperplane).

We now describe construction of a code to achieve the bound in Theorem 1. Let R be an integer not exceeding (3). Fix positive integers N and K. In the Polytope code, each packet has a *payload*, containing the primary data that is stored or transmitted in the DSS. This payload is an array of nonnegative integers with N columns, where N is fixed across the code. In order to achieve rate R, N needs to go to infinity. The integer K provides a bound on the integers stored in the payloads; K must also go to infinity.

Let $M = (K+1)^{RN}$. We refer to the data in the file to be stored in the DSS as f, which is an integer in $\{1, \ldots, M\}$. Define codewords $c_f \in \{0, \ldots, K\}^{R \times N}$ for $f \in \{1, \ldots, M\}$ where all codewords are distinct and

$$\{c_1, \ldots, c_M\} = \{0, \ldots, K\}^{R \times N}. \tag{4}$$

The array c_f is the initial representation of the file that will be sent through the network.

It will be important to refer to covariances of c_f and its linear combinations. Let $\Sigma_f := c_f c_f^T$. Choose $F \in \mathbb{N}^{n\alpha \times R}$ such that all $R \times R$ submatrices have full rank. All packets transmitted in this code have the following form:

$$p := (\Sigma_f, A, AFc_f). \tag{5}$$

where $A \in \mathbb{N}^{s \times n\alpha}$ and s is the capacity of the link along which p is sent. We say that A *paramaterizes* the packet p. We refer to the first two elements of (5) as the *header* and the third element as the *payload*. The above description of packets assumes all nodes are honest. Obviously when an adversary emits a packet it may alter either the header or the payload. This corruption may also affect downstream packets, so that in particular the payload held by any particular packet may differ from the true value of AFc_m.

We now calculate the number of bits required to store the packet p. Entries of Σ_m are nonnegative integers no larger than NK^2. Hence the number of bits required to store Σ_m is $R^2\lceil\log(NK^2)\rceil$. The number of bits to store A is no more than $sn\alpha\lceil\log(\|A\|_\infty + 1)\rceil$ where $\|\cdot\|_\infty$ is the elementwise infinity norm. The number of bits required to store AFc_m is no more than

$$sN\lceil\log(\|AFc_f\|_\infty + 1)\rceil \leq sN\lceil\log(\|A\|_\infty \|F\|_\infty n\alpha RK + 1)\rceil \tag{6}$$

Recall that a link with capacity s can hold sm bits. We choose m so that sm exceeds the total number of bits in a packet p for any s. Totaling the number of bits required to store a packet, we choose m to be the smallest value satisfying

$$m \geq \max_{s \geq 1} \frac{1}{s} R^2\lceil\log(NK^2)\rceil + n\alpha\lceil\log(\|A\|_\infty + 1)\rceil + N\lceil\log(\|A\|_\infty \|F\|_\infty n\alpha RK + 1)\rceil. \tag{7}$$

The right hand side of (7) is satisfied at $s = 1$. The rate achieved by this code is $\frac{1}{m}\log M = \frac{1}{m} NR\log(K+1)$. This can be made arbitrarily close to R for large enough N and K.

Since the covariance matrix Σ_f is included in all packets, all nodes have access to it. This is true even in the presence of the adversary. Since adversarial nodes always make up a minority of the transmitting nodes when a new node is formed (otherwise the cut-set bound would be 0), the new node can perform majority rule and determine the true value of Σ_f.

Given a set of links $\mathcal{A}$, let the matrix A be the vertical concatenation of all matrices parameterizing the packets sent along these links. Also let x be the vertical concatenation of the payloads in all the packets sent along these links. If all nodes were honest, then $x = AFc_f$, meaning the covariance for the packets sent along $\mathcal{A}$ is given be

$$\Sigma_{\mathcal{A}} := (Ac_f)(Ac_f)^T = A\Sigma_f A^T. \tag{8}$$

We refer to $\Sigma_{\mathcal{A}}$ as the *should-be* covariance. Because of the adversary, it may be that $x \neq AFc_f$, so we also consider the *actually-is* covariance, given by

$$\tilde{\Sigma}_{\mathcal{A}} := xx^T. \tag{9}$$

By our argument above that every node has access to Σ_f, any node that can view the packets sent along links in $\mathcal{A}$ can calculate both the should-be and the actually-is covariance associated with those packets. In particular, when a new storage node is formed, it can determine both covariance matrices for its set of incoming links. Any difference between them must have been caused by an upstream adversarial node.

The main advantage of using a Polytope code rather than a finite field linear code is the property given in the following lemma, which is a corollary to [12, Theorem 3]. In the statement of the lemma, $x^\star$ represents the payload of a packet or a group of packets as it should be, and x the payload as it actually is.

Lemma 2. *Given $x^\star, x \in \mathbb{N}^{s \times N}$, let $\Sigma_x = (x^\star)(x^\star)^T$ and $\tilde{\Sigma}_x = xx^T$. If $\tilde{\Sigma}_x = \Sigma_x$, then for any linear operation H, $Hx^\star = 0$ implies $Hx = 0$.*

Proof: $\|Hx\|_2^2 = x^T H^T H x = \mathrm{tr}(Hxx^T H^T) = \mathrm{tr}(H\tilde{\Sigma}_x H^T) = \mathrm{tr}(H\Sigma_x H^T) = \mathrm{tr}(H(x^\star)(x^\star)^T H^T) = \|Hx^\star\|_2^2 = 0$. Hence $Hx = 0$. ∎

We now describe operation of the code. In the following, we refer to random choices for linear operators B. The elements of these linear operations are paramaterized by a positive integer q. We take these random choices to be choices made about code design, rather than choices made at run time. (Recall that honest nodes cannot generate random numbers unknown to the omniscient adversary.) We argue that with positive probability, the choices form a code with the required properties, hence there is at least one code satisfying the necessary conditions.

Data stored on initial nodes: Given the file f, the encoder finds c_f and forms n initial packets, as described in (5), each of size α. The packet put on the ith node is parameterized by

$$[0_{\alpha\times(i-1)\alpha}\ I_\alpha\ 0_{\alpha\times(n-i)\alpha}] \tag{10}$$

where $0_{a\times b}$ is the a by b all zeros matrix, and I_a is the a by a identity matrix.

Transmissions to form new node: When a new node is formed, it contacts the d existing nodes and receives a packet from each one. The packet sent from node i to node j is constructed as follows. Let $A_i \in \mathbb{N}^{\alpha\times n\alpha}$ be the matrix parameterizing the packet stored on node i, and x_i the payload of this packet. A matrix $B_{i\to j} \in \{0,\dots,q\}^{\beta\times\alpha}$ is chosen randomly and uniformly. The packet transmitted by node i is given by $p_{i\to j} = (\Sigma_f, B_{i\to j}A_i, B_{i\to j}x_i)$.

Formation of new node: When node j is formed, the packet it stores is formed as follows. Let $\mathcal{A}$ be the set of d incoming links to node j. Let A be the concatenation of the matrices parameterizing the packets sent along these links, and let $x_{\to j}$ be the concatenation of the payloads of these packets. Node j calculates Σ_f using majority rule, and then forms $\Sigma_{\mathcal{A}}$ and $\tilde{\Sigma}_{\mathcal{A}}$ as described in (8)–(9). The relationship between these two covariances decides how node j forms its stored packet in the following way. Node j forms an undirected graph with vertices $\mathcal{A}$ and edge set $\mathcal{E}_j$ we refer to as the *syndrome graph*. The edge set $\mathcal{E}_j$ is determined as follows. For $u, v \in \mathcal{A}$, let $(\Sigma_{\mathcal{A}})_{u,v}$ be the $2\beta \times 2\beta$ submatrix of $\Sigma_{\mathcal{A}}$ representing to the cross-correlation of the packets sent on u and v. Define $(\tilde{\Sigma}_{\mathcal{A}})_{u,v}$ similarly. Now the edge set is given by

$$\mathcal{E}_j := \{(u,v) : (\Sigma_{\mathcal{A}})_{u,v} = (\tilde{\Sigma}_{\mathcal{A}})_{u,v}\}. \tag{11}$$

Note that if $(u,v) \notin \mathcal{E}_j$, then one of the packets on u or v must have been influenced by the adversary. Hence the set of uncorrupted packets will form a clique in the syndrome graph. Given the syndrome graph, node j does the following

1) Let $\mathcal{A}'$ be the set of links $u \in \mathcal{A}$ such that u is contained in a clique of size $d-b$ in the syndrome graph.
2) Let $\mathcal{A}^\star$ be the set of links $u \in \mathcal{A}'$ such that $(u,v) \in \mathcal{E}_j$ for all $v \in \mathcal{A}' \setminus \{u\}$.
3) Let $A^\star$ be the concatenation of matrices parameterizing the packets sent on links $\mathcal{A}^\star$, and let $x^\star$ be the concatenation of payloads of these packets. We choose a matrix $B_j \in \{0,\dots,q\}^{\alpha\times|\mathcal{A}^\star|\beta}$ randomly and uniformly. The packet stored at node j is given by $p_j = (\Sigma_f, B_j A^\star, B_j x^\star)$.

Decoding at a data collector: To decode the original message, the DC downloads the packets stored on k nodes. The decoding procedure is similar to the procedure described above to form a new node. A syndrome graph is formed, and $\mathcal{A}'$, $\mathcal{A}^*$, A^*, and x^* are calculated as described in steps (1)–(3) above (with k replacing d). Let $\hat{A}$ be the first R rows of $A^\star$, such that $\hat{A}$ is $R \times R$, and let $\hat{x}$ be the first R rows of $x^\star$. The DC constructs its estimate of the original file $\hat{f}$ such that $c_{\hat{f}} = \hat{A}^{-1}\hat{x}$.

Proof of correctness: Let $\mathcal{G}'$ be a subgraph of $\mathcal{G}$ including all its nodes and all its links except those of the form $u = (\mathsf{x}^i_{\text{out}}, \mathsf{x}^j_{\text{in}})$ where $u \notin \mathcal{A}^\star$ when node j is constructed, or those of the form $u = (\mathsf{x}^i_{\text{out}}, \mathsf{DC}_j)$ where $u \notin \mathcal{A}^\star$ when DC_j decodes. To prove that the above code allows the DC to recover the file f, we make two claims proved in the Appendix:

Claim 1: All links in $\mathcal{G}'$ hold truthful data, even if sent by an adversarial node. Note this implies all data stored on honest nodes is truthful.

Claim 2: When a new storage node is constructed $|\mathcal{A}^\star| \geq d - z_b$, and when a DC decodes $|\mathcal{A}^\star| \geq k - z_b$. Recall z_b is defined in the statement of Theorem 1.

By Claims 1 and 2, $\mathcal{G}'$ effectively forms a DSS without an adversary, but with d and k each reduced by z_b. It is shown in [1] that the min-cut of this network is at least

$$\sum_{i=1}^{k-z_b} \min\{(d - z_b - i + 1)\beta, \alpha\}. \tag{12}$$

This quantity is precisely the first element of the outer minimum in (3), so it upper bounds R. Since all new packets in $\mathcal{G}'$ are formed by random linear combinations of the incoming packets, by arguments similar to those used in classical linear network coding [14], for large enough q the end-to-end linear transformation has rank R with high probability, so the original file can be decoded at the DC.

V. Conclusion

We proposed a Polytope code for distributed storage systems in the presence of an active omniscient adversary. This code improves the state-of-the-art achievable storage rates for some regimes in which the node storage capacity is on the order of the bandwidth between nodes, but there is still a substantial gap to the best known upper bound. It may be possible to design more elaborate Polytope codes that achieve higher rates, but a complete theory of Polytope codes remains elusive.

Appendix

Proof of Claim 1: By construction, the initial honest nodes store only truthful data. We proceed by induction: assume all existing honest nodes hold truthful data, and show that when a new node j is formed, any packet sent along $u \in \mathcal{A}^\star$ must be truthful, even if sent by an adversarial node. Let $\mathcal{H}$ be the set of links from honest nodes. Let $\mathcal{T}$ be the set of incoming links from adversarial nodes. We assume without loss of generality

that $|\mathcal{T}| = b$. If there are fewer than b adversarial nodes, we may assume that some of the honest nodes are adversarial but simply act truthfully. Let $\mathcal{H}$ be the set of incoming links from honest nodes. Certainly $|\mathcal{H}| \geq d-b$. Since the links in $\mathcal{H}$ hold only truthful data, $\mathcal{H}$ forms a clique in the syndrome graph for node j, so $\mathcal{H} \subset \mathcal{A}'$. Let u be a link from an adversarial node. If $u \in \mathcal{A}^\star$, then $(u,v) \in \mathcal{E}_j$ for all $v \in \mathcal{H}$. Hence $\{u\} \cup \mathcal{H}$ forms a clique in the syndrome graph. That is, $\Sigma_{\{u\}\cup\mathcal{H}} = \tilde{\Sigma}_{\{u\}\cup\mathcal{H}}$, where the two covariances are defined in (8)–(9). By Lemma 2, the payloads that appear on the links in $\{u\} \cup \mathcal{H}$ satisfy the same linear constraints as if they were uncorrupted. The number of elements of $\{u\}\cup\mathcal{H}$ is at least $d-b+1$. Since R does not exceed (3), $R \leq (d-b)\beta$, so there are at least β linear constraints among these elements. Max-flow min-cut arguments can be used to show that these linear constraints enforce the data sent on u to be truthful. The same argument can be used for DC decoding by employing the $R \leq (k-b)\alpha$ condition.

Proof of Claim 2: Consider the construction of a new node. For any set of edges $\mathcal{E} \subset \mathcal{A} \times \mathcal{A}$, let

$$\mathcal{N}(u, \mathcal{E}) := \{v \in \mathcal{A}' : (u,v) \notin \mathcal{E}'\}. \tag{13}$$

Note that $\mathcal{N}(u, \mathcal{E}_j) = 0$ for $u \in \mathcal{A}^\star$ and $\mathcal{N}(u, \mathcal{E}_j) \geq 1$ for $u \in \mathcal{A}' \setminus \mathcal{A}^\star$. We construct a set of edges $\mathcal{E}'_j \supset \mathcal{E}_j$ as follows. Begin by setting $\mathcal{E}'_j = \mathcal{E}_j$. If there exists a pair $u, v \in \mathcal{A}'$ such that $(u,v) \notin \mathcal{E}'_j$ and $|\mathcal{N}(u, \mathcal{E}'_j)| > 1, |\mathcal{N}(v, \mathcal{E}'_j)| > 1$, then add (u,v) to $\mathcal{E}'_j$. Repeat until there is no such pair u, v. Note that for the resulting $\mathcal{E}'_j$, for $u \in \mathcal{A}'$, $\mathcal{N}(u, \mathcal{E}'_j) = 0$ if and only if $\mathcal{N}(u, \mathcal{E}_j) = 0$. Thus

$$\mathcal{A}^\star = \{u \in \mathcal{A}' : \mathcal{N}(u, \mathcal{E}'_j) = 0\}. \tag{14}$$

Moreover, for any pair $(u,v) \in \mathcal{A}'$ with $(u,v) \notin \mathcal{E}'_j$, either $|\mathcal{N}(u, \mathcal{E}'_j)| = 1$ or $|\mathcal{N}(v, \mathcal{E}'_j)| = 1$. For convenience we write $\mathcal{N}(u) := \mathcal{N}(u, \mathcal{E}'_j)$ from now on.

Let $u^\star$ be an element of $\mathcal{A}'$ maximizing $|\mathcal{N}(u)|$, and let $l^\star := |\mathcal{N}(u^\star)|$. Each element $u \in \mathcal{A}'$ is contained in a clique $\mathcal{C}(u)$ of size $d-b$. Since $\mathcal{E}'_j \supset \mathcal{E}_j$, $\mathcal{C}(u)$ is also a clique on the graph with edges $\mathcal{E}'_j$. Let $\mathcal{C}^\star := \mathcal{C}(u^\star) \setminus \{u^\star\}$. Fix $u \in \mathcal{C}^\star$, and suppose $(u,v) \notin \mathcal{E}'_j$ for $v \in \mathcal{A}'$. We claim v cannot be in $\mathcal{N}(u^\star)$. If it were, $\mathcal{N}(v) \geq 2$, in which case $l^\star \geq 2$, meaning $\mathcal{N}(u^\star) \geq 2$. But $(v, u^\star) \notin \mathcal{E}'_j$, which contradicts the construction of $\mathcal{E}'_j$. Moreover, v cannot be in $\mathcal{C}(u^\star)$ since by definition $(u,v) \in \mathcal{E}'_j$ for all $v \in \mathcal{C}(u^\star)$. Hence if $(u,v) \notin \mathcal{E}'_j$, then $v \in \mathcal{D}$ where $\mathcal{D} := \mathcal{A}' \setminus \mathcal{N}(u^\star) \setminus \mathcal{C}(u^\star)$. In particular, if $u \in \mathcal{C}^\star \cap \mathcal{A}' \setminus \mathcal{A}^\star$, then $(u,v) \notin \mathcal{E}'_j$ for some $v \in \mathcal{D}$; i.e. $u \in \mathcal{N}(v)$. Thus

$$\mathcal{A}' \setminus \mathcal{A}^\star \subset (\mathcal{A}' \setminus \mathcal{C}^\star \setminus \mathcal{A}^\star) \cup (\mathcal{A}' \cap \mathcal{C}^\star \setminus \mathcal{A}^\star) \tag{15}$$

$$\subset \{u^\star\} \cup \big(\mathcal{A}' \setminus \mathcal{C}(u^\star)\big) \cup \bigcup_{v \in \mathcal{D}} (\mathcal{N}(v) \cap \mathcal{C}^\star) \tag{16}$$

$$\subset \{u^\star\} \cup \mathcal{N}(u^\star) \cup \mathcal{D} \cup \bigcup_{v \in \mathcal{D}} (\mathcal{N}(v) \cap \mathcal{C}^\star). \tag{17}$$

Hence

$$|\mathcal{A}'| - |\mathcal{A}^\star| \leq 1 + |\mathcal{N}(u^\star)| + |\mathcal{D}| + \sum_{v \in \mathcal{D}} |\mathcal{N}(v)| \tag{18}$$

$$\leq (|\mathcal{D}| + 1)(l^\star + 1) \tag{19}$$

where we have used the fact that $|\mathcal{N}(u)| \leq l^\star$ for all $u \in \mathcal{A}'$. Since $\mathcal{N}(u^\star), \mathcal{C}(u^\star) \subset \mathcal{A}'$ and $\mathcal{N}(u^\star) \cap \mathcal{C}(u^\star) = \emptyset$,

$$|\mathcal{D}| = |\mathcal{A}'| - |\mathcal{N}(u^\star)| - |\mathcal{C}(u^\star)| = |\mathcal{A}'| - l^\star + b - d. \tag{20}$$

Finally we may write

$$|\mathcal{A}^\star| \geq |\mathcal{A}'| - (|\mathcal{D}| + 1)(l^\star + 1) \tag{21}$$

$$= |\mathcal{A}'| - (b - l^\star + |\mathcal{A}'| - d + 1)(l^\star + 1) \tag{22}$$

$$\geq d - (b - l^\star + 1)(l^\star + 1) \tag{23}$$

$$\geq d - (\lfloor \tfrac{b}{2} \rfloor + 1)(\lceil \tfrac{b}{2} \rceil + 1). \tag{24}$$

where (23) follows because $|\mathcal{A}'| \leq d$. This proves $|\mathcal{A}^\star| \geq d - z_b$. The same argument with k replacing d proves that $|\mathcal{A}^\star| \geq k - z_b$ when decoding at a DC.

REFERENCES

[1] A. Dimakis, P. Godfrey, Y. Wu, M. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *Information Theory, IEEE Transactions on*, vol. 56, no. 9, pp. 4539 –4551, Sep. 2010.

[2] A. Dimakis, K. Ramchandran, Y. Wu, and C. Suh, "A survey on network codes for distributed storage," *Proceedings of the IEEE*, vol. 99, no. 3, pp. 476 –489, Mar. 2011.

[3] R. W. Yeung and N. Cai, "Network error correction, part I: Basic concepts and upper bounds," *Comm. in Inf. and Syst.*, vol. 6, no. 1, pp. 19–36, 2006.

[4] N. Cai and R. W. Yeung, "Network error correction, part II: Lower bounds," *Comm. in Inf. and Syst.*, vol. 6, no. 1, pp. 37–54, 2006.

[5] T. Dikaliotis, A. Dimakis, and T. Ho, "Security in distributed storage systems by communicating a logarithmic number of bits," in *Information Theory Proceedings (ISIT), 2010 IEEE International Symposium on*, June 2010, pp. 1948 –1952.

[6] S. Pawar, S. El Rouayheb, and K. Ramchandran, "Securing dynamic distributed storage systems against eavesdropping and adversarial attacks," *Information Theory, IEEE Transactions on*, vol. 57, no. 10, pp. 6734 –6753, Oct. 2011.

[7] F. Oggier and A. Datta, "Byzantine fault tolerance of regenerating codes," in *Peer-to-Peer Computing (P2P), 2011 IEEE International Conference on*, 2011, pp. 112–121.

[8] N. Silberstein, A. Rawat, and S. Vishwanath, "Error resilience in distributed storage via rank-metric codes," in *Communication, Control, and Computing (Allerton), 2012 50th Annual Allerton Conference on*, 2012, pp. 1150–1157.

[9] K. V. Rashmi, N. Shah, K. Ramchandran, and P. Kumar, "Regenerating codes for errors and erasures in distributed storage," in *Information Theory Proceedings (ISIT), 2012 IEEE International Symposium on*, 2012, pp. 1202–1206.

[10] O. Kosut, L. Tong, and D. Tse, "Nonlinear network coding is necessary to combat general Byzantine attacks," in *Proc. Allerton Conference on Communication, Control, and Computing*, Sep. 2009.

[11] ——, "Polytope codes against adversaries in networks," in *Proc. Int. Symp. Information Theory*, 2010.

[12] ——, "Polytope codes against adversaries in networks," *submitted to IEEE Trans. on Information Theory*, dec. 2011, available at `http://arxiv.org/abs/1112.3307`.

[13] S. Kim, T. Ho, M. Effros, and S. Avestimehr, "Network error correction with unequal link capacities," in *Proc. Allerton Conference on Communication, Control, and Computing*, Sep. 2009.

[14] S. Li, R. Yeung, and N. Cai, "Linear network coding," *IEEE Trans. Inf. Theory*, vol. 49, no. 2, pp. 371–381, 2003.