Polytope Codes Against Adversaries in Networks

Oliver Kosut, *Member, IEEE*, Lang Tong, *Fellow, IEEE*, and David N. C. Tse, *Fellow, IEEE*

***Abstract*— This paper investigates a network coding problem wherein an adversary controls a subset of nodes in the network of limited quantity but unknown location. This problem is shown to be more difficult than that of an adversary controlling a given number of edges in the network, in that linear codes are insufficient. To solve the node problem, the class of polytope codes is introduced. Polytope codes are constant composition codes operating over bounded polytopes in integer vector fields. The polytope structure creates additional complexity, but it induces properties on marginal distributions of code vectors so that validities of codewords can be checked by internal nodes of the network. It is shown that polytope codes achieve a cut-set bound for a class of planar networks. It is also shown that this cut-set bound is not always tight, and a tighter bound is given for an example network.**

***Index Terms*— Active adversaries, Byzantine attack, network coding, network error correction, nonlinear codes, polytope codes, security.**

I. Introduction

NETWORK coding allows routers in a network to execute possibly complex codes in addition to routing; it has been shown that allowing them to do so can increase communication rate [1]. However, taking advantage of coding at internal nodes means that sources and destinations must rely on other nodes—nodes they may not have complete control over—to reliably perform certain functions. If these internal nodes do not behave correctly, or, worse, maliciously attempt to subvert the goals of the users—constituting a so-called Byzantine attack [2], [3]—standard network coding techniques fail.

Suppose an omniscient adversary controls an unknown portion of the network, and may arbitrarily corrupt the transmissions on certain communication links. We wish to determine how the size of the adversarial part of the network influences reliable communication rates. If the adversary may control any z unit-capacity edges in the network, then it was shown in [4] and [5] that for the multicast problem (one source sending the same message to many destinations), the capacity reduces by $2z$ compared to the no-adversary problem. To achieve this rate, only linear network coding is needed. Furthermore, if there is just one source and one destination, coding is needed only at the source node; internal nodes need only do routing.

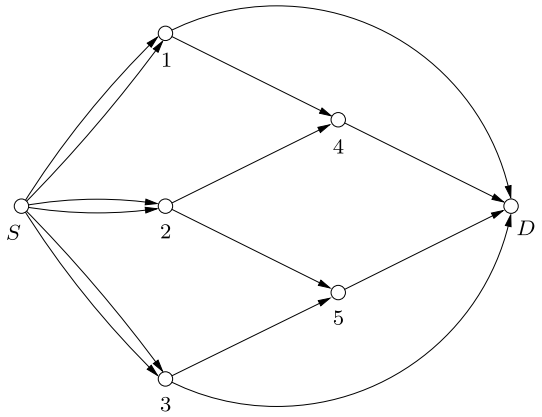

Fig. 1. The Cockroach network. All edges have unit capacity. The network capacity with one traitor node is 2, but no linear code can achieve a rate higher than 4/3 (proved in Appendix A). A capacity-achieving code wherein linear operations are supplemented by nonlinear comparisons is given in Sec. V, and a capacity-achieving Polytope Code is given in Sec. VII.

The above model assumes that any set of z edges may be adversarial, which may not accurately reflect all types of attacks. This model is accurate if the attacker is able to cut transmission lines and change messages that are sent along them. However, if instead the attacker is able to seize a router in a network, it will control the values on all links connected to that router. Depending on which router is attacked, the number of the links controlled by the adversary may vary. In an effort to more accurately model this situation, in this paper we assume that the adversary may control any set of s nodes. We focus on the problem of a single unicast (one source sending a message to one destination). We emphasize that with adversarial nodes, even unicast is an open problem.

Defeating node-based attacks is fundamentally different from defeating edge-based attacks. First, the edge problem does not immediately solve the node problem. Consider, for example, the Cockroach network, shown in Fig. 1. Suppose we wish to handle any single adversarial node in the network (*i.e.* $s = 1$). One simple approach would be to apply the edge-based result from [4], [5]: no node controls more than two unit-capacity edges, so we can defeat the node-based attack by using a code that can handle adversarial control of any two edges (*i.e.* set $z = 2$). However, the no-adversary capacity of this network is 4, so the resulting rate loss of $2z$ leaves us with an achievable rate of 0. As we will show, the actual capacity of the Cockroach network with one traitor node is 2. In effect, relaxing the node attack problem to the edge attack problem is too pessimistic, and we can do better if we treat the node problem differently.

Manuscript received December 14, 2011; revised May 29, 2013; accepted March 4, 2014. Date of publication April 1, 2014; date of current version May 15, 2014. This work was supported in part by the National Science Foundation under Award CCF-0635070 and in part by the Army Research Office under Grant AROW911NF-06-1-0346. This paper was presented at the 2009 Allerton Conference on Communication, Control and Computing, and the 2010 IEEE International Symposium on Information Theory.

O. Kosut is with the School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ 85287 USA (e-mail: okosut@asu.edu).

L. Tong is with the School of Electrical and Computer Engineering, Cornell University, Ithaca, NY 14853 USA (e-mail: ltong@ece.cornell.edu).

D. N. C. Tse is with the Department of Electrical Engineering, Stanford University, Stanford, CA 94305 USA (e-mail: dntse@stanford.edu).

Communicated by M. Motani, Associate Editor for Communication Networks.

Digital Object Identifier 10.1109/TIT.2014.2314642

Node-based attacks and edge-based attacks differ in an even more fundamental way. If the adversary may control any z unit-capacity edges, it is clear that it should always take over edges on the minimum cut of the network. However, if the adversary may control any s nodes, it is not so obvious: it may have to choose between proximity to the min-cut and the number of output edges. For example, in the Cockroach network, node 4 has only one output edge, but it is on the min-cut (which divides nodes $S, 1, 2, 3, 4, 5$ from D); node 1 has two output edges, so apparently more power, but it is one step removed from the min-cut, and therefore its impact may be diminished. This uncertainty about where a network is most vulnerable seems to make the problem hard. Indeed, we find that many standard network coding techniques—in particular, linear codes over a finite field—fail to achieve capacity. We introduce a new class of nonlinear codes called Polytope Codes that can do better.

It this paper (as in [4] and [5]) we always assume an *omniscient adversary*. It was shown in [6] that for the edge-based attack model, achievable rates depend on the eavesdropping capability of the adversary. The omniscient adversary is one that has complete eavesdropping power: it knows the message, it knows the code, and it knows all messages sent in the network. This may be an overly pessimistic model for many cases, but it has the advantage that any strategy formulated against this model is guaranteed to work against a weaker adversary. In addition, for example in wireless settings, one expects an adversary's eavesdropping capability to be greater that its ability to impersonate nodes.

A. Related Work

Byzantine attacks on network coding were first studied in [7], which looked at the problem of detecting adversaries in a random linear coding environment. The z unit-capacity edge adversary problem was solved in [4] and [5]. In [6], the same problem is studied, providing distributed and low complexity coding algorithms to achieve the same asymptotically optimal rates. In addition, [6] looks at two adversary models with limited eavesdropping capability. They show that higher rates can be achieved under these alternate models. In [8], a more general view of the adversary problem is given, whereby the network itself is abstracted into an arbitrary linear transformation.

Network coding under Byzantine attacks that are more general than the simple edge-based model was first studied in [9], a conference version of this work, and [10]. The latter looked at the problem of edge-based attacks when the edges may have unequal capacities. This problem was found to have similar complications to the node-based problem. In particular, both [9] and [10] found that linear coding is suboptimal, and that simple nonlinear operations used to augment a linear code can improve throughput. Indeed, [10] used a network almost identical to what we call the Cockroach network to demonstrate that nonlinear operations are necessary for the unequal edge problem. We show in Sec. XI that the unequal-capacity edge problem is subsumed by the node problem.

These works seek to correct for the adversarial errors at the destination. An alternative strategy known as the watchdog, studied for wireless network coding in [11], is for nodes to police downstream nodes by overhearing their messages to detect modifications. In [12], a similar approach is taken, and they found that nonlinear operations similar to ours can be helpful, in which comparisons are made to detect errors.

B. Main Results

Many achievability results in network coding have been proved using linear codes over a finite field. In this paper we demonstrate that linear codes are insufficient for this problem. Moreover, we develop a class of codes called Polytope Codes, originally introduced in [9] under the less descriptive term "bounded-linear codes". Polytope Codes are used to prove that a cut-set bound, stated and proved in Sec. III, is tight for a certain class of networks. Polytope Codes differ from linear codes in three ways:

1) *Comparisons:* A significant tool we use to defeat the adversary is that internal nodes in the network perform comparisons: they check whether their received data could have occurred if all nodes had been honest. If not, then there must be an upstream traitor that altered one of the received values, in which case this traitor can be localized. The result of the comparison, a bit signifying whether or not it succeeded, can be transmitted downstream through the network. The destination receives these comparison bits and uses them to determine who may be the traitors, and how to decode. These comparison operations are nonlinear, and, as we will demonstrate in Sec. V, incorporating them into a standard finite-field linear code can increase achieved rate. However, even standard linear codes supplemented by these nonlinear comparison operations appears to be insufficient to achieve capacity in general. Polytope Codes also incorporate comparisons, but of a more sophisticated variety.
2) *Constant Composition Codebooks:* Unlike usual linear network codes, Polytope Codes are essentially constant composition codes [13]. In particular, each Polytope Code is governed by a joint probability distribution on a set of random variables, one for each edge in the network. The codebook is composed of the set of all sequences with joint type exactly equal to this distribution. The advantage of this method of code construction is that an internal node knows exactly what joint type to expect of its received sequences, because it knows the original distribution. In a Polytope Code, comparisons performed inside the network consist of checking whether the observed joint type matches the expected distribution. If it does not, then the adversary must have influenced one of the received sequences, so it can be localized.
3) *Distributions Over Polytopes:* The final difference between linear codes and Polytope Codes—and the one for which the latter are named—comes from the nature of the probability distributions that form the basis of the code, as described above. They are uniform distributions over the set of integer lattice points on polytopes in real vector fields. This choice for distribution provides

two useful properties. First, the entropy function for these distributions can be easily calculated merely from properties of the linear space in which the polytope sits. For this reason, they share characteristics with finite-field linear codes. In fact, a linear code can typically be converted into a Polytope Code achieving the same rate.[1] The second useful property has to do with how the comparisons inside the network are used. These distributions are such that if enough comparisons succeed, the adversary is forced to act as an honest node and transmit correct information. We consider this to be the fundamental property of Polytope Codes. It will be elaborated in examples in Sec. VI and Sec. VII, and then stated in its most general form as Theorem 4 in Sec. VIII.

We formulate the problem in Sec. II. After stating and proving a cut-set bound in Sec. III, in Sec. IV we state an achievability bound that is proved using Polytope Codes. This bound is used to show that the cut-set bound can be achieved for a class of planar networks. Planarity requires that the graph can be embedded in a plane such that intersections between edges occur only at nodes. This ensures that enough opportunities for comparisons are available, allowing the code to defeat the adversary. Achievability is proved in Sec. IX, but first we develop the theory of Polytope Codes through several examples in Sec. V–VII. In addition, we study the relationship between linear codes and Polytope Codes in Sec. X, and in Sec. XI we prove an upper bound that is strictly tighter than the cut-set bound for an example network. We conclude in Section XII. The appendices contain several auxiliary results and proofs.

II. Problem Formulation

Let (V, E) be an directed acyclic graph. We assume all edges have unit capacity, and there may be more than one edge connecting the same pair of nodes. We consider a single unicast: one node in V is denoted S, the source, and one is denoted D, the destination. We wish to determine the maximum achievable communication rate from S to D when any set of s nodes in $V\backslash\{S, D\}$ are *traitors*; *i.e.* they are controlled by the adversary. Given a rate R and a blocklength n, the message w is chosen at random from the set $\{1, \ldots, 2^{nR}\}$. Each edge $e \in E$ holds a value $X_e \in \{1, \ldots, 2^n\}$.

A *code* is made up of three components:

1) an encoding function at the source, which takes the message as input and produces values to place on all output edges,
2) a coding function at each internal node $i \in V\backslash\{S, D\}$, which takes the values on all input edges to i, and produces values to place on all output edges from i,
3) and a decoding function at the destination, which takes the values on all input edges and produces an estimate $\hat{w}$ of the message.

Suppose $T \subseteq V \setminus \{S, D\}$ is the set of traitors, with $|T| = s$. They may subvert the coding functions at nodes $i \in T$ by placing arbitrary values on all the output edges from these nodes. Let Z_T be the set of values on these edges. For a particular code, specifying the message w as well as Z_T determines exactly the values on all edges in the network, in addition to the destination's estimate $\hat{w}$. A rate R is *achievable* if there exists a code operating at that rate with some blocklength n such that for all messages, all sets of traitors T, and all values of Z_T, $w = \hat{w}$. That is, the destination always decodes correctly no matter what the adversary does. Let the *capacity* C be the supremum of all achievable rates.

Notation: For a positive integer a, we write $[a] := \{1, 2, \ldots, a\}$. For an edge $e \in E$, with $e = (i, j)$, where $i, j \in V$, let $\text{head}(e) = j$ and $\text{tail}(e) = i$. For a node $i \in V$, let $\mathcal{E}_{\text{in}}(i)$ be the set of edges e with $\text{head}(e) = i$, and let $\mathcal{E}_{\text{out}}(i)$ be the set of edges e with $\text{tail}(e) = i$. Let $\mathcal{N}_{\text{in}}(i)$ be the set of input neighbors of i; that is, the set of $\text{tail}(e)$ for each $e \in \mathcal{E}_{\text{in}}(i)$. Similarly, let $\mathcal{N}_{\text{out}}(i)$ be the set of output neighbors of i. For integers a, b, let $\mathcal{V}_{a,b}$ be the set of nodes with a inputs and b outputs. We will refer to such nodes as *a-to-b nodes*. For $l \in \{1, 2\}$, let $\bar{l} = 2 - l$. A *path* $\mathbf{p}$ is defined as an ordered list alternating between nodes and edges $\mathbf{p} = (v_1, e_1, v_2, e_2, \ldots, v_k, e_L, v_{L+1})$ where $v_l \in V$ and $e_l \in E$, and where $\text{tail}(e_l) = v_l$ and $\text{head}(e_l) = v_{l+1}$ for $l \in [L]$. The tail and head of a path are defined as v_1 and v_{L+1} respectively. For an edge e, we write $e \in \mathbf{p}$ if e is in the list defining $\mathbf{p}$, and for a node v we define $v \in \mathbf{p}$ similarly. A node i is said to *reach* a node j if there exists a path with tail i and head j. By convention, a node can reach itself.

III. Cut-Set Upper Bound

It is shown in [4] and [5] that if an adversary controls z unit-capacity edges, the network coding capacity reduces by $2z$. This is a special case of a more general principle: an adversary-controlled part of the network does as much damage in rate as if a part of network twice as large were merely removed. This doubling effect is for the same reason that, in a classical error correction code, the Hamming distance between codewords must be at least twice the number of errors to be corrected; this is the Singleton bound [14]. Theorem 1 is a cut-set upper bound that generalizes the Singleton bound and its analogue in [4] and [5] to the node-based adversary problem. We state and prove the bound after the following definitions.

We define a *cut* in a network as a subset of nodes $A \subset V$ containing the source but not the destination. Given a cut A, a *forward edge* is an edge $(i, j) \in E$ with $i \notin A$ and $j \in A$. A *backward edge* is an edge $(i, j) \in E$ with $i \notin A$ and $j \in A$.

Theorem 1: Fix a cut A with no backward edges, and a set $U \subset V \setminus \{S, D\}$ with $|U| = 2s$. (Recall s is the number of traitor nodes.) The capacity is upper bounded by

$$C \leq |\{(i, j) \in E : i \in A \setminus U, j \notin A\}|. \tag{1}$$

Proof: Divide U into two sets T_1 and T_2 with $|T_1| = |T_2| = s$. Let E_1 and E_2 be the sets of forward edges for the cut A out of nodes in T_1 and T_2 respectively. Let $\bar{E}$ be the set of all forward edges not out of nodes in T_1 or T_2. Observe that the upper bound in (1) is precisely the total capacity of all edges in $\bar{E}$. Note also that, since there are no backward edges for the cut A, the values on edges in $\bar{E}$

[1] There would be no reason to do this in practice, since Polytope Codes require much longer blocklengths. See Sec. X for more details.

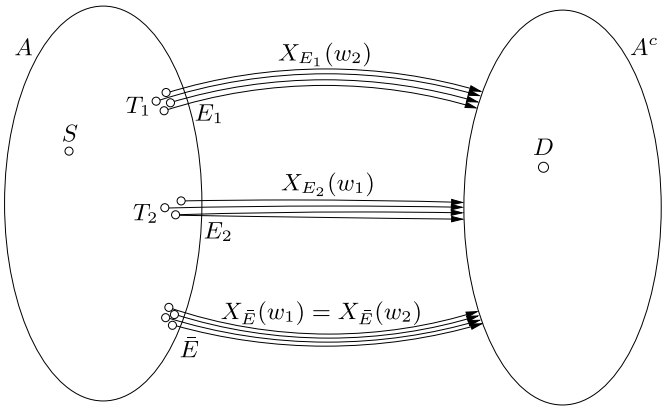

Fig. 2. Diagram of the proof of Theorem 1. The values on the links crossing the cut are such that it is impossible to determine whether T_1 or T_2 is the true set of traitors, and which of w_1 or w_2 is the true message.

are not influenced by the values on edges in E_1 or E_2. This setup is diagrammed in Fig. 2.

Suppose (1) does not hold. Therefore there exists a code with blocklength n achieving a rate R higher than the right hand side of (1). For any set of edges $F \subseteq E$, for this code, we can define a function

$$X_F : [2^{nR}] \to \prod_{e \in F} [2^n] \tag{2}$$

such that for a message w, assuming all nodes act honestly, the values on edges in F is given by $X_F(w)$. Since R is greater than the total capacity for all edges in $\bar{E}$, there exists two messages w_1 and w_2 such that $X_{\bar{E}}(w_1) = X_{\bar{E}}(w_2)$.

We demonstrate that it is possible for the adversary to confuse the message w_1 with w_2. Suppose w_1 were the true message, and the traitors are T_1. The traitors replace the values sent along edges in E_1 with $X_{E_1}(w_2)$. If there are edges out of nodes in T_1 that are not in E_1—*i.e.* they do not cross the cut—the traitors do not alter the values on these edges. Thus, the values sent along edges in $\bar{E}$ are given by $X_{\bar{E}}(w_1)$. Now suppose w_2 were the true message, and the traitors are T_2. They replace the messages going along edges in E_2 with $X_{E_2}(w_1)$, again leaving all other edges alone. Note that in both these cases, the values on E_1 are $X_{E_1}(w_2)$, the values on E_2 are $X_{E_2}(w_1)$, and the values on $\bar{E}$ are $X_{\bar{E}}(w_1)$. This comprises all edges crossing the cut, so the destination receives the same values under each case; therefore it cannot distinguish w_1 from w_2. ■

1) Example Use of Theorem 1: We illustrate the use of Theorem 1 on the Cockroach network, as shown in Fig. 1, with a single adversary node (*i.e.* $s = 1$). To apply the bound, we choose a cut A and a set U with $|U| = 2s = 2$. Take $A = \{S, 1, 2, 3, 4, 5\}$, and $U = \{1, 4\}$. There are no backward edges, and four forward edges, but the only forward edges not out of nodes U are $(3, D)$ and $(5, D)$. Hence, by Theorem 1, the capacity is at most 2. Alternatively, we could take $A = \{S, 1, 2, 3\}$ and $U = \{1, 2\}$, again yielding an upper bound of 2. It is not hard to see that 2 is the smallest upper bound given by Theorem 1 for the capacity of the Cockroach network. In fact, rate 2 is achievable, as will be shown in Sec. V using a linear code supplemented by comparison operations, and again in Sec. VII using a Polytope Code.

2) Discussion of Theorem 1: In traditional network coding, without adversaries, backward edges are not relevant to the cut-set bound; the bound is simply the sum of the capacities of all forward edges. As illustrated in Theorem 1, backward edges matter in the adversarial problem. This is because the bound requires that forward edges controlled by the adversary do not influence forward edges not controlled by the adversary. (In the notation of the proof of Theorem 1, E_1 and E_2 do not influence $\bar{E}$.) This is not guaranteed in the presence of a backward edge. In Appendix B, we give an example network where the right-hand side of (1) is strictly smaller than capacity, due to a backward edge. Theorem 1 resolves the backward edge problem by restricting to cuts without backward edges. Theorem 1 will be strong enough to find the capacity of the class of planar networks to be specified in Sec. IV, but for the general problem it can be tightened. We state and prove a tighter version of the cut-set bound in Appendix C. Unlike the problem without adversaries, there appears to be no straightforward notion of a cut-set bound. Some even more elaborate bounds are found in [10] and [15]. These papers study the unequal-edge problem, but the bounds can be readily applied to the node problem.

It was originally conjectured in [10] that even the best cut-set bound is not tight in general. In Sec. XI, we demonstrate that there can be an active upper bound on capacity fundamentally unlike a cut-set bound. The example used to demonstrate this, though it is a node adversary problem, can be easily modified to confirm the conjecture stated in [10].

IV. Achievability Bounds

We define the following properties a network (V, E) may satisfy:

2-OUT: No node other than the source has more than two output edges.

IN-OUT: No node other than the source has more output edges than input edges.

PLANAR: The graph defined by (V, E) is planar.

Almost all networks we study in this paper satisfy 2-OUT and IN-OUT. The following theorem gives the capacity for networks also satisfying PLANAR with one traitor. It is a consequence of Theorem 3 and Lemma 1, stated below.

Theorem 2: Let $s = 1$, and let (V, E) be a network satisfying 2-OUT, IN-OUT, and PLANAR. The capacity is equal to the cut-set bound given in Theorem 1.

Perhaps the most interesting condition in the statement of Theorem 2 is PLANAR. Recall that a graph is said to be *embedded* in a surface (typically a two-dimensional manifold) when it is drawn in this surface so that edges intersect only at nodes. A graph is *planar* if it can be embedded in the plane. However, we will prove a stronger achievability bound than Theorem 2 that also applies to non-planar graphs, based on an additional network property that we define in several steps as follows.

Let M be the number of edges connected to the destination (*i.e.* $M = |\mathcal{E}_{\text{in}}(D)|$). Note that for any network satisfying

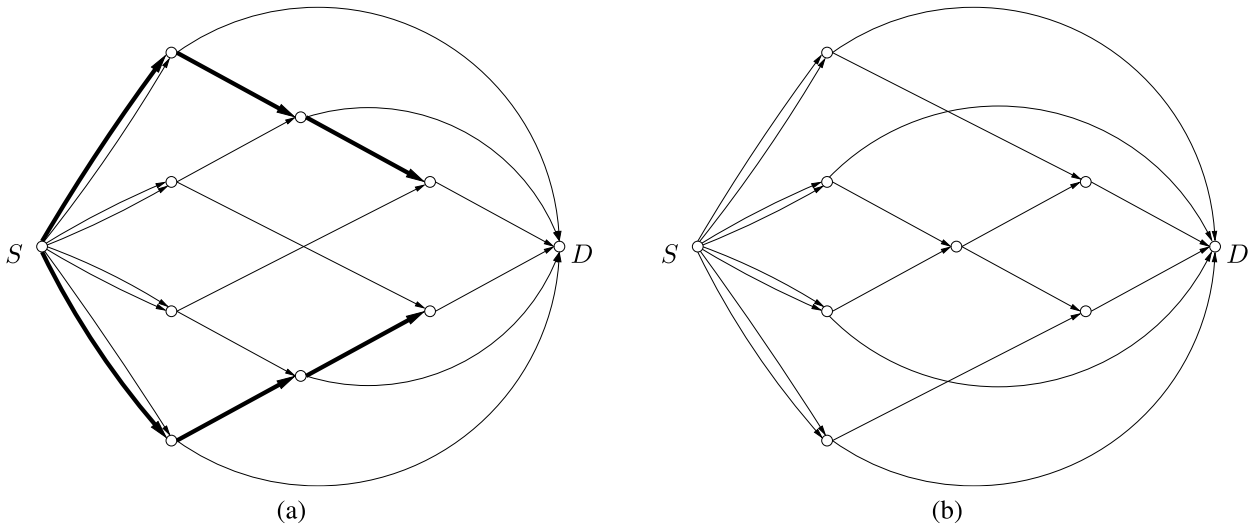

Fig. 3. Examples of non-planar networks. Subplot (a) shows a network satisfying PATHS but not PLANAR. The two paths $\mathbf{p}_v$ are indicated by bold edges. Subplot (b) shows a network satisfying neither PATHS nor PLANAR. Both networks have $M = 6$ and cut-set bound $M - 2 = 4$. By Theorem 3, rate 4 is achievable for network (a), but the theorem only asserts that rate 3 is achievable for network (b). In fact, the capacity of network (b) is unknown.

IN-OUT, no node has more output edges than input edges, so the min-cut is that between the destination and the rest of the network. Hence, the value of the min-cut is M. For each node v with exactly one more input than output (*i.e.* a 2-to-1 node or a 3-to-2 node), let Γ_v be the set of nodes i such that v is the only node reachable from i with more inputs than outputs. Define the following condition on a network (V, E):

PATHS: For each a-to-b node with $a > b$, there exists $b - a$ paths $\mathbf{p}_{v,i}$ for $i \in [a - b]$ such that $\text{tail}(\mathbf{p}_{v,i}) = S$, $\text{head}(\mathbf{p}_{v,i}) = v$, all paths are edge-independent, and

$$\mathcal{N}_{\text{in}}(D) \cup \bigcup_{\substack{a>0 \\ v\in\mathcal{V}_{a+1,a}}} \Gamma_v \subset \bigcup_{\substack{a,b\in\mathbb{Z} \\ v\in\mathcal{V}_{a,b} \\ i\in[b-a]}} \mathbf{p}_{v,i}. \tag{3}$$

For a-to-b nodes v with $a - b = 1$, we adopt the simplified notation $\mathbf{p}_v := \mathbf{p}_{v,1}$. The Cockroach network can be easily seen to satisfy PATHS: The only nodes in the Cockroach network with more inputs than outputs are the 2-to-1 nodes 4 and 5. Moreover $\Gamma_4 = \{1, 4\}$, $\Gamma_5 = \{3, 5\}$, and $\mathcal{N}_{\text{in}}(D) = \{1, 3, 4, 5\}$. We may satisfy (3) by setting

$$\mathbf{p}_4 := \big(S, (S, 1), 1, (1, 4), 4\big) \tag{4}$$

$$\mathbf{p}_5 := \big(S, (S, 3), 3, (3, 5), 5\big). \tag{5}$$

Fig. 3 illustrates PATHS for two non-planar graphs. Fig. 3(a) shows a network satisfying PATHS but not PLANAR, and Fig. 3(b) shows a network satisfying neither PATHS nor PLANAR. As we show in Lemma 1, stated below, the condition PATHS is a more general one than PLANAR. The usefulness of PATHS will become clear after we develop the theory of Polytope Codes in Sec. V–VIII.

The following theorem is our most general achievability bound. It is proved in Sec. IX.

Theorem 3: Let $s = 1$, and let the network (V, E) satisfy 2-OUT and IN-OUT.

1) The capacity satisfies $C \geq M - 4$.
2) If the cut-set bound given by Theorem 1 is at least $M-3$, then $C \geq M - 3$.
3) If (V, E) satisfies PATHS, then $C = M - 2$.
4) The cut-set bound is not greater than $M - 2$.

Note that if the cut-set bound is no more than $M - 3$, then Theorem 3 already establishes Theorem 2. The following lemma, proved in Appendix D, completes the proof of Theorem 2.

Lemma 1: If a network (V, E) satisfies 2-OUT, IN-OUT, and PLANAR, and the cut-set bound with $s = 1$ is $M - 2$, then the network also satisfies PATHS.

Polytope Codes are used to prove achievability for Theorem 3 (and hence Theorem 2). Before proving the theorem, we develop Polytope Codes by means of several examples in Sec. V–VII, and then we describe their general properties of in Sec. VIII.

V. A Linear Code With Comparisons for the Cockroach Network

The Cockroach network is easily seen to satisfy 2-OUT and IN-OUT. Fig 1 shows a plane embedding; hence it satisfies PLANAR. Paths $\mathbf{p}_v$ satisfying (3) were given in (4)–(5); hence it also satisfies PATHS. Therefore, since the smallest cut-set bound given by Theorem 1 for a single traitor node is 2, Theorem 2 (or Theorem 3) asserts that the capacity of the Cockroach network is 2. In this section, we present a capacity-achieving code for the Cockroach network that is composed of a linear code over a finite-field supplemented by nonlinear comparisons. This illustrates the usefulness of comparisons in defeating adversaries network coding. Before doing so, we provide an intuitive argument that linear codes are insufficient. A more technical proof that the linear capacity is 4/3 is given in Appendix A.

Is it possible to construct a linear code achieving rate 2 for the Cockroach network? We know from the

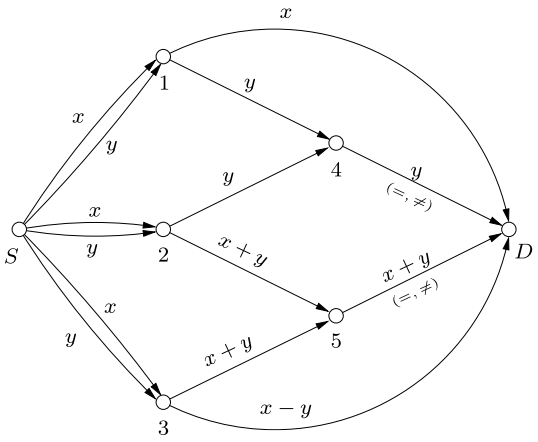

Fig. 4. A nonlinear code for the Cockroach Network achieving the capacity of 2.

argument underlying the Singleton bound—and the proof of Theorem 1—that, in order to defeat a single traitor node, if we take out everything controlled by two nodes, the destination must be able to decode from whatever remains. Suppose we take out nodes 2 and 3. These nodes certainly control the values on $(5, D)$ and $(3, D)$, so if we hope to achieve rate 2, the values on $(1, D)$ and $(4, D)$ must be incorruptible by nodes 2 and 3. Edge $(1, D)$ is not a problem, but consider $(4, D)$. With a linear code, the value on this edge is a linear combination of the values on $(1, 4)$ and $(2, 4)$. In order to keep the value on $(4, D)$ incorruptible by node 2, the coefficient used to construct the value on $(4, D)$ from $(2, 4)$ must be zero. In other words, node 4 must ignore the value on $(2, 4)$ when constructing the value it sends on $(4, D)$. If this is the case, we lose nothing by removing $(2, 4)$ from the network. However, without this edge, we may apply Theorem 1 with $A = \{S, 1, 2, 3\}$ and $U = \{1, 3\}$ to conclude that the capacity is no more than 1. Therefore no linear code can successfully achieve rate 2.

This argument does not rigorously show that the linear capacity is less than 2, because it shows only that a linear code cannot achieve exactly rate 2, but it does not bound the achievable rate with a linear code away from 2. However, it is meant to be an intuitive explanation for the limitations of linear codes for this problem, as compared with the successful nonlinear codes that we will subsequently present. The complete proof that the linear capacity is 4/3 is given in Appendix A.

We now introduce a nonlinear code to achieve the capacity of 2. We work in the finite field of p elements. Let the message w be a $2k$-length vector split into two k-length vectors x and y. We will use a blocklength large enough to place one of $2p^k$ values on each link. In particular, this is enough so that a link can hold some linear combination of x and y, as well as one additional bit. For large enough k, this extra bit becomes insignificant, so the code achieves rate 2.

The scheme is shown in Figure 4. Node 4 receives the vector y from both nodes 1 and 2. It forwards one of these copies to D (it does not matter which). In addition, it performs a nonlinear comparison between the two received copies of y, resulting in a bit comprised of one of the special symbols $=$ or $\neq$. If the two received copies of y agree, it sends $=$, otherwise it sends $\neq$. The link $(4, D)$ can accommodate this, since it may have up to $2p^k$ messages placed on it. Node 5 does the same with its two copies of the vector $x + y$.

The destination's decoding strategy depends on the two comparison bits sent from nodes 4 and 5, as follows:

- If node 5 sends $\neq$ but node 4 sends $=$, then the traitor must be one of nodes 1, 2, or 4. In any case, the vector $x - y$ received from node 3 is certainly trustworthy. Moreover, $x + y$ can be trusted, because even if node 2 is the traitor, its transmission must have matched whatever was sent by node 3; if not, node 5 would have transmitted $\neq$. Therefore the destination can trust both $x + y$ and $x - y$, from which it can decode the message $w = (x, y)$.
- If node 5 sends $\neq$ but node 4 sends $=$, then we are in the symmetric situation and can reliably decode w from x and y.
- If both nodes 4 and 5 send $\neq$, then the traitor must be node 2, in which case the destination can reliable decode from x and $x - y$.
- If both messages are $=$, then the destination cannot eliminate any node as a possible traitor. However, we claim that at most one of $x, y, x + y, x - y$ can have been corrupted by the traitor. If node 1 is the traitor, it may choose whatever it wants for x, and the destination would never know. However, node 1 cannot impact the value of y without inducing a $\neq$, because its transmission to node 4 is verified against that from node 2. Similarly, node 3 controls $x - y$ but not $x + y$. Nodes 4 and 5 control only y and $x + y$ respectively. Node 2 controls nothing, because both y and $x + y$ are checked against other transmissions. Therefore, if the destination can find three of $x, y, x + y, x - y$ that all agree on the message w, then this message must be the truth because only one of the four could be corrupted, and w can be decoded from the other two. Conversely, there must be a group of three of $x, y, x + y, x + 2y$ that agree, because at most one has been corrupted. Hence, the destination can always decode w.

Even though our general proof of Theorem 2 uses a Polytope Code, which differs significantly from this one, the manner in which the comparisons comes into play is essentially the same. The key insight is to consider the code from the perspective of the traitor. Suppose it is node 1, and consider the choice of what value for y to send along edge $(1, 4)$. If it sends a false value for y, then the comparison at node 4 will fail, which will lead the destination to suspect the upper part of the network, and thereby ignore all values influenced by node 1. The only other choice for node 1 is to cause the comparison at node 4 to succeed; but this requires sending the true value of y, which means it has no hope to corrupt the decoding process. This is the general principle that makes our codes work: force the traitor to choose between acting honestly, or acting otherwise and thereby giving away its position.

We make one further note on this code, regarding why the specific approach used here for the Cockroach network fails on the more general problem. Recall that in order to make an effective comparison, the values sent along edges

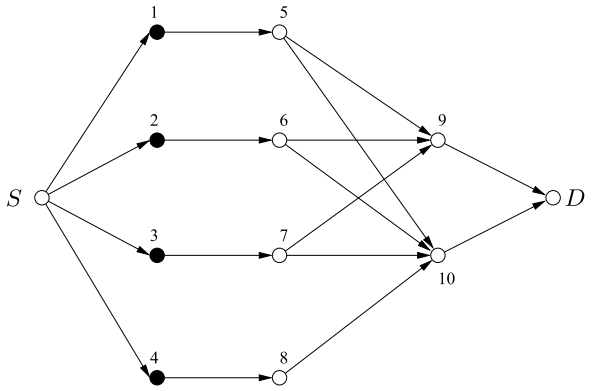

Fig. 5. The Caterpillar Network. All edges have unit capacity. One node may be a traitor, but only one of the black nodes: nodes 1–4.

$(1, 4)$ and $(2, 4)$ needed to be exactly the same. If they had been independent vectors, no comparison could be useful. This highly constrains the construction of the code. While it succeeds for the Cockroach network, it fails for many other examples, such as the Caterpillar network to be introduced in the next section. The advantage of the Polytope Code is that it relaxes the constraint on the values that must be available in order to form a useful comparison; in fact, it becomes possible to have useful comparisons between nearly independent variables, which is not possible with a code built on a finite-field.

VI. An Example Polytope Code: The Caterpillar Network

The Caterpillar Network is shown in Figure 5. We consider a slightly different version of the node-based attack on this network: at most one node may be a traitor, but only nodes 1–4. This network does not satisfy the conditions of Theorem 2 or 3, but we introduce it in order to motivate the Polytope Code.

Even though this problem differs from the one defined earlier in that not every node in the network may be a traitor, it is easy to see that we may still apply the cut-set bound of Theorem 1 as long as we take the set U to be a subset of the allowable traitors. If we apply Theorem 1 with $A = \{S, 1, 2, 3, 4\}$ and $U = \{1, 2\}$, we find that the capacity of this network is no more than 2. As we will show, the capacity is 2. Before proving this, we pause to develop intuition for the problem.

A. Properties of Capacity-Achieving Codes

Suppose we are given a code that achieves rate 2. What properties must it have? The following analysis is not rigorous, but we provide it to build intuition for the Polytope Code solution. Of the four values on the edges $(1, 5)$, $(2, 6)$, $(3, 7)$, and $(4, 8)$, one may be corrupted by the adversary. This means that in any capacity-achieving code, these four values form a $(4, 2)$ MDS code. That is, given any uncorrupted pair of these four values, it is possible to decode the message exactly. Since each edge has capacity 1, in order to achieve rate 2, the values on each pair of edges are independent. An example code satisfying this property is to take the massage to be composed of two elements x, y of a finite field, and transmit on these four edges $x, y, x + y, x - y$. However, as we will show, this choice does not succeed.

For any code, we may assume without loss of generality that nodes 5–8 forward whatever they receive on their incoming edges to all their outgoing edges. Any code that does not have this property can be converted into one that does without changing the achieved rate. For example, suppose node 5 transmits something different along $(5, 9)$ than it received along $(1, 5)$. Let $x_{(5,9)}$ and $x_{(1,5)}$ be these two values respectively. There is some function f such that $x_{(5,9)} = f(x_{(1,5)})$. We construct a new code where instead $x_{(5,9)} = x_{(1,5)}$, and then node 9 computes $f(x_{(5,9)})$ and proceeds as in the original code with this value in place of $x_{(5,9)}$. All other values are unaffected, so the rate does not change. The same argument can be used for all edges out of nodes 5–8.[2]

Now consider the two edges $(9, D)$ and $(10, D)$. As these are the only edges incident to the destination, to achieve rate 2, both must hold values guaranteed to be uncorrupted by the traitor. Consider, in particular, the operation at node 9. Since, as argued above, nodes 5–8 forward whatever they receive, node 9 receives three of the four values sent from nodes 1–4. That is, in a capacity-achieving code, node 9 receives the values sent on edges $(1, 5)$, $(2, 6)$, $(3, 7)$ (one of which may be corrupted) and constructs a trustworthy value to send on $(9, D)$.

Given the properties derived above for any code achieving rate 2, how can we construct such a code? The key problem is to design values on edges $(1, 5)$, $(2, 6)$, $(3, 7)$ that are pairwise independent, but such that if one of them is corrupted, one can construct a trustworthy value to transmit on $(9, D)$. The finite field example mentioned above would not work: Suppose node 9 receives values for $x, y, x + y$, one of which may be corrupted by the traitor. If the linear constraint among these three values does not hold—that is, if the received value for $x + y$ does not match the sum of the value for x and the value for y—then any of the three values may be the incorrect one. Therefore, from node 9's perspective, any of nodes 1, 2, or 3 could be the traitor. One way to produce a trustworthy symbol is to correctly conclude that one of these three nodes is not the traitor. If, for example, it could determine that the traitor was node 1 or node 2 but not node 3, then the value sent along $(3, 7)$ could be forwarded to $(9, D)$ with a guarantee of correctness. A Polytope Code allows for precisely this.

B. Coding Strategy

We now begin to describe a capacity-achieving Polytope Code for the Caterpillar network. We do so first by describing how the code is built out of a probability distribution, and the properties we need this probability distribution to satisfy. Subsequently, we give an explicit construction for a probability

[2]This argument—and, indeed, much of the paper—relies on the fact that the adversary is omniscient. If instead the adversary had limited eavesdropping capability, as in [16], then there might be reason to hide data sent along some edges even if not limited by link capacity. Because we assume an omniscient adversary, this is not a concern.

distribution derived from a polytope, and show that it has the desired properties.

Let X, Y, Z, W be jointly distributed random variables, each defined over the finite alphabet $\mathcal{X}$. Assume all probabilities on these random variables are rational. Let $T^n(XYZW) \in \mathcal{X}^{4n}$ be the set of joint sequences $(x^n y^n z^n w^n)$ with joint type equal to the distribution on X, Y, Z, W. Our coding strategy will be to associate each element of $T^n(XYZW)$ with a distinct message. Given the message, we find the associated four sequences x^n, y^n, z^n, w^n, and transmit them on the four edges out of nodes 1, 2, 3, 4 respectively.

Typically used in a channel coding context, a *constant composition code* [13] is one in which every codeword has the same type. We are using a similar strategy here, in that $(x^n y^n z^n w^n)$ always has the same type. Our reason for using a constant composition code is that it can be used to detect the traitor; that is, if the observed type differs from the anticipated type, then the traitor must have touched one of the sequences.

We now calculate the rate achieved by the code described above. For n such that $T^n(XYZW)$ is not empty, a basic result from the theory of types states that

$$|T^n(XYZW)| \geq \frac{1}{(n+1)^{|\mathcal{X}|^4}} 2^{nH(XYZW)}. \tag{6}$$

Each edge holds a sequence in $\mathcal{X}^n$. Since the number of messages is precisely the cardinality of $T^n(XYZW)$, the rate is

$$\frac{\log|T^n(XYZW)|}{n\log|\mathcal{X}|} \geq \frac{H(XYZW)}{\log|\mathcal{X}|} - \frac{|\mathcal{X}|^4\log(n+1)}{n\log|\mathcal{X}|}. \tag{7}$$

In particular, for sufficiently large n, we may operate at a rate arbitrarily close to $\frac{H(XYZW)}{\log|\mathcal{X}|}$. Therefore, to achieve rate 2, we would like the following property to hold.

Property 1: $\frac{H(XYZW)}{\log|\mathcal{X}|} = 2$.

The adversary may alter the value of one of the sequences sent out of nodes 1–4. By the Singleton bound argument, it must be possible to reconstruct the message from any two of these four sequences. We therefore need the following property.

Property 2: Any two of X, Y, Z, W determine the other two.

For reasons that will become clear, we also need the following property. It is an example of the fundamental property of Polytope Codes.

Property 3: Any random variables $\tilde{X}, \tilde{Y}, \tilde{Z}$ satisfying the three conditions

$$(\tilde{X}, \tilde{Y}) \sim (X, Y) \tag{8}$$
$$(\tilde{X}, \tilde{Z}) \sim (X, Z) \tag{9}$$
$$(\tilde{Y}, \tilde{Z}) \sim (Y, Z) \tag{10}$$

also satisfy

$$(\tilde{X}, \tilde{Y}, \tilde{Z}) \sim (X, Y, Z). \tag{11}$$

Suppose we are given random variables X, Y, Z, W satisfying Properties 1–3. We now describe what nodes 9 and 10 transmit to the destination. Let $\tilde{x}^n, \tilde{y}^n, \tilde{z}^n, \tilde{w}^n$ be the four sequences that are sent on the edges out of nodes 1–4; because of the traitor at most one of these may differ from x^n, y^n, z^n, w^n. Let random variables $\tilde{X}, \tilde{Y}, \tilde{Z}, \tilde{W}$ have joint distribution equal to the joint type of $(\tilde{x}^n, \tilde{y}^n, \tilde{z}^n, \tilde{w}^n)$. This is a formal definition; these variables do not exist *per se* in the network, but defining them makes it convenient to describe the behavior of the code. Since node 9 receives $\tilde{x}^n, \tilde{y}^n, \tilde{z}^n$, it knows exactly the joint distribution of $\tilde{X}, \tilde{Y}, \tilde{Z}$. In particular, it can check which of (8)–(11) are satisfied for these variables.

Suppose (11) holds. Then all three sequences $\tilde{x}^n, \tilde{y}^n, \tilde{z}^n$ are trustworthy, because if a traitor is among nodes 1–3, it must have transmitted the true value of its output sequence, or else the empirical type would not match, due to Property 2. In this case, node 9 forwards $\tilde{x}^n$ to the destination, confident that it is correct. Meanwhile, node 10 can also observe $\tilde{X}, \tilde{Y}, \tilde{Z}$, and so it forwards $\tilde{y}^n$ to the destination.

Now suppose (11) does not hold. Then by Property 3, one of (8)–(10) must not hold. Suppose, for example, that $(\tilde{X}, \tilde{Y}) \not\sim (X, Y)$. Because of our constant composition code construction, this can only occur if either node 1 or 2 is the traitor. Hence, node 3 is honest, so Node 9 forwards $\tilde{z}^n$ to the destination without error. Similarly, no matter which pairwise distribution does not match, node 9 always forwards the sequence not involved in the mismatch. Meanwhile, node 10 forwards $\tilde{w}^n$ to the destination, since in any case the traitor has been localized to nodes 1–3. The destination always receives two of the four sequences, both correct; therefore it may decode.

C. The Polytope Distribution

All that remains to prove that rate 2 can be achieved for the Caterpillar network is to show that there exists variables X, Y, Z, W such that Properties 1–3 hold. In fact, this is not quite possible. In particular, Property 1 implies that X, Y, Z, W are pairwise independent. If so, Property 3 cannot hold, because we can take $\tilde{X}, \tilde{Y}, \tilde{Z}$ to be jointly independent with $\tilde{X} \sim X$, $\tilde{Y} \sim Y$, and $\tilde{Z} \sim Z$. This satisfies (8)–(10) but not (11). We therefore replace Property 1 with the following slight relaxation.

Property 4: $\frac{H(XYZW)}{\log|\mathcal{X}|} \geq 2 - \epsilon$.

If for every $\epsilon > 0$, there exists a set of random variables satisfying Properties 2–4, then by (7) we achieve rate 2.

The most unusual aspect of the Polytope Code is Property 3 and its generalization, to be stated as Theorem 4 in Sec. VIII. Therefore, before constructing a distribution satisfying all three properties, we illustrate in Table I a very simple distribution on three binary variables variables that satisfies just Property 3. This distribution is only on X, Y, Z; for simplicity we leave out W, as it is not involved in Property 3. We encourage the reader to manually verify Property 3 for this distribution. Observe that X, Y, Z as given in Table I may be alternatively expressed as being uniformly distributed on the following set:

$$\{x, y, z \in \{0, 1\} : x + y + z = 1\}. \tag{12}$$

TABLE I

A SIMPLE DISTRIBUTION ON THREE BINARY RANDOM VARIABLES THAT SATISFIES PROPERTY 3. THAT IS, FOR ANY ALTERNATIVE DISTRIBUTION $q(xyz)$, IF $q(xy) = p(xy)$, $q(xz) = p(xz)$, AND $q(yz) = p(yz)$, THEN $q(xyz) = p(xyz)$. THIS SORT OF PROPERTY COMPRISES THE MAIN ADVANTAGE OF POLYTOPE CODES. UNDER THE DISTRIBUTION p, THE VARIABLES X, Y, Z SIT IN A POLYTOPE DEFINED BY $X + Y + Z = 0$ AND $X, Y, Z \in [0, 1]$

x	y	z	$p(xyz)$
0	0	0	0
0	0	1	1/3
0	1	0	1/3
0	1	1	0
1	0	0	1/3
1	0	1	0
1	1	0	0
1	1	1	0

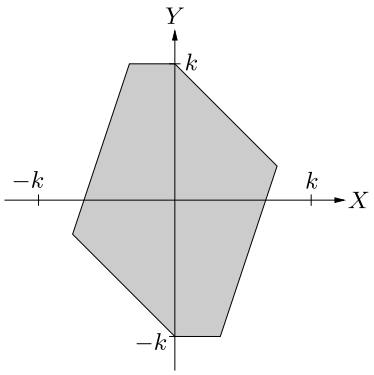

Fig. 6. An example polytope projected into the (x, y) plane.

This is the set of integer lattice points in a polytope. (Hence "Polytope Code.") The distribution in Table I—and in particular the parameterization in (12)—is a special case of the construction of the distributions in the sequel.

We now construct a distribution satisfying Properties 2–4 for arbitrarily small ϵ. Take k to be a positive integer, and let X, Y, Z, W be uniform over the set

$$\big\{(x, y, z, w) \in \{-k, \ldots, k\}^4 : x + y + z = 0 \text{ and } 3x - y + 2w = 0\big\}. \quad (13)$$

Like (12), this is the set of integer lattice points in a polytope.

By the linear constraints in (13), this distribution satisfies Property 2. Now consider Property 4. The region of (X, Y) pairs with positive probability is shown in Figure 6. Note that X and Y are not independent, because the boundedness of Z and W requires that X and Y satisfy certain linear inequalities. Nevertheless, the area of the polygon shown in Figure 6 grows as $\mathcal{O}(k^2)$. Hence

$$\frac{\log H(XYZW)}{\log |\mathcal{X}|} = \frac{\log \mathcal{O}(k^2)}{\log(2k+1)} \geq 2 - \epsilon \quad (14)$$

where the last inequality holds for sufficiently large k. Thus these variables satisfy Property 4.

We now consider Property 3. Assuming $\tilde{X}, \tilde{Y}, \tilde{Z}$ satisfy (8)–(10), we may write

$$\mathbb{E}\big[(\tilde{X} + \tilde{Y} + \tilde{Z})^2\big] \quad (15)$$
$$= \mathbb{E}\big[\tilde{X}^2 + \tilde{Y}^2 + \tilde{Z}^2 + 2\tilde{X}\tilde{Y} + 2\tilde{X}\tilde{Z} + 2\tilde{Y}\tilde{Z}\big] \quad (16)$$
$$= \mathbb{E}\big[X^2 + Y^2 + Z^2 + 2XY + 2XZ + 2YZ\big] \quad (17)$$
$$= \mathbb{E}\big[(X + Y + Z)^2\big] \quad (18)$$
$$= 0 \quad (19)$$

where (17) holds from (8)–(10), and because each term in the sum involves at most two of the three variables; and (19) holds because $X+Y+Z = 0$ by construction. Hence $\tilde{X}+\tilde{Y}+\tilde{Z} = 0$, so we may write

$$(\tilde{X}, \tilde{Y}, \tilde{Z}) = (\tilde{X}, \tilde{Y}, -\tilde{X} - \tilde{Y}) \quad (20)$$
$$\sim (X, Y, -X - Y) \quad (21)$$
$$= (X, Y, Z) \quad (22)$$

where (21) holds by (8). This verifies Property 3, and we may now conclude that the distribution on X, Y, Z, W satisfies all desired properties, so the induced Polytope Code achieves rate 2 for the Caterpillar network.

D. Remarks on the Polytope Code for the Caterpillar Network

The above argument took advantage of the linear constraint $X + Y + Z = 0$, but this constraint was in no way special. Property 3 would hold as long as X, Y, Z are subject to any linear constraint with nonzero coefficients for all three variables.

Observe that when k is large, any pair of the four variables are nearly independent, in that their joint entropy is close to the sum of their individual entropies. We have therefore constructed something like a $(4, 2)$ MDS code. In fact, if we reinterpret the linear constraints in (13) as constraints on elements x, y, z, w from a finite field, the resulting finite subspace would be exactly a $(4, 2)$ MDS code. This illustrates a general principle of Polytope Codes: any code construction on a finite field can be used to construct a Polytope Code, and many of the properties of the original code will hold over. The resulting code will be substantially harder to implement, in that it involves much longer blocklengths, and more complicated coding functions, but additional properties, such as Property 3, may hold. Sec. X provides details on transforming any linear code into a mostly-equivalent Polytope Code.

The code described above illustrates one significant drawback of Polytope Codes compared to linear codes: the rates they achieve are asymptotic. For linear codes, it is often the case that the maximum rate can be achieved with a finite field that is not too large. This is not the case for a Polytope Code. In fact, in order to achieve the best rate, both the integers n and k must go to infinity. We need n to be large in order for the cardinality of $T^n(XYZW)$ to be close to $2^{nH(XYZW)}$ (see (6)), and we need k to be large in order to satisfy Property 4 (see (14)). This paper is primarily interested in fundamental limits rather than implementation, so we will not comment much on how large n and k must be for the asymptotics to "kick in".

VII. A Polytope Code for the Cockroach Network

We return to the Cockroach network, and demonstrate a capacity-achieving Polytope Code for it. We do this not to find the capacity for the network, because we have already done so with the simpler code in Sec. V, but rather to illustrate a Polytope Code on a network satisfying the conditions of Theorem 3, which have a somewhat different flavor than the Caterpillar network.

In Sec. V, we illustrated how performing comparisons and transmitting comparison bits through the network can help defeat traitors. In Sec. VI, we illustrated how a code can be built out a distribution on a polytope, and how a special property of that distribution comes into play in the operation of the code. The Polytope Code for the Cockroach network combines these two ideas: the primary data sent through the network comes from the distribution on a polytope, but then comparisons are performed in the network in order to localize the traitor.

The first step in constructing a Polytope Code is to describe a distribution over a polytope. That is, we define a linear subspace in a real vector field, and take a uniform distribution over the polytope defined by the set of vectors with entries in $\{-k, \ldots, k\}$ for some integer k. The nature of this distribution depends on the characteristics of the linear subspace. For our code for the Cockroach network, we need one that is equivalent to a $(6, 2)$ MDS code. That is, the linear subspace sits in $\mathbb{R}^6$, has dimension 2, and is defined by four constraints such that any two variables determine the others. One choice for the subspace, for example, is the set of (a, b, c, d, e, f) satisfying

$$a + b + c = 0 \tag{23}$$

$$a - b + d = 0 \tag{24}$$

$$a + 2b + e = 0 \tag{25}$$

$$2a + b + f = 0. \tag{26}$$

Let the random variables A, B, C, D, E, F have joint distribution uniformly distributed over the polytope defined by (23)–(26) and $a, b, c, d, e, f \in \{-k, \ldots, k\}$. By a similar argument to that in Sec. VI, for large k,

$$\frac{H(ABCDEF)}{\log(2k+1)} \approx 2. \tag{27}$$

We choose a blocklength n and associate each message with a joint sequence $(a^n b^n c^n d^n e^n f^n)$ with joint type exactly equal to the distribution of the six variables. Like the code in Sec. VI, to achieve capacity we need both n and k to be large. Indeed, for large n and k, we may place one sequence a^n–f^n on each unit capacity edge in the network and operate near rate 2. These six sequences are generated at the source and then routed through the network as shown in Fig. 7. For convenience, the figure omits the n superscript.

As in Sec. VI, we define $\tilde{A}, \tilde{B}, \tilde{C}, \tilde{D}, \tilde{E}, \tilde{F}$ to have joint distribution equal to the type of the six sequences an they actually appear in the network, which may differ from the sequences sent by the source because of the adversary. In addition to forwarding a sequence, nodes 4 and 5 perform more elaborate operations. Like in the code for the Cockroach network described in Sec. V, they each perform a comparison and transmit either $=$ or $\neq$ depending on whether the comparison succeeds. In particular, they compare the types of their received sequences with the original distribution. For example, node 4 receives the two sequences b^n and c^n, from which it can construct $\tilde{B}$ and $\tilde{C}$. If the joint distribution of $(\tilde{B}, \tilde{C})$ matches that of (B, C), it sends $=$ to the destination; if not, it sends $\neq$. This single bit costs asymptotically negligible rate, so it has no effect on the achieved rate of the code for large n and k. Node 5 performs a similar action, comparing the distribution of $(\tilde{D}, \tilde{E})$ with that of (D, E), and transmitting a comparison bit to the destination.

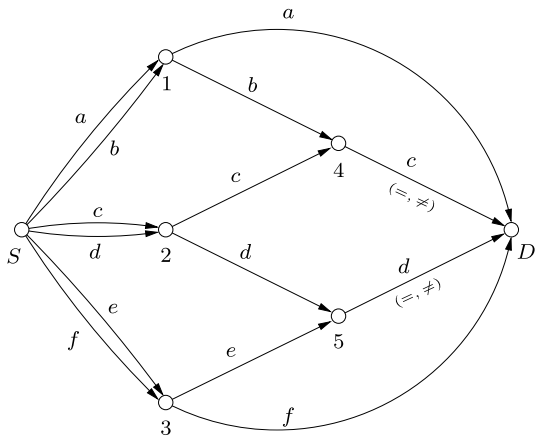

Fig. 7. A capacity-achieving Polytope Code for the Cockroach Network. The symbols shown in the network represent a sequences that have a predetermined joint type. At nodes 4 and 5, the joint types of the incoming sequences are compared against what they "should be", and $=$ or $\neq$ is sent to the destination depending on the outcome of the comparison.

Now we prove that it is possible to decode at the destination. If the traitor is node i, then the set of values received at the destination is a deterministic function of the message w and the values sent on the output of node i, which we denote z_i. We denote this function $f_i(w, z_i)$. Assuming an optimal decoder, an error can only if two messages lead to the same set of values at the destination. There are two cases: (1) for two messages w_1, w_2 there exists a node i and $z_{i,1}, z_{i,2}$ such that

$$f_i(w_1, z_{i,1}) = f_i(w_2, z_{i,2}) \tag{28}$$

and (2) for two messages w_1, w_2 there exists a pair of distinct nodes i, j and z_i, z_j such that

$$f_i(w_1, z_i) = f_j(w_2, z_j). \tag{29}$$

Because any error is of the form either (28) or (29), we never need to consider more than 2 potential traitor nodes at a time.[3] Effectively, (28) represents a single node confusing the destination between two messages, even though the destination might know the identity of the traitor, whereas (29) represents a traitor node acting in a way that appears as if a different node could be the traitor. We use this idea to parameterize different

[3]It is for the same reason that the Singleton bound is often tight, since the Singleton bound represents an error of the form (29), where the decoder cannot distinguish between two possible traitors.

actions of the traitor. In particular, given a set of values y_D received at the destination, we define the set $\mathcal{L}$ as

$$\mathcal{L} := \left\{ i \in [5] \middle| \exists w, z_i : y_D = f_i(w, z_i) \right\}. \tag{30}$$

That is, $\mathcal{L}$ is a list of nodes that could be the traitor given the received values at the destination y_D. Our decoding process at the destination is parameterized by the set $\mathcal{L}$.[4] Note that the true traitor is always contained in $\mathcal{L}$. If $|\mathcal{L}| = 1$, then the destination is able to identify the traitor, and the only possible error is (28). If $|\mathcal{L}| \geq 2$, then the destination cannot identify the traitor, so (29) becomes possible as well.

We now describe the decoding operation at the destination. The first step is to construct $\mathcal{L}$ as given by (30). Next, the destination uses $\mathcal{L}$ to decide which of the four symbols available at the destination to use to decode. Since any pair of the six original symbols contain all the information in the message, if at least two of the four symbols a, c, d, f can be determined to be trustworthy by the destination, then it can decode. The decoding rule at the destination is as follows:

Decoding Rule: Discard any symbol that was touched by every node in $\mathcal{L}$, and decode from the rest.

For example, if $\mathcal{L} = \{2\}$, then the destination discards c, d—the two symbols touched by node 2—and decodes from a, f. If $\mathcal{L} = \{2, 4\}$, the destination discards just c—the only symbol touched by both nodes 2 and 4—and decodes from a, d, f. If $\mathcal{L} = \{1, \ldots, 5\}$, then it discards no symbols and decodes from all four.

The prove the correctness of this code, we must show that the destination never decodes from a symbol that was altered by the traitor. This is easy to see if $|\mathcal{L}| = 1$, because in this case the destination knows exactly which node is the traitor, and it simply discards all symbols that may have been influenced by this node. Since no node touches more than two of the symbols available at the destination, there are always at least two remaining to decode from.

If $|\mathcal{L}| \geq 2$, the destination may use symbols touched by the traitor when decoding. For example, suppose node 2 were the traitor, and $1 \in \mathcal{L}$. Since $\mathcal{L}$ always contains the true traitor, $\{1, 2\} \subseteq \mathcal{L}$. No symbols are touched by both nodes 1 and 2, so by the decoding rule (even if $\mathcal{L}$ contains other nodes) the destination decodes using all four of its received symbols. In particular, the destination uses c and d to decode, even though both are touched by node 2. To prove correctness we must show that node 2 could not have transmitted anything but the true values of c and d. What we use to prove this is the fact that $\mathcal{L}$ contains node 1, meaning that node 2 must have acted in a way such that it appears to the destination that node 1 could be the traitor. This induces constraints on the behavior of node 2. The first is that the comparison that occurs at node 5 between d and e must succeed. If it did not, then the destination would learn it, and conclude that node 1 could not be the traitor, in which case 1 would not be in $\mathcal{L}$. Hence the distribution of $(\tilde{D}, \tilde{E})$ must match that of (D, E). This constitutes a constraint on node 2 in its transmission of d. Moreover, $(\tilde{D}, \tilde{F}) \sim (D, F)$, because the destination may observe d and f, so it could detect a difference between these two distributions if it existed. Because both symbols are untouched by node 1 and $1 \in \mathcal{L}$, the distributions must match. Furthermore, because neither e nor f are touched by the traitor node 2, $(\tilde{E}, \tilde{F}) \sim (E, F)$. To summarize:

$$(\tilde{D}, \tilde{E}) \sim (D, E), \tag{31}$$

$$(\tilde{D}, \tilde{F}) \sim (D, F), \tag{32}$$

$$(\tilde{E}, \tilde{F}) \sim (E, F). \tag{33}$$

Using these three conditions, we apply Property 3 from Sec. VI to conclude that $(\tilde{D}, \tilde{E}, \tilde{F}) \sim (D, E, F)$. We may do this because, as we argued in Sec. VI, Property 3 holds for for any three variables in a polytope subject to a single linear constraint with nonzero coefficients on each one. Since we have constructed the 6 variables to be a $(6, 2)$ MDS code, this is true here. (In the space defined by (23)–(26), the three variables D, E, F are subject to $D + E - F = 0$.) Since e and f together specify the entire message, in order for this three-way distribution to match, the only choice for d is the true value of d. Now we have to show that c can also not be corrupted by the traitor. Since the only symbol seen by the destination that could be touched by node 1 is a, we must have $(\tilde{C}, \tilde{D}, \tilde{F}) \sim (C, D, F)$, or else 1 would not be in $\mathcal{L}$. Again since any two symbols specify the entire message, and both d and f are uncorrupted by the traitor, the value for c sent by node 2 must also be its true value. Therefore the destination will not make an error by using c and d to decode.

The above analysis holds for any $\mathcal{L}$ containing $\{1, 2\}$. To demonstrate correctness of the code for all cases where $|\mathcal{L}| \geq 2$, it is enough to consider the case that $\{i, j\} \subseteq \mathcal{L}$ for all pairs of nodes i, j (whether or not $\mathcal{L}$ also contains additional nodes). In particular, for every pair of nodes i, j, we must show that if i is the traitor and $j \in \mathcal{L}$, node i is forced to transmit the true value of every symbol that is not also touched by node j. If this can be shown for each pair, the destination always decodes correctly by following the decoding rule.

Moreover, it is enough to consider each unordered pair only once. For example, as we have already performed the analysis for $i = 2$ and $j = 1$, we do not need to perform the same analysis for $i = 1$ and $j = 2$. This is justified as follows. We have shown that when node 2 is the traitor and $1 \in \mathcal{L}$, symbols c and d are uncorrupted. Therefore $(\tilde{A}, \tilde{C}, \tilde{D}, \tilde{F}) \sim (A, C, D, F)$. Hence if $1 \in \mathcal{L}$ and $(\tilde{A}, \tilde{C}, \tilde{D}, \tilde{F}) \not\sim (A, C, D, F)$, node 2 cannot be the traitor, so $2 \notin \mathcal{L}$. Now, if node 1 is the traitor and $2 \in \mathcal{L}$, then it must be the case that $(\tilde{A}, \tilde{C}, \tilde{D}, \tilde{F}) \sim (A, C, D, F)$. Since of these four symbols only a is touched by node 1, it cannot be corrupted. This same argument can apply to any pair of nodes.

We now complete the proof of correctness of the proposed Polytope Code for the Cockroach network by considering all pairs of potential traitors in the network:

[4]Because we define $\mathcal{L}$ in this non-constructive and essentially code-independent manner, our arguments for code correctness may sometimes seem backwards. We will make assumptions about $\mathcal{L}$, and then argue about what the traitor must have done, even though this is opposite to the causal relationship. We do this because it is most convenient to partition possible traitor actions based on the $\mathcal{L}$ that results. As long as our analysis considers every possible $\mathcal{L}$, we can be assured that the code can handle any possible traitor action.

$(1,2)$: Proof above.

$(1,3)$: Suppose node 1 is the traitor and $3 \in \mathcal{L}$. We must show that node 1 cannot corrupt a. We have that $(\tilde{A}, \tilde{C}, \tilde{D}) \sim (A, C, D)$, because these three symbols are not touched by node 3, and are available at the destination. Since c and d determine the message, this single constraint is enough to conclude that node 1 cannot corrupt a. This illustrates a more general principle: when considering the pair of nodes (i, j), if the number of symbols available at the destination untouched by both i or j is at least as large as the rate of the code, we may immediately conclude that no symbols can be corrupted. In fact, this principle works even for finite-field linear codes.

$(1,4)$: Follows exactly as $(1,3)$.

$(1,5)$: Follows exactly as $(1,3)$.

$(2,3)$: Follows exactly as $(1,2)$.

$(2,4)$: Suppose node 4 is the traitor and $2 \in \mathcal{L}$. The only symbol touched by both nodes 1 and 4 is c, so the destination will decode from a, d, f. But node 4 does not touch any of these symbols, so it cannot corrupt them.

$(2,5)$: Follows exactly as $(2,4)$.

$(3,4)$: Follows exactly as $(1,3)$.

$(3,5)$: Follows exactly as $(1,3)$.

$(4,5)$: Follows exactly as $(1,3)$.

VIII. The Polytope Code

We now describe the general structure of Polytope Codes and state their important properties. In Sec. VIII-A, we introduce the general distribution on a polytope, and find the joint entropy of random variables with this distribution. In Sec. VIII-B, we state the fundamental property of Polytope Codes, and provide some examples and discussion. In Sec. VIII-C, we prove several corollaries to the fundamental property that will be convenient in the proof of Theorem 3.

A. Distributions on Polytopes

Given a matrix $F \in \mathbb{Z}^{u \times m}$, consider the polytope

$$\mathcal{P}_k = \left\{ \mathbf{x} \in \mathbb{Z}^m : F\mathbf{x} = 0, |x_i| \le k \text{ for } i = 1, \ldots, m \right\}. \quad (34)$$

We may also describe this polytope in terms of a matrix K whose columns form a basis for the null-space of F. Let $\mathbf{X}$ be an m-dimensional random vector uniformly distributed over $\mathcal{P}_k$. Take n to be a multiple of the least common denominator of the distribution of $\mathbf{X}$ and let $T^n(\mathbf{X})$ be the set of sequences $\mathbf{x}^n$ with joint type exactly equal to this distribution. In a Polytope Code, each message is associated with an element of $T^n(\mathbf{X})$. Given a message and the corresponding sequence $\mathbf{x}^n$, each edge in the network holds a sequence x_i^n for some $i = 1, \ldots, m$.

As for the example codes in Sec. VI and VII, Polytope Codes are asymptotic in both n and k. The asymptotics in n are required in order to make the cardinality of $T^n(\mathbf{X})$ close to $2^{nH(\mathbf{X})}$. In particular, we are taking advantage of the fact from the theory of types that

$$\lim_{n\to\infty} \frac{1}{n} \log |T^n(\mathbf{X})| = H(\mathbf{X}). \quad (35)$$

The asymptotics in k are required in order to for the entropy function (i.e. the vector of $H(\mathbf{X}_S)$ for each $S \subset [m]$) to converge. This convergence was illustrated in Sec. VI and VII, where for large k all joint entropies can be calculated using only properties of the linear subspace defined by F. Proposition 1 states this property in general.

Proposition 1: There exists a constant $C > 0$ depending only on F (or K) such that for any $S \subseteq [m]$

$$-C - \mathcal{O}\left(\tfrac{1}{k}\right) \le H(\mathbf{X}_S) - \operatorname{rank}(K_S) \log(2k) \le \mathcal{O}\left(\tfrac{1}{k}\right). \quad (36)$$

where K_S is the matrix made up of the rows of K corresponding to the elements of S. In particular

$$\lim_{k\to\infty} \frac{H(\mathbf{X}_S)}{\log k} = \operatorname{rank}(K_S) \quad (37)$$

Proof: Let $S_1 \subset [m]$ be a minimal set such that $H(\mathbf{X}) = H(\mathbf{X}_{S_1})$. Note that $|S_1| = \operatorname{rank}(K)$. Let $S_2 := S_1^c$. Because of the linear constraint $F\mathbf{X} = 0$, there exists a matrix $\tilde{K}$ such that $\mathbf{X}_{S_2} = \tilde{K}\mathbf{X}_{S_1}$. We may assume without loss of generality that $\tilde{K}$ has only rational values. In particular, there exists a positive integer υ such that $\upsilon\tilde{K}$ is integer valued. Let $\tilde{\mathcal{P}}_k$ be the projection of $\mathcal{P}_k$ onto the sub-vector $\mathbf{x}_{S_1}$. That is, $\tilde{\mathcal{P}}_k$ is the set of vectors $\mathbf{x}_{S_1}$ for which there exists $\mathbf{x}_{S_2}$ satisfying $(\mathbf{x}_{S_1}, \mathbf{x}_{S_2}) \in \mathcal{P}_k$. We may write this set as

$$\tilde{\mathcal{P}}_k = \left\{ \mathbf{x}_{S_1} \in \mathbb{Z}^{|S_1|} \middle| \begin{array}{c} \|\mathbf{x}_{S_1}\|_\infty \le k, \\ \|\tilde{K}\mathbf{x}_{S_1}\|_\infty \le k, \\ \tilde{K}\mathbf{x}_{S_1} \in \mathbb{Z}^{|S_2|} \end{array} \right\}. \quad (38)$$

Note that $|\mathcal{P}_k| = |\tilde{\mathcal{P}}_k|$. We now find a lower bound on the cardinality of $\tilde{\mathcal{P}}_k$. Note that $\|\tilde{K}\mathbf{x}_{S_1}\|_\infty \le \|\tilde{K}\|_{\infty,1}\|\mathbf{x}_{S_1}\|_\infty$ where

$$\|\tilde{K}\|_{\infty,1} := \max_i \sum_j |\tilde{K}_{i,j}|. \quad (39)$$

Also recall that if all elements of $\mathbf{x}_{S_1}$ are multiples of υ, then $\tilde{K}\mathbf{x}_{S_1}$ is integer-valued. Thus, if we define

$$\underline{\mathcal{P}}_k := \left\{ \mathbf{x}_{S_1} \in \upsilon\mathbb{Z}^{|S_1|} : \|\mathbf{x}_{S_1}\|_\infty \le \frac{k}{\|\tilde{K}\|_{\infty,1}} \right\} \quad (40)$$

then $\underline{\mathcal{P}}_k \subset \tilde{\mathcal{P}}_k$. Now we may write

$$\begin{aligned} H(\mathbf{X}) &= \log |\mathcal{P}_k| && (41)\\ &\ge \log |\underline{\mathcal{P}}_k| && (42)\\ &= |S_1| \log\left(2\left\lfloor \frac{k}{\upsilon\|\tilde{K}\|_{\infty,1}} \right\rfloor + 1\right) && (43)\\ &= \operatorname{rank}(K)\log(2k) - C - \mathcal{O}(\tfrac{1}{k}) && (44) \end{aligned}$$

where (41) holds because $\mathbf{X}$ is uniform over $\mathcal{P}_k$, (42) holds because $|\tilde{\mathcal{P}}_k| = |\mathcal{P}_k|$, (43) holds by the definition of $\underline{\mathcal{P}}_k$, and (44) holds since $|S_1| = \operatorname{rank}(K)$, and we have defined $C := \operatorname{rank}(K)\log(\upsilon\|\tilde{K}\|_{\infty,1})$.

Now consider any $S \subset [m]$. Let $\tilde{S} \subset S$ be a minimal set such that $H(\mathbf{X}_S) = H(\mathbf{X}_{\tilde{S}})$. Note that $|\tilde{S}| = \operatorname{rank}(K_S)$. Of course $\|\mathbf{X}_{\tilde{S}}\|_\infty \le k$, so by the uniform bound

$$\begin{aligned} H(\mathbf{X}_S) &= H(\mathbf{X}_{\tilde{S}}) \\ &\le |\tilde{S}| \log(2k+1) = \operatorname{rank}(K_S)\log(2k) + \mathcal{O}(\tfrac{1}{k}). \quad (45) \end{aligned}$$

Let $T \subset [m]$ be a minimal set such that $\text{rank}(K_{S,T}) = \text{rank}(K)$; *i.e.* such that $X_{S,T}$ completely specifies X under the constraint $FX = 0$. Note that $\text{rank}(K_T) = \text{rank}(K) - \text{rank}(K_S)$. Hence

$$\begin{aligned} H(\mathbf{X}_S) &= H(\mathbf{X}_{S,T}) - H(\mathbf{X}_T|\mathbf{X}_S) && (46)\\ &\geq H(\mathbf{X}) - H(\mathbf{X}_T) && (47)\\ &\geq \text{rank}(K)\log(2k) - \text{rank}(K_T)\log(2k) - C - \mathcal{O}(\tfrac{1}{k}) && (48)\\ &= \text{rank}(K_S)\log(2k) - C - \mathcal{O}(\tfrac{1}{k}) && (49)\end{aligned}$$

where in (48) we have used both (44) and (45). Combining (45) with (49) proves (36). Taking the limit proves (37). ∎

Recall that in a Polytope Code, each element in $T^n(\mathbf{X})$ is associated with a message. Using Proposition 1, we can thus lower bound the number of messages by

$$\begin{aligned} \log|T^n(\mathbf{X})| &\geq nH(\mathbf{X}) - |\mathcal{P}_k|\log(n+1) && (50)\\ &\geq n\,\text{rank}(K)\log(2k) - nC - n\mathcal{O}(\tfrac{1}{k}) \\ &\quad -(2k+1)^{\text{rank}(K)}\log(n+1). && (51)\end{aligned}$$

Since in a Polytope Code each edge holds a value in $\{-k,\ldots,k\}^n$, the coding rate is bounded by

$$\begin{aligned} R = \frac{\log|T^n(\mathbf{X})|}{n\log(2k+1)} &\geq \text{rank}(K) - \frac{C}{\log(2k+1)} \\ &\quad - \frac{(2k+1)^{\text{rank}(K)}\log(n+1)}{n\log(2k+1)} - \mathcal{O}(\tfrac{1}{k}) && (52)\end{aligned}$$

It is clear from (52) that for sufficiently large n and k the rate approaches $\text{rank}(K)$ (*i.e.* the dimension of the hyperspace $F\mathbf{x} = 0$). The bound in (52) also gives some indication of how large n and k must be to achieve a certain gap to the asymptotic rate $\text{rank}(K)$.

In a linear code operating over the finite field $\mathbb{F}$, we may express the elements on the edges in a network $\mathbf{x} \in \mathbb{F}^m$ as a linear combination of the message $\mathbf{x} = K\mathbf{w}$, where K is a linear transformation over the finite field, and $\mathbf{w}$ is the message vector. Taking a uniform distribution on $\mathbf{w}$ imposes a distribution on $\mathbf{X}$ satisfying

$$H(\mathbf{X}_S) = \text{rank}(K_S)\log|\mathbb{F}|. \quad (53)$$

This differs from (37) only by a constant factor, and also that (37) holds only in the limit of large k. Hence, Polytope Codes achieve a similar set of entropy functions as standard linear codes. In Sec. X we return to this point in more detail.

B. Fundamental Property of Polytope Codes

In Sec. VI and VII, we saw that Property 3 played an important role in the functionality of the Polytope Codes. The following theorem states the more general version of this property. It comprises the major advantage of Polytope Codes over linear codes. After the proof of the theorem, we illustrate its application to prove Property 3. Subsequently we discuss implications and possible generalizations of the theorem. Then we prove three corollaries to be used in the sequel.

Theorem 4 (Fundamental Property of Polytope Codes): Let $\mathbf{X} \in \mathbb{R}^m$ be a random vector satisfying $F\mathbf{X} = 0$. Fix a positive integer L. For each $l \in [L]$, fix a positive integer u_l and a matrix $A_l \in \mathbb{R}^{u_l \times m}$. Suppose a second random vector $\tilde{\mathbf{X}} \in \mathbb{R}^m$ satisfies the L constraints:

$$A_l\tilde{\mathbf{X}} \sim A_l\mathbf{X} \quad \text{for } l = 1,\ldots,L. \quad (54)$$

The two vectors $\mathbf{X}$ and $\tilde{\mathbf{X}}$ are equal in distribution if the following hold:

1) There exists a positive definite $C \in \mathbb{R}^{u\times u}$ and matrices $\Sigma_l \in \mathbb{R}^{u_l \times u_l}$ such that
$$F^T C F = \sum_{l=1}^{L} A_l^T \Sigma_l A_l. \quad (55)$$
2) For some $l^* \in [L]$, there exists a matrix $Q \in \mathbb{R}^{m\times u_l}$ where $\mathbf{x} = QA_{l^*}\mathbf{x}$ for all $\mathbf{x} \in \mathbb{R}^m$ with $F\mathbf{x} = 0$.[5]

Proof: The following proof follows almost exactly the same argument as the proof of Property 3 in Sec. VI. We may write

$$\begin{aligned} \mathbb{E}\big[(F\tilde{\mathbf{X}})^T C(F\tilde{\mathbf{X}})\big] &= \sum_{l=1}^{L}\mathbb{E}\big[(A_l\tilde{\mathbf{X}})^T\Sigma_l(A_l\tilde{\mathbf{X}})\big] && (56)\\ &= \sum_{l=1}^{L}\mathbb{E}\big[(A_l\mathbf{X})^T\Sigma_l(A_l\mathbf{X})\big] && (57)\\ &= \mathbb{E}\big[(F\mathbf{X})^T C(F\mathbf{X})\big] && (58)\\ &= 0 && (59)\end{aligned}$$

where (56) and (58) follow from (55); (57) follows from (54), and because each term in the sum involves $A_l\mathbf{X}$ for some l; and (59) follows because $F\mathbf{X} = 0$. Because C is positive definite, (59) implies $F\tilde{\mathbf{X}} = 0$. Hence, the second condition in the statement of the theorem implies $QA_{l^*}\tilde{\mathbf{X}} = \tilde{\mathbf{X}}$, so we may write

$$\begin{aligned} \tilde{\mathbf{X}} &= QA_{l^*}\tilde{\mathbf{X}} && (60)\\ &\sim QA_{l^*}\mathbf{X} && (61)\\ &= \mathbf{X}. && (62)\end{aligned}$$

∎

1) Example Application of Theorem 4: We illustrate Theorem 4 by using it to re-prove Property 3 from Sec. VI. Recall that variables $X, Y, Z \in \{-k,\ldots,k\}$ satisfy $X + Y + Z = 0$, and the three pairwise distributions of $\tilde{X}, \tilde{Y}, \tilde{Z}$ match those of X, Y, Z as stated in (8)–(10). The condition (54) specifies than arbitrary projections of $\mathbf{X}$ and $\tilde{\mathbf{X}}$ are equal in distribution. This is a more general form of the conditions in (8)–(10), which specify that three two-dimensional projections of (X, Y, Z) and $(\tilde{X}, \tilde{Y}, \tilde{Z})$ are equal in distribution. In terms of the notation of Theorem 4, we have $m = 3$, $L = 3$, and

$$\begin{aligned} F &= \begin{bmatrix}1 & 1 & 1\end{bmatrix}, && (63)\\ A_1 &= \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\end{bmatrix}, && (64)\\ A_2 &= \begin{bmatrix}1 & 0 & 0\\ 0 & 0 & 1\end{bmatrix}, && (65)\\ A_3 &= \begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}. && (66)\end{aligned}$$

[5]It is equivalent to say that $\begin{bmatrix}F\\ A_{l^*}\end{bmatrix}$ has full column rank.

TABLE II
A DISTRIBUTION ON THREE BINARY RANDOM VARIABLES THAT FORMS A COUNTEREXAMPLE TO THE CONVERSE OF THEOREM 4. IN PARTICULAR, FOR ANY $q(xyz)$, IF $q(xy) = p(xy)$, $q(xz) = p(xz)$, AND $q(yz) = p(yz)$, THEN $q(xyz) = p(xyz)$. HOWEVER, THIS RANDOM VECTOR $\mathbf{X} = (X, Y, Z)$ DOES NOT SATISFY THE SECOND CONDITION IN THEOREM 4. THAT IS, NO PAIR OF THE THREE RANDOM VARIABLES DETERMINE THE THIRD

x	y	z	$p(xyz)$
0	0	0	0
0	0	1	0
0	1	0	1/6
0	1	1	1/6
1	0	0	1/6
1	0	1	1/6
1	1	0	1/6
1	1	1	1/6

We satisfy the second condition of Theorem 4 by choosing $l^* = 1$ since if $x + y + z = 0$ then $(x, y, z) = (x, y, -x - y)$.[6] To verify the first condition, we need to check that there exist Σ_l for $l = 1, 2, 3$ and a positive definite C (in this case, a positive scalar, because F has only one row) satisfying (55). If we let

$$\Sigma_l = \begin{bmatrix} \sigma_{l,11} & \sigma_{l,12} \\ \sigma_{l,21} & \sigma_{l,22} \end{bmatrix} \tag{67}$$

then, for instance,

$$A_1^T \Sigma_1 A_1 = \begin{bmatrix} \sigma_{1,11} & \sigma_{1,12} & 0 \\ \sigma_{1,21} & \sigma_{1,22} & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{68}$$

The right hand side of (55) expands to

$$\sum_{l=1}^{3} A_l^T \Sigma_l A_l = \begin{bmatrix} \sigma_{1,11} + \sigma_{2,11} & \sigma_{1,12} & \sigma_{2,12} \\ \sigma_{1,21} & \sigma_{1,22} + \sigma_{3,11} & \sigma_{3,12} \\ \sigma_{2,21} & \sigma_{3,21} & \sigma_{2,22} + \sigma_{3,22} \end{bmatrix}. \tag{69}$$

Therefore, for suitable choices of $\{\Sigma_l\}_{l=1}^3$, we can produce any matrix for the right hand side of (55). We may simply set $C = 1$ and calculate the resulting matrix for the left hand side, then set $\{\Sigma_l\}_{l=1}^3$ appropriately. This allows us to apply Theorem 4 to conclude that $(\tilde{X}, \tilde{Y}, \tilde{Z}) \sim (X, Y, Z)$.

2) Discussion of Theorem 4: Theorem 4 gives a condition under which equality in distribution of subsets of random variables imply equality in distribution of all random variables. Such properties appear fundamental to the adversary problem. Indeed, the proof in Sec. XI of a bound tighter than the cut-set bound also makes use of alternative distributions on the same alphabet, where a subset of the random variables are equal in distribution. One intriguing question is whether there are ways to prove the existence of distributions with such properties other than Theorem 4.

Interestingly, the converse of Theorem 4 is false. That is, there exists a real random vector $\mathbf{X} \in \mathbb{R}^m$ such that for some $\{A_l\}$, (54) implies $\mathbf{X} \sim \tilde{\mathbf{X}}$, even though the two conditions in the statement of the theorem do not hold for any F for which $F\mathbf{X} = 0$. An example is given in Table II. One may ask for the precise condition under which (54) implies $\mathbf{X} \sim \tilde{\mathbf{X}}$. We do not know the answer in general, but for the special case illustrated in Table II (and Table I) it is not hard to find. We state this precisely in the following proposition, proved in Appendix E.

Proposition 2: Consider binary random variables X, Y, Z. The following are equivalent:

1) For any variables $(\tilde{X}, \tilde{Y}, \tilde{Z})$, if
$$(X, Y) \sim (\tilde{X}, \tilde{Y}),\ (X, Z) \sim (\tilde{X}, \tilde{Z}),\ (Y, Z) \sim (\tilde{Y}, \tilde{Z}), \tag{70}$$
then $(X, Y, Z) \sim (\tilde{X}, \tilde{Y}, \tilde{Z})$.
2) The support set of (X, Y, Z) excludes at least one triple in each of the following sets:
$$\{(0, 0, 0), (0, 1, 1), (1, 0, 1), (1, 1, 0)\} \tag{71}$$
$$\{(0, 0, 1), (0, 1, 0), (1, 0, 0), (1, 1, 1)\}. \tag{72}$$

It would be instructive to generalize Proposition 2 to all alphabets and collections of marginal distribution constraints, but even doing so would not answer the question: When do a particular set of marginal distribution constraints exist at all? To state this very general question more formally, fix sets $\mathcal{A}_l \in [m]$ for $l = 1, \ldots, L$. Also fix an entropic vector $(h_S)_{S \subset [m]}$. Under what conditions is it possible to construct a distribution on variables $\mathbf{X} = (X_1, \ldots, X_m)$ (with arbitrary alphabet) such that (1) for each $S \subset [m]$, $H(\mathbf{X}_S) = \lambda h_S$ for some $\lambda > 0$, and (2) if the variables $\tilde{\mathbf{X}} = (\tilde{X}_1, \ldots, \tilde{X}_m)$ satisfy $\mathbf{X}_{\mathcal{A}_l} \sim \tilde{\mathbf{X}}_{\mathcal{A}_l}$ for $l = 1, \ldots, L$, then $\mathbf{X} \sim \tilde{\mathbf{X}}$? If this problem could be solved in general, it would substantially improve understanding of node-based (and even more general) adversarial attacks.

C. Corollaries to Theorem 4

In our proof of Theorem 2, we will not use Theorem 4 in its most general form. Instead, we state several corollaries that will be more convenient. The first restricts Theorem 4 to A_l matrices that simply pick off sub-vectors of $\mathbf{X}$.

Corollary 1: Let $\mathbf{X} \in \mathbb{R}^m$ be a random vector satisfying $F\mathbf{X} = 0$. Fix a positive integer L, and sets $\mathcal{A}_l \subset [m]$ for $l = 1, \ldots, L$. Suppose a second random vector $\tilde{\mathbf{X}} \in \mathbb{R}^m$ satisfies

$$\tilde{\mathbf{X}}_{\mathcal{A}_l} \sim \mathbf{X}_{\mathcal{A}_l} \text{ for } l = 1, \ldots, L. \tag{73}$$

Then $\tilde{\mathbf{X}} \sim \mathbf{X}$ if the following hold:

1) There exists a positive definite $C \in \mathbb{R}^{u \times u}$ where $G := F^T C F$ has the property that for for all pairs $i, j \in [m]$, if $\{i, j\} \not\subset \mathcal{A}_l$ for all $l \in [L]$, then $G_{i,j} = 0$.
2) For some $l^* \in [L]$, there is a matrix $Q \in \mathbb{R}^{m \times |\mathcal{A}_l|}$ where $\mathbf{x} = Q\mathbf{x}_{\mathcal{A}_{l^*}}$ for all $\mathbf{x} \in \mathbb{R}^m$ with $F\mathbf{x} = 0$.

Proof: Given the $\mathcal{A}_l$ sets, we apply Theorem 4 by constructing A_l matrices as follows. Let $e_i \in \mathbb{R}^m$ be the ith standard unit vector. That is, $(e_i)_j = 1$ if $i = j$ and 0 otherwise. Note $e_i^T \mathbf{X} = X_i$. For each $l \in [L]$, let A_l be the vertical concatenation of e_i^T for all $i \in \mathcal{A}_l$. Hence $A_l \mathbf{X} = \mathbf{X}_{\mathcal{A}_l}$, so (73) is equivalent to (54). Moreover, for suitable choice of $\Sigma_l \in \mathbb{R}^{|\mathcal{A}_l| \times |\mathcal{A}_l|}$, the product $G_l := A_l^T \Sigma_l A_l$ may be any matrix in $\mathbb{R}^{m \times m}$ where $(G_l)_{i,j} = 0$ if $i \notin \mathcal{A}$ or $j \notin \mathcal{A}$. (This fact was illustrated in (68).) Hence $\sum_l G_l$ may be any matrix such that $G_{i,j} = 0$ if $\{i, j\} \not\subset \mathcal{A}_l$ for

[6] Setting l^* to 2 or 3 would also work.

every $l \in [L]$. This is precisely the condition on G in the statement of the corollary. Thus, we can choose Σ_l so that $G = \sum_l G_l$. Since $G = F^T C F$ where C is positive definite, we have satisfied the first condition in Theorem 4. The second condition in Theorem 4 follows immediately from the second condition in the statement of the corollary and the fact that $\mathbf{x}_{\mathcal{A}_{l^*}} = A_{l^*}\mathbf{x}$. ∎

Corollary 1 is still too general for convenient use in the proof of Theorem 2. We apply Corollary 1 to three specific cases. First, we prove a generalization of Property 3 for more than three variables.

Corollary 2: Let $\mathbf{X} \in \mathbb{R}^m$ satisfy $F\mathbf{X} = 0$ for some $F \in \mathbb{R}^{1\times m}$ with $F_1 \neq 0$. If $\tilde{\mathbf{X}}$ satisfies

$$(\tilde{X}_i, \tilde{X}_j) \sim (X_i, X_j) \text{ for all } i, j \in [m] \tag{74}$$

$$(\tilde{X}_2, \cdots, \tilde{X}_m) \sim (X_2, \cdots, X_m) \tag{75}$$

then $\tilde{\mathbf{X}} \sim \mathbf{X}$.

Proof: The conditions (74)–(75) fall into the framework of Corollary 1. By (74), there are no constraints on the matrix G. Hence we may simply set $C = 1$ to satisfy the first condition in Corollary 1. The second condition for Corollary 1 is satisfied by (75), since the linear constraint $F\mathbf{x} = 0$ implies

$$x_1 = F_1^{-1}(F_2x_2 + F_3x_3 + \cdots + F_mx_m) \tag{76}$$

where F_1^{-1} is finite since $F_1 \neq 0$. ∎

Corollary 2 considers the case with m variables and $m-1$ degrees of freedom; *i.e.* a single linear constraint. The following corollary considers a case with m variables and $m-2$ degrees of freedom.

Corollary 3: Let $F \in \mathbb{R}^{2\times m}$ be such that any 2×2 submatrix of F is non-singular. Let $\mathbf{X}$ satisfy $F\mathbf{X} = 0$. Assume that $m \geq 4$, and define $\mathbf{Z} := (X_5, \ldots, X_m)$ and $\tilde{\mathbf{Z}} := (\tilde{X}_5, \ldots, \tilde{X}_m)$. If $\tilde{\mathbf{X}}$ satisfies

$$(\tilde{X}_1, \tilde{X}_2, \tilde{\mathbf{Z}}) \sim (X_1, X_2, \mathbf{Z}), \tag{77}$$

$$(\tilde{X}_3, \tilde{X}_4, \tilde{\mathbf{Z}}) \sim (X_3, X_4, \mathbf{Z}), \tag{78}$$

$$(\tilde{X}_1, \tilde{X}_3) \sim (X_1, X_3), \tag{79}$$

$$(\tilde{X}_2, \tilde{X}_4) \sim (X_2, X_4), \tag{80}$$

$$(\tilde{X}_1, \tilde{X}_4) \sim (X_1, X_4) \tag{81}$$

then $\tilde{\mathbf{X}} \sim \mathbf{X}$. Fig. 8 diagrams the constraints on $\tilde{\mathbf{X}}$.

Proof: The non-singularity of any 2×2 submatrix of F implies that any $m-2$ variables specify the other two. In particular, any subset of $m-1$ variables are subject to a single linear constraint with all nonzero coefficients. We prove Corollary 3 with two applications of Corollary 2. First, consider the group of variables $(X_1X_2X_4\mathbf{Z})$. These $m-1$ variables are subject to a single linear constraint, as in Corollary 2. From (77), (78), (80), and (81) we have all pairwise marginal constraints, satisfying (74). Furthermore, (77) satisfies (75). We may therefore apply Corollary 2 to conclude

$$(\tilde{X}_1, \tilde{X}_2, \tilde{X}_4, \tilde{\mathbf{Z}}) \sim (X_1, X_2, X_4, \mathbf{Z}). \tag{82}$$

A similar application of Corollary 2 using (77), (78), (79), and (81) allows us to conclude

$$(\tilde{X}_1, \tilde{X}_3, \tilde{X}_4, \tilde{\mathbf{Z}}) \sim (X_1, X_3, X_4, \mathbf{Z}). \tag{83}$$

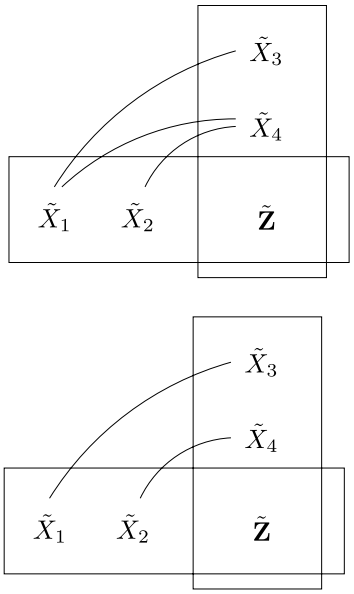

Fig. 8. The constraints on the random vector $\tilde{\mathbf{X}}$ in Corollaries 3 (above) and 4 (below). Rectangles represent a constraint on the marginal distribution of all enclosed variables; lines represent pairwise constraints on the two connected variables.

Observe that (82) and (83) share the m variables $(\tilde{X}_1, \tilde{X}_4, \tilde{\mathbf{Z}})$, which together determine $\tilde{X}_2$ and $\tilde{X}_3$ in exactly the same way that $(X_1, X_4, \mathbf{Z})$ determine X_2 and X_3. Therefore we may combine (82) and (83) to conclude $\tilde{\mathbf{X}} \sim \mathbf{X}$. ∎

All five constraints (77)–(81) are not always necessary, and we may sometimes apply Theorem 4 without (81). However, this depends on an interesting additional property of the linear constraint matrix F, as stated in the final corollary to Theorem 4.

Corollary 4: Let $F \in \mathbb{R}^{2\times m}$ be such that any 2×2 submatrix of F is non-singular, and let $\mathbf{X}$ satisfy $F\mathbf{X} = 0$. In addition, assume

$$|K_{X_1X_2\mathbf{Z}}|\ |K_{X_3X_4\mathbf{Z}}|\ |K_{X_1X_3\mathbf{Z}}|\ |K_{X_2X_4\mathbf{Z}}| < 0 \tag{84}$$

where again K is a basis for the null space of F, and $K_{\mathbf{X}_S}$ for $S \subset [m]$ is the matrix made up of the rows of K corresponding to the variables $(X_i)_{i\in S}$. If $\tilde{\mathbf{X}}$ satisfies (77)–(80) (Fig. 8 diagrams these constraints), then $\tilde{\mathbf{X}} \sim \mathbf{X}$.

Proof: We apply Corollary 1. Either (77) or (78) satisfies the second condition in Corollary 1. To verify the first condition, note that in the four constraints in (77)–(80), each pair of variables appears together except for (X_1, X_4) and (X_2, X_3). Hence G is constrained to satisfy

$$G_{1,4} = G_{2,3} = G_{3,2} = G_{4,1} = 0. \tag{85}$$

We must show that such a G exists satisfying

$$F^T C F = G \tag{86}$$

for some positive definite C. Let $\hat{G}$ be the upper left 2×2 block of G. Note that $\hat{G} = \hat{F}^T C \hat{F}$, where $\hat{F} \in \mathbb{R}^{2\times 2}$ is the first two columns of F. By assumption $\hat{F}$ is non-singular, so $\hat{G}$ is positive definite precisely when C is positive definite. Hence, it will be enough to show that it is possible to construct G satisfying (85) so that $\hat{G}$ is positive definite.

By (86), each row of G is a linear combination of rows of F; *i.e.* it forms the coefficients of a linear equality constraint imposed on the random vector $\mathbf{X}$. Consider the first row of G. Since $G_{1,4}=0$, this row represents a linear constraint on the $m-1$ variables variables $\mathbf{X}_{[m]\setminus\{4\}}$. Since any $m-2$ variables specify the other two, there is exactly one linear constraint (up to a constant) on the variables $\mathbf{X}_{[m]\setminus\{4\}}$. To proceed, we derive the coefficients of this linear constraint in terms of determinants of submatrices of K.

More generally, consider the set of $m-1$ variables $\mathbf{X}_{[m]\setminus\{i\}}$ for any $i\in[m]$. Since $\mathbf{X}_{[m]\setminus\{i\}}\in \text{span}(K_{\mathbf{X}_{[m]\setminus\{i\}}})$, we may compactly write the (unique, up to a constant) linear constraint on these variables as

$$\left|\ \mathbf{X}_{[m]\setminus\{i\}}\ \ K_{\mathbf{X}_{[m]\setminus\{i\}}}\ \right|=0. \tag{87}$$

Expanding the determinant gives

$$\sum_{j\in[m]\setminus\{i\}}\sigma_j X_j|K_{\mathbf{X}_{[m]\setminus\{i,j\}}}|=0 \tag{88}$$

where σ_j alternates between 1 and -1.

Applying (88) with $i=4$, we may write the first row of G as

$$\alpha\Big[|K_{\mathbf{X}_{[m]\setminus\{1,4\}}}|,\ -|K_{\mathbf{X}_{[m]\setminus\{2,4\}}}|,|K_{\mathbf{X}_{[m]\setminus\{3,4\}}}|,0,\\ -|K_{\mathbf{X}_{[m]\setminus\{5,4\}}}|,\ldots,\sigma_m|K_{\mathbf{X}_{[m]\setminus\{m,4\}}}|\Big] \tag{89}$$

for some $\alpha\neq 0$. Since $\mathbf{Z}=(X_5,\ldots,X_m)$, we can write

$$G_{1,1}=\alpha|K_{X_2X_3\mathbf{Z}}|, \tag{90}$$

$$G_{1,2}=-\alpha|K_{X_1X_3\mathbf{Z}}|. \tag{91}$$

Now consider the second row of G. Since $G_{2,3}=0$, this row represents a linear constraint on $\mathbf{X}_{[m]\setminus\{3\}}$. By the same argument as above,

$$G_{2,1}=\beta|K_{X_2X_4\mathbf{Z}}|, \tag{92}$$

$$G_{2,2}=-\beta|K_{X_1X_4\mathbf{Z}}| \tag{93}$$

for some $\beta\neq 0$. Moreover, G is symmetric, so $G_{1,2}=G_{2,1}$, which by (91) and (92) gives

$$\beta=-\frac{|K_{X_1X_3\mathbf{Z}}|}{|K_{X_2X_4\mathbf{Z}}|}\alpha. \tag{94}$$

Therefore

$$\hat{G}=\alpha\begin{bmatrix}|K_{X_2X_3\mathbf{Z}}| & -|K_{X_1X_3\mathbf{Z}}|\\ -|K_{X_1X_3\mathbf{Z}}| & \frac{|K_{X_1X_4\mathbf{Z}}|\ |K_{X_1X_3\mathbf{Z}}|}{|K_{X_2X_4\mathbf{Z}}|}\end{bmatrix}. \tag{95}$$

For $\hat{G}$ to be positive definite, we need

$$0<G_{1,1}=\alpha|K_{X_2X_3\mathbf{Z}}|, \tag{96}$$

$$0<|\hat{G}|=\alpha^2\left[\frac{|K_{X_2X_3\mathbf{Z}}||K_{X_1X_4\mathbf{Z}}||K_{X_1X_3\mathbf{Z}}|}{|K_{X_2X_4\mathbf{Z}}|}-|K_{X_1X_3\mathbf{Z}}|^2\right]. \tag{97}$$

To satisfy (96), we may choose any α with the same sign as $|K_{X_2X_3\mathbf{Z}}|$. The condition (97) is equivalent to

$$|K_{X_1X_3\mathbf{Z}}|\ |K_{X_2X_4\mathbf{Z}}|\Big(|K_{X_2X_3\mathbf{Z}}|\ |K_{X_1X_4\mathbf{Z}}|\\ -|K_{X_2X_4\mathbf{Z}}|\ |K_{X_1X_3\mathbf{Z}}|\Big)>0 \tag{98}$$

which may also be written as (84). ■

The necessity of satisfying (84) in order to apply Corollary 4 substantially complicates code design. When building a linear code, one need only worry about the rank of certain matrices; *i.e.* certain determinants need be nonzero. Here, we see that the signs of these determinants may be constrained as well.

IX. Achievability Proof

To prove Theorem 3, we need to specify Polytope Codes for networks satisfying 2-OUT and IN-OUT. This involves specifying the linear relationships between various symbols in the network, the comparisons that are done among them at internal nodes, and then how the destination uses the comparison information it receives to decode. We then proceed to prove that the destination always decodes correctly. The key observation in the proof is that the important comparisons that go on inside the network are those that involve a variable that does not reach the destination. This is because symbols that do reach the destination can be examined there, so further comparisons inside the network do not add anything. Therefore we will carefully route these *non-destination symbols* to maximize the utility of their comparisons. In fact, the paths $\mathbf{p}_v$ in the definition of PATHS are precisely the paths along which these non-destination symbols travel. We therefore refer to these paths as *non-destination paths*. The requirement in (3) implies that for every node having one direct edge to the destination and one other output edge, the output edge not going to the destination holds a non-destination symbol. The advantage of this is that any symbol, before exiting the network, is guaranteed to cross a non-destination symbol at a node where the two variables may be compared. This is described in much more detail in the sequel.

Before proving Theorem 3, we define one additional network property:

2-IN: All nodes in V other than the source and destination are either 2-to-1 or 2-to-2.

Note that 2-IN implies 2-OUT and IN-OUT. The following lemma allows us to assume without loss of generality that the network satisfies 2-IN. It is proved in Sec. IX-A.

Lemma 2: Let network (V,E) satisfy 2-OUT and IN-OUT. There exists a network (V',E') such that

1) (V',E') satisfies 2-IN.
2) The capacity of (V',E') is no higher than that of (V,E).
3) The value of the cut-set bound from Theorem 1 is the same for (V',E') and (V,E).
4) (V',E') satisfies PATHS if and only if (V,E) satisfies PATHS.
5) (V',E') satisfies PLANAR if and only if (V,E) satisfies PLANAR.

Proof of Theorem 3: Statement (4) in the theorem, that the cut-set bound from Theorem 1 is not greater than $M-2$ comes immediately from particularizing (1) with $A=V\setminus\{D\}$ and U containing any two nodes in $\mathcal{N}_{\text{in}}(D)$.

Next, we prove statement (3), that if the network satisfies PATHS, then rate $M-2$ is achievable. This proof comprises the bulk of the work. The proofs of statements (1) and (2), proving the achievability of rates $M-4$ and $M-3$, are simpler

and use similar arguments to the proof for $M-2$. These statements are proved in Section IX-F.

Assume the network satisfies 2-OUT, IN-OUT, and PATHS. Consider the transformed network (V', E') given by Lemma 2, which satisfies 2-IN and PATHS. It will be enough to show that rate $M-2$ is achievable for (V', E'), because by Lemma 2, then it is also achievable for (V, E). For notational convenience, we drop the primes and merely assume that (V, E) satisfies 2-IN and PATHS, and show that rate $M-2$ is achievable. We may further assume that all nodes in the network are reachable from the source. Certainly edges out of these nodes cannot carry any information about the message, so we may simply discard this portion of the network, if it exists, without changing the capacity.

We are now ready to describe how to construct a Polytope Code to achieve rate $M-2$. We proceed in several steps. The correctness of the code will be proved in Lemmas 3–6, which are stated during the description of the construction process. These Lemmas are proved in Sections IX-B–IX-E.

1) Edge Labeling: We first label all the edges in the network except those in $\mathcal{E}_{\text{in}}(D)$. These labels are denoted by the following function

$$\phi : E \setminus \mathcal{E}_{\text{in}}(D) \to \mathcal{V}_{2,1} \tag{99}$$

For a 2-to-1 node v, let $\Lambda(v)$ be the set of edges e with $\phi(e) = v$. The set $\Lambda(v)$ represents the edges carrying symbols that interact with the non-destination symbol that travels along $\mathbf{p}_v$ and terminates at node v. The following lemma states the existence of label function ϕ with the necessary properties. An example labeling satisfying the properties in the lemma is shown in Fig. 9.

Lemma 3: Assume (V, E) satisfies PATHS. There exists a function ϕ such that:

- **A** If $\phi(e) = v$, then either $\text{tail}(e) = v$ or there is an edge $e' \in \mathcal{E}_{\text{out}}(\text{tail}(e))$ such that $\phi(e') = v$. That is, the set of edges $\Lambda(v)$ with label v forms a tree rooted at v.
- **B** For every 2-to-2 node i with output edges e_1, e_2, either $e_1 \in \mathbf{p}_{\phi(e_1)}$, $e_2 \in \mathbf{p}_{\phi(e_2)}$, or $\phi(e_1) \neq \phi(e_2)$.

2) Internal Node Operation: Assume that ϕ is defined to satisfy properties (A)–(B) in Lemma 3. Given these labels, we specify how internal nodes in the network operate. Every edge in the network holds a symbol representing a linear combination of the message, as well as possibly some comparison bits. We also define a function

$$\rho : E \to [|\mathcal{E}_{\text{out}}(S)|] \tag{100}$$

that will serve as an accounting tool to track symbols as they pass through the network. We begin by assigning distinct and arbitrary values to $\rho(e)$ for all $e \in \mathcal{E}_{\text{out}}(S)$ (ρ therefore constitutes an ordering on $\mathcal{E}_{\text{out}}(S)$). Further assignments of ρ will be made recursively. This is made explicit below, but if a symbol is merely forwarded, it travels along edges with a constant ρ. When linear combinations occur at internal nodes, ρ values are manipulated, and the ρs determine exactly how this is done.

Fix a node i. Let f_1, f_2 be its two input edges. (Recall by 2-IN all nodes have two input edges.) If i is 2-to-2, let e_1, e_2 be its two output edges; if it is 2-to-1, let e be its output edge. If $\phi(f_1) = \phi(f_2)$, then node i compares the symbols on f_1 and f_2. If node i is 2-to-2, then $\phi(e_l) = \phi(f_1)$ for either $l = 1$ or 2. Node i transmits its comparison bit on e_l. If node i is 2-to-1, then it transmits its comparison bit on e. All 2-to-2 nodes forward all received comparison bits on the output edge with the same ϕ value as the input edge on which the bit was received. All 2-to-1 nodes forward all received comparison bits on its output edge.

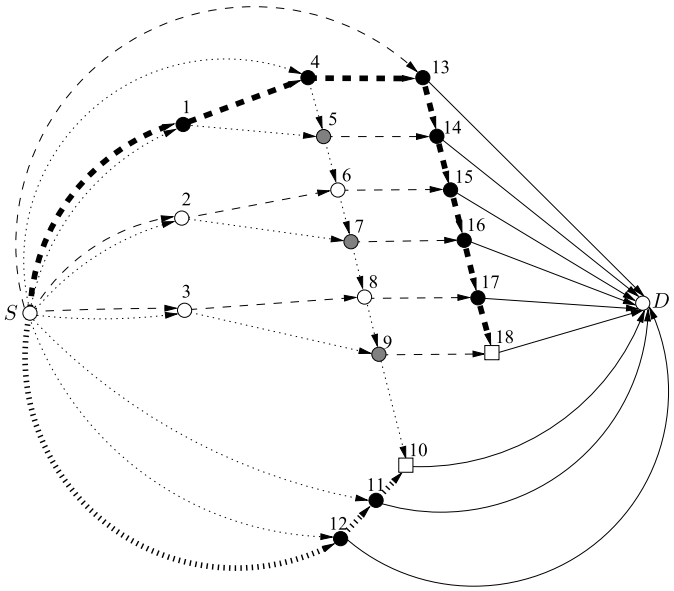

Fig. 9. A network illustrating an edge labeling satisfying the properties required by Lemma 3, as well as the $\mathcal{W}$ sets defined in (101)–(103). The edge labels are such that (A) the set of edges with a given label forms a tree rooted at the node associated with that label, and (B) each 2-to-2 node has either an output edge on a non-destination path, or two differently labeled output edges. The only 2-to-1 nodes are 10 and 18 (shown as squares). For an edge e, if $\phi(e) = 10$ it is shown with a dotted line, and if $\phi(e) = 18$ it is shown with a dashed line. Note that edges in $\mathcal{E}_{\text{in}}(D)$ are unlabeled. The non-destination paths $\mathbf{p}_{10}$ and $\mathbf{p}_{18}$ are drawn with thick lines. The proof of Lemma 3 in Sec. IX-B shows how to construct the labeling function for any network with non-destination paths satisfying PATHS. For 2-to-2 nodes, those in $\mathcal{W}_1$ (not on non-destination paths, and with differently labeled input edges) are white, those in $\mathcal{W}_2$ (branch nodes: not on non-destination paths, and with identically labeled input edges) are shaded, and those in $\mathcal{W}_3$ (on non-destination paths) are black.

We divide 2-to-2 nodes into the following sets. The linear transformation performed at node i will depend on which set it is in.

$$\mathcal{W}_1 = \left\{ i \in \mathcal{V}_{2,2} : i \notin \bigcup_{v \in \mathcal{V}_{2,1}} \mathbf{p}_v, \phi(f_1) \neq \phi(f_2) \right\} \tag{101}$$

$$\mathcal{W}_2 = \left\{ i \in \mathcal{V}_{2,2} : i \notin \bigcup_{v \in \mathcal{V}_{2,1}} \mathbf{p}_v, \phi(f_1) = \phi(f_2) \right\} \tag{102}$$

$$\mathcal{W}_3 = \left\{ i \in \mathcal{V}_{2,2} : i \in \bigcup_{v \in \mathcal{V}_{2,1}} \mathbf{p}_v \right\} \tag{103}$$

These sets are illustrated for a particular edge labeling in Fig. 9. We will sometimes refer to nodes in $\mathcal{W}_2$ as *branch nodes*, since they represent branches in $\Lambda(\phi(f_1))$. Moreover,

branch nodes are significant because a failed comparison at a branch node will cause the forwarding pattern within $\Lambda(\phi(f_1))$ to change. For an edge e, X_e denotes the symbol transmitted on e. The following gives the relationships between these symbols, which are determined by internal nodes, depending partially on the comparison bits they receive. For each node i, the action of node i depends on which set it falls in as follows:

- $\mathcal{W}_1$: Let l be such that $\phi(e_l) = \phi(f_1)$. The symbol on f_1 is forwarded to e_l, and the symbol on f_2 is forwarded onto $e_{\bar{l}}$. Set $\rho(e_l) = \rho(f_1)$, and $\rho(e_{\bar{l}}) = \rho(f_2)$.
- $\mathcal{W}_2$: Let l be such that $\phi(e_l) = \phi(f_1) = \phi(f_2)$. Let l' be such that $\rho(f_{l'}) < \rho(f_{\bar{l}})$. We will show in Lemma 4 that our construction is such that $\rho(f_1) \neq \rho(f_2)$ at all nodes, so l' is well defined. If neither f_1 nor f_2 hold a failed comparison bit, the output symbols are

$$X_{e_l} = \gamma_{i,1} X_{f_1} + \gamma_{i,2} X_{f_2} \tag{104}$$

$$X_{e_{\bar{l}}} = X_{f_{l'}} \tag{105}$$

  where coefficients $\gamma_{i,1}, \gamma_{i,2}$ are nonzero integers to be chosen later. Set output ρ values to

$$\rho(e_l) = \rho(f_{\bar{l}'}) \tag{106}$$

$$\rho(e_{\bar{l}}) = \rho(f_{l'}). \tag{107}$$

  Note that the symbol on the input edge with smaller ρ value is forwarded without linear combination. If the input edge $f_{l'}$ reports a failed comparison anywhere previously in $\Lambda(\phi(f_1))$, then (105) changes to

$$X_{e_{\bar{l}}} = X_{f_{\bar{l}'}}. \tag{108}$$

- $\mathcal{W}_3$: Let l be such $f_l \in \mathbf{p}_v$ where $v = \phi(f_l)$, and let l' be such that $e_{l'} \in \mathbf{p}_v$. The symbol on f_l is forwarded to $e_{l'}$, and the symbol on $f_{\bar{l}}$ is forwarded to $e_{\bar{l}'}$, with the following exception. If $\phi(f_1) = \phi(f_2)$ and there is a failed comparison bit sent from $f_{\bar{l}}$, then the forwarding swaps: the symbol on f_l is forwarded to $e_{\bar{l}'}$, and the symbol on $f_{\bar{l}}$ is forwarded to $e_{l'}$. Set $\rho(e_{l'}) = \rho(f_l)$ and $\rho(e_{\bar{l}'}) = \rho(f_{\bar{l}})$. Again, ρ is consistent along forwarded symbols, but only when all comparisons succeed.
- $\mathcal{V}_{2,1}$: Let l be such that $f_l \in \mathbf{p}_i$, where again i is the current node. The symbol from $f_{\bar{l}}$ is forwarded on e, unless there is a failed comparison bit sent from $f_{\bar{l}}$, in which case the symbol from f_l is forwarded on e. Set $\rho(e) = \rho(f_{\bar{l}})$.

See Fig. 10 for an illustration of the linear transformations performed at internal nodes and how they change when a comparison fails. The following Lemma gives some properties of the internal network behavior as prescribed above.

Lemma 4: The following hold:

1) For any integer $a \in [|\mathcal{E}_{\text{out}}(S)|]$, the set of edges with e with $\rho(e) = a$ form a path (we refer to this in the sequel as the $\rho = a$ *path*). Consequently, there is no node i with input edges f_1, f_2 such that $\rho(f_1) = \rho(f_2)$.
2) If there are no failed comparisons that occur in the network, then the linear transformations are such that, without the adversary, the decoder can decode any symbol in the network except those on non-destination paths.

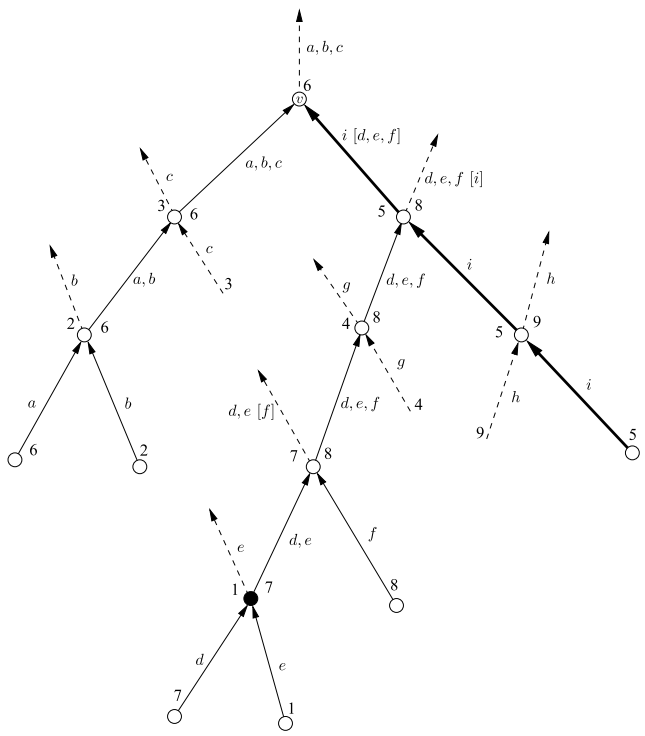

Fig. 10. An example of the linear transformations performed in $\Lambda(v)$ for some v (labeled as such). Solid edges denote $\phi(e) = v$, dashed edges denote $\phi(e) \neq v$. Thick edges denote edges in $\mathbf{p}_v$. Near the tail of each edge is the corresponding ρ value. Also shown is the symbol transmitted along that edge, given initial symbols a–i at the furthest upstream edges in the network. When several symbols are written on an edge, this indicates that the edge carries a linear combination of those symbols. The symbols indicated in brackets are those carried by the edges when the comparison at the indicated black node fails. Symbols on edges labeled without brackets do not change when the comparison fails.

3) Suppose a comparison fails at a branch node k with input edges f_1, f_2 with $v = \phi(f_1) = \phi(f_2)$. Assume without loss of generality that $\rho(f_1) < \rho(f_2)$. The forwarding pattern within $\Lambda(v)$ changes such that symbols sent along the $\rho = \rho(f_2)$ path are not decodable at the destination, but what was the non-destination symbol associated with v is decodable.

3) MDS Code Construction: The rules above explain how the symbols are combined and transformed inside the network. In addition, when the initial set of symbols are sent into the network from the source, they are subject to linear constraints. We now describe exactly how this is done. Assume that no comparisons fail in the network, so the linear relationships between symbols are unmodified. For a 2-to-1 node v, let e_v^* be the input edge to v with $e_v^* \in \mathbf{p}_v$; *i.e.* it is the last edge to hold the non-destination symbol terminating at v. Observe that it will be enough to specify the linear relationships among the symbols on $\{e_v^* : v \in \mathcal{V}_{2,1}\}$ as well as the M edges in $\mathcal{E}_{\text{in}}(D)$. These collectively form the Polytope Code equivalent of a $(M + |\mathcal{V}_{2,1}|, M - 2)$ MDS code. We must construct this code so as to satisfy certain instances of (84), so that we may apply Corollary 4 as necessary. The following Lemma states the existence of a set of linear relationships among the $M + |\mathcal{V}_{2,1}|$ variables with the required properties.

Lemma 5: For each 2-to-1 node v, let $\Xi(v)$ be the set of edges $e \in \mathcal{E}_{\text{in}}(D)$ with $\text{tail}(e) \in \mathbf{p}_v$. That is, the symbol on e, just before being sent to the destination, was compared against the non-destination symbol that terminates at v. (Note that any edge $e \in \mathcal{E}_{\text{in}}(D)$ is contained in $\Xi(v)$ for some 2-to-1 node v.) There exists a generator matrix $K \in \mathbb{Z}^{M+|\mathcal{V}_{2,1}|\times M-2}$ where each row is associated with an edge in $\{e_v^* : v \in \mathcal{V}_{2,1}\} \cup \mathcal{E}_{\text{in}}(D)$ such that for all $v_1, v_2 \in \mathcal{V}_{2,1}$ and all $f_1 \in \Xi(v_1)$, $f_2 \in \Xi(v_2)$, the constraints

$$(\tilde{X}_{f_1}, \tilde{X}_{f_2}, \tilde{\mathbf{Z}}) \sim (X_{f_1}, X_{f_2}, \mathbf{Z}) \tag{109}$$

$$(\tilde{X}_{e^*_{v_1}}, \tilde{X}_{e^*_{v_2}}, \tilde{\mathbf{Z}}) \sim (X_{e^*_{v_1}}, X_{e^*_{v_2}}, \mathbf{Z}) \tag{110}$$

$$(\tilde{X}_{f_1}, \tilde{X}_{e^*_{v_1}}) \sim (X_{f_1}, X_{e^*_{v_1}}) \tag{111}$$

$$(\tilde{X}_{f_2}, \tilde{X}_{e^*_{v_2}}) \sim (X_{f_2}, X_{e^*_{v_2}}) \tag{112}$$

imply

$$(\tilde{X}_{f_1}, \tilde{X}_{f_2}, \tilde{X}_{e^*_{v_1}}, \tilde{X}_{e^*_{v_2}} \tilde{\mathbf{Z}}) \sim (X_{f_1}, X_{f_2}, X_{e^*_{v_1}}, X_{e^*_{v_2}} \tilde{\mathbf{Z}}) \tag{113}$$

where

$$\mathbf{Z} = (X_e : e \in \mathcal{E}_{\text{in}}(D) \setminus \{f_1, f_2\}). \tag{114}$$

4) Decoding Procedure: To decode, the destination first compiles a list $\mathcal{L} \subset V$ of which nodes may be the traitor. It does this by taking all its available data: received comparison bits from interior nodes as well as the symbols it has direct access to, and determines whether it is possible for each node, if it were the traitor, to have acted in a way to cause these data to occur. If so, it adds this node to $\mathcal{L}$. For each node i, let K_i be the linear transformation from the message vector $\mathbf{W}$ to the symbols on the output edges of node i. With a slight abuse of notation, regard K_D represent the symbols on the input edges to D instead. For a set of nodes $S \subset V$, let $K_{D\perp S}$ be a basis for the subspace spanned by K_D orthogonal to

$$\bigcap_{j\in S} \text{span}(K_{j\to D}). \tag{115}$$

The destination decodes from $K_{D\perp\mathcal{L}}\mathbf{W}$. If i is the traitor, it must be that $i \in \mathcal{L}$, so

$$\text{rank}(K_{D\perp\mathcal{L}}) \geq M - \dim\left(\bigcap_{j\in S} \text{span}(K_j)\right) \tag{116}$$

$$\geq M - \text{rank}(K_i) \tag{117}$$

$$\geq M - 2 \tag{118}$$

where we used the fact that node i has at most two output edges. Since $K_{D\perp\mathcal{L}}$ has rank at least $M-2$, this is a large enough space for the destination to decode the entire message. The following Lemma allows us to conclude that all variables in the subspace spanned by $K_{D\perp\mathcal{L}}$ are trustworthy.

Lemma 6: Consider any pair of nodes i, j. Suppose i is the traitor, and acts in a way such that $j \in \mathcal{L}$. Node i cannot have corrupted any value in $K_{D\perp\{i,j\}}\mathbf{W}$. ∎

A. Proof of Lemma 2

Assume (V, E) satisfies 2-OUT and IN-OUT. We modify the network to construct (V', E') as follows. Consider an a-to-b node $i \in V$. By IN-OUT, $a \geq b$ and by 2-OUT, $b \leq 2$. We replace i with a cascade of $a-b$ 2-to-1 nodes followed by a b-to-b node. This transformation is illustrated in Fig. 11.[7] Denote the b-to-b node in the transformation i^*. Doing so for all nodes leaves the network with only 1-to-1 nodes, 2-to-2 nodes, and 2-to-1 nodes. After this process, for any 1-to-1 node not directly connected to both the source and destination, the node and the two edges connected to it are replaced with a single edge. For a 1-to-1 node directly connected to both the source and destination, add an additional edge from the source to the node, turning it into a 2-to-1 node. This completes the construction of (V', E'). Since (V', E') has only 2-to-2 and 2-to-1 nodes, it satisfies 2-IN. For any $i \in V$, We define $\mathcal{T}(i) \subset V'$ to be the set of nodes that i is transformed into, as follows. If i is a-to-b with $a > 2$, let $\mathcal{T}(i)$ be the set containing cascade of 2-to-1 nodes as well as i^*, as shown in Fig. 11. If i is 1-to-1 and not connected to both the source and destination, let $\mathcal{T}(i)$ contain the head of the edge that i is replaced with. Otherwise, let $\mathcal{T}(i)$ contain the equivalent node to i in (V', E').

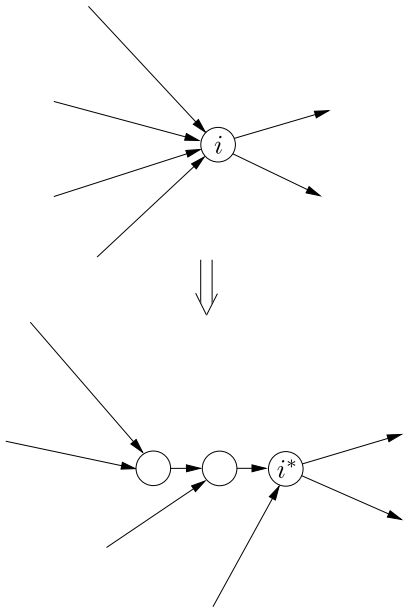

Fig. 11. An illustration of the transformation from a 4-to-2 node (above) to an equivalent set of two 2-to-1 nodes and a 2-to-2 node (below).

Next we show that the capacity of (V', E') is no higher than that of (V, E). Note that for any $i \in V$, any coding operations performed by the node or nodes in $\mathcal{T}(i)$ can be accomplished by i in (V, E). Additionally, the adversary taking control of node i in (V, E) does exactly as much damage as the traitor taking control of i^* in (V', E'), since i^* controls all edges sent to other nodes. This proves the capacity relationship.

Now we show that the cut-set bound for (V', E') is the same as that for (V, E). Let A be any cut on (V, E) with no backward edges, and $U \subset V$ with $|U| = 2s$. Define $A', U' \subset V'$ as follows:

$$A' := \bigcup_{i\in A} \mathcal{T}(i), \tag{119}$$

[7]This transformation is quite similar to the one used in [17] used to construct equivalent networks for network coding problems that are "simple": *i.e.* each node is connected to at most 3 edges.

$$U' := \{i^* : i \in U\}. \tag{120}$$

The cut A' also has no backward edges. The value of (1) will be the same for A', U' as for A, U. Hence, the cut-set bound for (V', E') is no higher than that for (V, E). To show the converse, we take A' to be any cut on (V', E') with no backward edges, and $U' \subset V'$ with $|U'| = 2s$. Define $A, U \subset V$ as follows:

$$A := \{i \in V : \mathcal{T}(i) \cap A' \neq \emptyset\}, \tag{121}$$
$$U := \{i \in V : \mathcal{T}(i) \cap U' \neq \emptyset\}. \tag{122}$$

The cut A also has no backward edges, and $|U| \leq 2s$. We need to show that the value of (1) with A, U is no higher than that with A', U'. Evaluating (1) with A', U' gives

$$|\{(i', j') \in E' : i' \in A' \setminus U', j' \notin A'\}| \tag{123}$$
$$= \sum_{i \in A} |\{(i', j') \in E' : i' \in \mathcal{T}(i) \cap A' \setminus U', j' \notin A'\}| \tag{124}$$
$$\geq \sum_{i \in A \setminus U} |\{(i', j') \in E' : i' \in \mathcal{T}(i) \cap A', j' \notin A'\} \tag{125}$$
$$= \sum_{i \in A \setminus U, i^* \in A'} |\{(i', j') \in E' : i' \in \mathcal{T}(i) \cap A', j' \notin A'\}| + \sum_{i \in A \setminus U, i^* \notin A'} |\{(i', j') \in E' : i' \in \mathcal{T}(i) \cap A', j' \notin A'\}| \tag{126}$$

where (124) holds because if $i' \in A'$, then $i' \in \mathcal{T}(i)$ for some $i \in A$, and (125) holds because if $i' \in \mathcal{T}(i)$ and $\mathcal{T}(i) \notin U$, then $i' \notin U'$. Bounding the first term in (126):

$$\sum_{i \in A \setminus U, i^* \in A'} |\{(i', j') \in E' : i' \in \mathcal{T}(i) \cap A', j' \notin A'\}| \tag{127}$$
$$= \sum_{i \in A \setminus U, i^* \in A'} |\{(i^*, j') \in E' : j' \notin A'\}| \tag{128}$$
$$\geq \sum_{i \in A \setminus U, i^* \in A'} |\{(i, j) \in E : j \notin A\}| \tag{129}$$

where

- (128) holds because if $i^* \in A'$, then $\mathcal{T}(i) \subset A'$ because A' has no backward edges, so for an edge (i', j') with $i' \in \mathcal{T}(i) \cap A'$, and $j' \notin A$, j' must not be in $\mathcal{T}(i)$. The only node in $\mathcal{T}(i)$ that connects to nodes not in $\mathcal{T}(i)$ is i^*, so the only choice for i' is i^*.
- (129) holds because each edge $(i^*, j') \in E'$ corresponds exactly to an edge $(i, j) \in E$ with $j' \in \mathcal{T}(j)$, and if $j \notin A$, then $j' \notin A'$ for all $j' \in \mathcal{T}(j)$.

Bounding the second term in (126):

$$\sum_{i \in A \setminus U, i^* \notin A'} |\{(i', j') \in E' : i' \in \mathcal{T}(i) \cap A', j' \notin A'\}| \tag{130}$$
$$= \sum_{\substack{i \in A \setminus U, \\ i^* \notin A'}} |\{(i', j') \in E' : i' \in \mathcal{T}(i) \cap A', j' \in \mathcal{T}(i) \setminus A'\}| \tag{131}$$
$$\geq \sum_{i \in A \setminus U, i^* \notin A'} |\mathcal{E}_{\text{out}}(i^*)| \tag{132}$$
$$= \sum_{i \in A \setminus U, i^* \notin A'} |\mathcal{E}_{\text{out}}(i)| \tag{133}$$
$$= \sum_{i \in A \setminus U, i^* \notin A'} |\{(i, j) \in E : j \notin A\}| \tag{134}$$

where

- (131) holds because if $i^* \notin A'$, then any $i' \in \mathcal{T}(i) \setminus A'$ cannot be i^*, and so for any edge (i', j'), $j' \in \mathcal{T}(i)$,
- (132) holds because $\mathcal{T}(i)$ contains no edges with more inputs than outputs, so any cut through $\mathcal{T}(i)$ has no fewer forward edges than the number of output edges from $\mathcal{T}(i)$, which is $|\mathcal{E}_{\text{out}}(i^*)|$,
- (133) holds because $\mathcal{E}_{\text{out}}(i^*) = \mathcal{E}_{\text{out}}(i)$,
- (134) holds because if $i^* \notin A'$, then because A' contains no backward edges, all nodes downstream from i^* are not in A', and so all nodes downstream from i in (V, E) are not in A.

Applying (129) and (134) to (126) gives that the value of (1) with V, U is no greater than that for V', U'. This completes the proof that the cut-set bounds are identical.

Now we show that (V', E') satisfies PATHS if and only if (V, E) satisfies PATHS. Suppose (V, E) satisfies PATHS. Thus there is a set of paths $\mathbf{p}_{v,i}$ satisfying (3). We may transform these paths into (V', E') so that if a path travels through v, its transformed path travels trough the nodes in $\mathcal{T}(v)$ in an analogous way, and when a path $\mathbf{p}_{v,i}$ terminates at v, its transformed paths terminates at one of the 2-to-1 nodes in $\mathcal{T}(v)$. These transformed paths have appropriate heads and tails, and remain edge-independent. Moreover, the union of sets on the right-hand side of (3) for the transformed paths includes k^* for each node k in $\mathbf{p}_{v,i}$ other than the head and tail. Note that for any $(a+1)$-to-a node $v \in V$, $\Gamma_v \subset V$ is equivalent to $\Gamma_{v'} \subset V'$, where v' is the single 2-to-1 node in $\mathcal{T}(v)$. Hence, for (V', E'), the left-hand side of (3) contains only nodes k^* where $k \in \mathcal{N}_{\text{in}}(D)$ or $k \in \Gamma_v$ for some $(a+1)$-to-a node v. Thus the transformed paths satisfy (3) in (V', E'), so (V', E') satisfies PATHS.

Now we prove the converse: suppose (V', E') satisfies PATHS, and we show (V, E) satisfies PATHS. There is a set of paths $\mathbf{p}_{v'}$ for each 2-to-1 node $v' \in V$. We may transform these paths into (V, E) so that if a path travels through a node in $\mathcal{T}(k)$ in (V, E), then its transformed paths travels through k in (V, E). For each a-to-b node $k \in V$, there are $b - a$ 2-to-1 nodes in $\mathcal{T}(k)$, so there are $b - a$ transformed paths terminating at k, as required. The fact that (3) holds for these transformed paths from similar arguments as above.

Finally, we must show that (V', E') satisfies PLANAR if and only if (V, E) satisfies PLANAR. It is easy to see that the transformation shown in Fig. 11 can be embedded in a plane (as it is in the figure). Thus this transformation does not change the planarity of the underlying graph.

B. Proof of Lemma 3

Initialize all other edges with $\phi(e) = 0$. We perform the following algorithm to set $\phi(e)$ to satisfy properties (A) and (B). We refer to an edge e as *labeled* if $\phi(e) \neq 0$. We refer to a node as *labeled* if any of its output edges are labeled.

1) For each 2-to-1 node v, set $\phi(e) = v$ for each edge $e \in \mathbf{p}_v$.

2) For any edge e such that there exists an $e' \in \mathcal{E}_{\text{out}}(\text{head}(e))$ where $e' \in \mathbf{p}_v$, set $\phi(e) = v$. By PATHS, the $\mathbf{p}_v$ paths includes all nodes in $\mathcal{N}_{\text{in}}(D)$, so any path eventually reaches a labeled edge. Furthermore, the head of any unlabeled edge cannot be in Γ_v for any v, so it can reach at least two 2-to-1 nodes.
3) Repeat the following until every edge other than those connected directly to the destination is labeled. Consider two cases:
   - *There is no 2-to-2 node not in $\mathcal{N}_{\text{in}}(D)$ with exactly one labeled output edge*: Pick an unlabeled node i. Select any path of unlabeled edges out of i until reaching a labeled node. Let v be the label of a labeled output edge from this node. For all edges e on the selected path, set $\phi(e) = v$. Observe that every node on this path except the head was previously an unlabeled 2-to-2 node, so after this step all these nodes have exactly one labeled output edge.
   - *There is a 2-to-2 node i not in $\mathcal{N}_{\text{in}}(D)$ with exactly one labeled output edge*: Let v_1 be the label on the labeled output edge from i. Select any path of unlabeled edges beginning with the unlabeled output edge from i until reaching a node with an output edge labeled v_2 with $v_2 \neq v_1$. This is always possible because any unlabeled edge must lead to at least two 2-to-1 nodes, including one other than v_1. For all edges e on the selected path, set $\phi(e) = v_2$. Observe that before this step, no node in the path except the head had an output edge labeled v_2, because if it did, the path would have stopped there. Hence, if after this step a node has 2 labeled output edges, they have different labels (satisfying property (B)).

Whenever the above algorithm sets $\phi(e) = v$, either head $(e) = v$, or e is on a path all of whose edges are labeled with v. Thus the final labeling satisfies property (A). Moreover, whenever an edge e becomes labeled, if there was another edge e' with $\text{tail}(e) = \text{tail}(e')$, either e' was unlabeled, or $\phi(e) \neq \phi(e')$. Therefore, the final labeling also satisfies property (B).

C. Proof of Lemma 4

Observe that for any 2-to-2 node, the two ρ values on the input edges are identical to the two ρ values on the output edges. For a 2-to-1 node, the ρ value on the output edge is equal to the ρ value on one of the input edges. Therefore beginning with any edge in $\mathcal{E}_{\text{out}}(S)$, we may follow a path along only edges with the same ρ value, and clearly we will hit all such edges. Property (1) immediately follows.

Property (2) follows from the fact that 2-to-2 nodes always operate such that from the symbols on the two output edges, it is possible to decode the symbols on the input edges. Therefore the destination can always reverse these transformations to recover any earlier symbols sent in the network. The only exception is 2-to-1 nodes, which drop one of their two input symbols. The dropped symbol is a non-destination symbol, so it is clear that the destination can always decode the rest.

We now prove property (3). We claim that when the comparison fails at node k, it is impossible for the destination to decode X_{f_2}. (Recall the assumption without loss of generality that $\rho(f_1) < \rho(f_2)$.) We may assume that the destination has direct access to all symbols on edges immediately subsequent to edges in $\Lambda(v)$. This can only make X_{f_2} easier to decode. Recall that $\rho(f_1) < \rho(f_2)$, so X_{f_1} is forwarded directly on the output edge of k not in $\Lambda(v)$. Therefore the destination can only decode X_{f_2} if it can decode the symbol on the output edge of k in $\Lambda(v)$. Continuing to follow the path through $\Lambda(v)$, suppose we reach an edge e_1 with $\text{tail}(e_1) = k'$, where k' is a branch node. Let e_2 be the other input edge of k'. Even if $\rho(e_1) < \rho(e_2)$, meaning k' would normally forward X_{e_1} outside of $\Lambda(v)$, because e_1 carries a failed comparison bit, k' will instead forward X_{e_2} outside of $\Lambda(v)$. Again, the destination can only decode X_{f_2} (or equivalently X_{e_1}) if it can decode the symbol on the output edge of k' in $\Lambda(v)$. If we reach a node interacting with the non-destination symbol associated with v, then because of the failed comparison bit, the formerly non-destination symbol is forwarded outside of $\Lambda(v)$ and the symbol to decode continues traveling through $\Lambda(v)$. It will finally reach v, at which point it is dropped. Therefore it is never forwarded out of $\Lambda(v)$, so the destination cannot recover it.

D. Proof of Lemma 5

From Corollary 4, it is enough to prove the existence of a K matrix satisfying

$$|K_{e^*_{v_1},e^*_{v_2},\mathbf{Z}}|\,|K_{f_1,f_2,\mathbf{Z}}|\,|K_{e^*_{v_1},f_1,\mathbf{Z}}|\,|K_{e^*_{v_2},f_2,\mathbf{Z}}| < 0. \tag{135}$$

We construct a Vandermonde matrix K to satisfy (135) for all v_1, v_2 and all f_1, f_2 in the following way. We will construct a bijective function (an ordering) α given by

$$\alpha : \{e^*_v : v \in \mathcal{V}_{2,1}\} \cup \mathcal{N}_{\text{in}}(D) \to [M + |\mathcal{V}_{2,1}|]. \tag{136}$$

For each $v \in \mathcal{V}_{2,1}$, set $\alpha(e^*_v)$ to an arbitrary but unique number in $1, \ldots, |\mathcal{V}_{2,1}|$. We may now refer to a 2-to-1 node as $\alpha^{-1}(a)$ for an integer $a \in [|\mathcal{V}_{2,1}|]$. Now set $\alpha(e)$ for $e \in \mathcal{E}_{\text{in}}(D)$ such that, in α order, the edge set $\{e^*_v : v \in \mathcal{V}_{2,1}\} \cup \mathcal{N}_{\text{in}}(D)$ is written

$$e^*_{\alpha^{-1}(1)}, e^*_{\alpha^{-1}(2)}, \ldots, e^*_{\alpha^{-1}(|\mathcal{V}_{2,1}|)}, \Xi(\alpha^{-1}(|\mathcal{V}_{2,1}|)), \Xi(\alpha^{-1}(|\mathcal{V}_{2,1}| - 1)), \ldots, \Xi(\alpha^{-1}(1)). \tag{137}$$

That is, each $\Xi(v)$ set is consecutive in the ordering, but in the opposite order as the associated non-destination edges e^*_v. Now let K be the Vandermonde matrix with constants given by α. That is, the row associated with edge e is given by

$$\begin{bmatrix} 1 & \alpha(e) & \alpha(e)^2 & \cdots & \alpha(e)^{M-3} \end{bmatrix}. \tag{138}$$

We claim the matrix K given by (138) satisfies (135). Fix v_1, v_2, and $f_1 \in \Xi(v_1)$, $f_2 \in \Xi(v_2)$. Due to the Vandermonde structure of K, we can write the determinant of a square submatrix in terms of the constants $\alpha(e)$. For instance,

$$|K_{e^*_{v_1},e^*_{v_2},\mathbf{Z}}| = [\alpha(e^*_{v_2}) - \alpha(e^*_{v_1})] \cdot \prod_{e \in \mathbf{Z}} [\alpha(e) - \alpha(e^*_{v_1})][\alpha(e) - \alpha(e^*_{v_2})] \prod_{\substack{e,e' \in \mathbf{Z}, \\ \alpha(e) < \alpha(e')}} [\alpha(e') - \alpha(e)] \tag{139}$$

where we have assumed without loss of generality that the rows of $K_{\mathbf{Z}}$ are ordered according to α. Expanding the determinants in (135) as such gives

$$\begin{aligned}|K_{e^*_{v_1},e^*_{v_2},\mathbf{Z}}|\ |K_{f_1,f_2,\mathbf{Z}}|\ |K_{e^*_{v_1},f_1,\mathbf{Z}}|\ |K_{e^*_{v_2},f_2,\mathbf{Z}}| \\ = [\alpha(e^*_{v_2})-\alpha(e^*_{v_1})][\alpha(f_2)-\alpha(f_1)][\alpha(f_1)-\alpha(e^*_{v_1})] \\ \cdot[\alpha(f_2)-\alpha(e^*_{v_2})]\prod_{e\in\mathbf{Z}}[\alpha(e)-\alpha(e^*_{v_1})]^2[\alpha(e)-\alpha(e^*_{v_2})]^2 \\ \cdot[\alpha(e)-\alpha(f_1)]^2[\alpha(e)-\alpha(f_1)]^2 \\ \cdot\prod_{e,e'\in\mathbf{Z},\alpha(e)<\alpha(e')}[\alpha(e')-\alpha(e)]^4. \end{aligned} \tag{140}$$

Recall $f_1 \in \Xi(v_1)$, $f_2 \in \Xi(v_2)$. Since we chose α such that the Ξ sets are in opposite order to the edges e^*_v, we have

$$[\alpha(e^*_{v_2})-\alpha(e^*_{v_1})][\alpha(f_2)-\alpha(f_1)]<0. \tag{141}$$

Moreover, since all the Ξ sets have larger α values than the edges e^*_v,

$$\alpha(f_1)-\alpha(e^*_{v_1})>0, \tag{142}$$

$$\alpha(f_2)-\alpha(e^*_{v_2})>0. \tag{143}$$

Hence, there is exactly one negative term in (140), from which we may conclude (135).

E. Proof of Lemma 6

The random vector $\mathbf{W}$ is distributed according to the type of the message vector as it is produced as the source. We formally introduce the random vector $\tilde{\mathbf{W}}$ representing the message as it is transformed in the network. As in our examples, this vector is distributed according to the joint type of the sequences as they appear in the network, after being corrupted by the adversary. For each edge e, we define X_e and $\tilde{X}_e$ similarly as random variables jointly distributed with $\mathbf{W}$ and $\tilde{\mathbf{W}}$ respectively with distributions given by the expected and corrupted joint types.

For every pair of nodes (i, j), we need to prove both of the following:

$$\text{If } i \text{ is the traitor, and } j\in\mathcal{L}, \text{ then } i \text{ cannot corrupt values in } K_{D\perp\{i,j\}}\mathbf{W}. \tag{144}$$

$$\text{If } j \text{ is the traitor, and } i\in\mathcal{L}, \text{ then } j \text{ cannot corrupt values in } K_{D\perp\{i,j\}}\mathbf{W}. \tag{145}$$

First we show that each of these implies the other, so it will be enough to prove just one. Suppose (144) holds. Therefore, if the distribution observed by the destination of $K_{D\perp\{i,j\}}\tilde{\mathbf{W}}$ does not match that of $K_{D\perp\{i,j\}}\tilde{\mathbf{W}}$, then at least one of i, j will not be in $\mathcal{L}$. If they both were in $\mathcal{L}$, it would have had to be possible for node i to be the traitor, make it appear as if node j were the traitor, but also corrupt part of $K_{D\perp\{i,j\}}W$. By (144), this is impossible. Hence, if j is the traitor and $i\in\mathcal{L}$, then the distribution of the $K_{D\perp\{i,j\}}Y_D$ must remain uncorrupted. This vector includes $K_{D\perp j}W$, a vector that can certainly not be corrupted by node j. Since $\text{rank}(K_{D\perp j})\geq M-2$, and there are only $M-2$ degrees of freedom, the only choice node j has to ensure that the distribution of $K_{D\perp\{i,j\}}W$ matches p is to leave this entire vector uncorrupted. That is, (145) holds.

Fix a pair (i, j). We proceed to prove either (144) or (145). Doing so will require placing constraints on the actions of the traitor imposed by comparisons that occur inside the network, then applying one of Corollary 2–4 in Sec. VIII. Let $K_{\perp i}$ be a basis for the space orthogonal to K_i. If node i is the traitor, we have that $K_{\perp i}\tilde{\mathbf{W}}\sim K_{\perp i}\mathbf{W}$. Moreover, since $j\in\mathcal{L}$, $K_{D\perp j}\tilde{\mathbf{W}})\sim K_{D\perp j}\mathbf{W}$. These two constraints are analogous to (78) and (77) respectively, where the symbols on the output of node i are analogous to X_1, X_2. The subspace of K_D orthogonal to both K_i and K_j corresponds to $\mathbf{Z}$ in the example. We now seek pairwise constraints of the form (79)–(81) from successful comparisons to apply Theorem 4.

Being able to apply Theorem 4 requires that $K_{D\perp j}$ has rank $M-2$ for all j. Ensuring this has to do with the choices for the coefficients $\gamma_{i,1}, \gamma_{i,2}$ used in (104). A rank deficiency in $K_{D\perp j}$ is a singular event, so it is not hard to see that random choices for the γ will cause this to occur with small probability. Therefore such γ exist.

We now discuss how pairwise constraints on the output symbols of i or j are found. Consider the following cases and sub-cases:

- $i, j\in\mathcal{W}_1\cup\mathcal{W}_2$: Suppose node i is the traitor. Let e_1, e_2 be the output edges of node i. For each $l=1,2$, we look for constraints on X_{e_l} by following the $\rho=\rho(e_l)$ path until one of the following occurs:
  - *We reach an edge on the $\rho=\rho(e_l)$ path carrying a symbol influenced by node j*: This can only occur immediately after a branch node k with input edges f_1, f_2 where $\rho(f_1)=\rho(e_l)$, $\rho(f_2)<\rho(f_1)$, and X_{f_2} is influenced by node j. At node k, a comparison occurs between $\tilde{X}_{f_1}$, which is influenced by node i but not j, and $\tilde{X}_{f_2}$. If the comparison succeeds, then this places a constraint on the distribution of $(\tilde{X}_{f_1},\tilde{X}_{f_2})$. If the comparison fails, the forwarding pattern changes such that the $\rho=\rho(e_l)$ path becomes a non-destination path; *i.e.* the value placed on e_l does not affect any variables available at the destination. Hence, the subspace available at the destination that is corruptible by node i is of dimension at most one.
  - *We reach node j itself*: In this situation, we make use of the fact that we only need to prove that node i cannot corrupt values available at the destination that cannot also be influenced by node j. Consider whether the $\rho=\rho(e_l)$ path, between i and j, contains a branch node k with input edges f_1, f_2 such that $\rho(f_1)=\rho(e_l)$ and $\rho(f_2)>\rho(f_1)$. If there is no such node, then X_{e_l} cannot influence any symbols seen by the destination that are not also being influenced by j. That is, X_{e_l} is in $\text{span}(K_{i\to D}\cap K_{j\to D})$, so we do not have anything to prove. If there is such a branch node k, then the output edge e of k with $\rho(e)=\rho(f_2)$ contains a symbol influenced by i and not j. We may now follow the $\rho=\rho(e)$ path from here to find a constraint on X_{e_l}. If a comparison fails further along causing the forwarding pattern to change such that the $\rho=\rho(e)$ path does not reach the destination, then the potential influence of X_{e_l} on

a symbol seen by the destination not influenced by node j is removed, so again we do not have anything to prove.

– *The $\rho = \rho(e_l)$ path leaves the network without either of the above occurring*: Immediately before leaving the network, the symbol will be compared with a non-destination symbol. This comparison must succeed, because j cannot influence the non-destination symbol. This gives a constraint $\tilde{X}_{e_l}$.

We may classify the fates of the two symbols out of i as discussed above as follows:

1) Either the forwarding pattern changes such that the symbol does not reach the destination, or the symbol is in $\text{span}(K_{i\to D} \cap K_{j\to D})$, and so we do not need to prove that it cannot be corrupted. Either way, we may ignore this symbol.
2) The symbol leaves the network, immediately after a successfully comparison with a non-destination symbol.
3) The symbol is successfully compared with a symbol influenced by node j. In particular, this symbol from node j has a strictly smaller ρ value than $\rho(e_l)$.

We divide the situation based on which of the above cases occur for $l = 1, 2$ as follows:

– *Case 1 occurs for both $l = 1, 2$*: We have nothing to prove.

– *Case 1 occurs for (without loss of generality) $l = 1$*: Either case 2 or 3 gives a successful comparison involving a symbol influenced by $\tilde{X}_{e_{\bar{l}}}$. Applying Corollary 2 allows us to conclude that $\tilde{X}_{e_{\bar{l}}}$ cannot be corrupted.

– *Case 2 occurs for both $l = 1, 2$*: If the two paths reach different non-destination symbols, then we may apply Lemma 5 to conclude that node i cannot corrupt either $\tilde{X}_{e_1}$ nor $\tilde{X}_{e_2}$. Suppose, on the other hand, that each path reaches the same non-destination path, in particular the one associated with 2-to-1 node v. Since $\phi(e_1) \neq \phi(e_2)$, assume without loss of generality that $\phi(e_1) \neq v$. We may follow the path starting from e_1 through $\Gamma(v)$ to find an additional constraint, after which we may apply Corollary 3. All symbols on this path are influenced by $\tilde{X}_{e_1}$. This path eventually crosses the non-destination path associated with v. If the symbol compared against the non-destination symbol at this point is not influenced by j, then the comparison succeeds, giving an additional constraint. Otherwise, there are two possibilities:

  ∗ *The path through $\Gamma(v)$ reaches j*: There must be a branch node on the path to $\Gamma(v)$ before reaching j such that the path from e_1 has the smaller ρ value. If there were not, then case 1 would have occurred. Consider the most recent such branch node k in $\Gamma(v)$ before reaching j. Let f_1, f_2 be the input edges to k, where f_1 is on the path from e_1. We know $\rho(f_1) < \rho(f_2)$. The comparison at k must succeed. Moreover, this successful comparison comprises a substantial constraint, because the only way the destination can decode X_{f_2} is through symbols influenced by node j.

  ∗ *The path through $\Gamma(v)$ does not reach j*: Let k be the first common node on the paths from i and j through $\Gamma(v)$. Let f_1, f_2 be the input edges of k, where f_1 is on the path from i and f_2 is on the path from j. If the comparison at k succeeds, this provides a constraint. If it fails, then the forwarding pattern changes such that the $\rho = \rho(f_1)$ path becomes a non-destination path. Since we are not in case 1, $\rho(e_1) \neq \rho(f_1)$, but a symbol influenced by X_{e_1} is compared against a symbol on the $\rho = \rho(f_1)$ path at a branch node in $\Gamma(v)$. This comparison must succeed, providing an additional constraint.

– *Case 3 occurs for (without loss of generality) $l = 1$, and either case 2 or 3 occurs for $l = 2$*: We now suppose instead that node j is the traitor. That is, we will prove (145) instead of (144). Recall that a successful comparison occurs at a branch node k with input edges f_1, f_2 where $\tilde{X}_{f_1}$ is influenced by $\tilde{X}_{e_1}$, $\tilde{X}_{f_2}$ is influenced by node j, and $\rho(f_2) < \rho(f_1)$. Let e_1', e_2' be the output edges of node j, and suppose that $\rho(e_1') = \rho(f_2)$; *i.e.* the symbol X_{f_2} is influenced by $X_{e_1'}$. The success of the comparison gives a constraint on $\tilde{X}_{e_1'}$. Since $\rho(f_2) < \rho(f_1)$, we may continue to follow the $\rho = \rho(f_2)$ path from node k, and it continues to be not influenced by node i. As above, we may find an additional constraint on $X_{e_1'}$ by following this ρ path until reaching a non-destination symbol or reaching another significant branch node. Furthermore, we may find a constraint on $\tilde{X}_{e_2'}$ in a similar fashion. This gives three constraints on $\tilde{X}_{e_1'}, \tilde{X}_{e_2'}$, enough to apply Corollary 3, and conclude that node j cannot corrupt its output symbols.

- $i \in \mathcal{W}_3 \cup \mathcal{V}_{2,1} \setminus \mathcal{N}_{\text{in}}(D)$, $j \in \mathcal{W}_1 \cup \mathcal{W}_2$: Assume node i is the traitor. If $i \in \mathcal{V}_{2,1}$ with single output edge e such that $e \in \mathbf{p}_v$ for some v, then node i controls no symbols received at the destination and we have nothing to prove. Otherwise, it controls just one symbol received at the destination, so any single constraint on node i is enough. If i is 2-to-1, let e' be its output edge; if i is 2-to-2 and in $\mathcal{W}_3$, let e' be its output edge such that e' is not in $\mathbf{p}_v$ for any v. Since we assume $i \notin \mathcal{N}_{\text{in}}(D)$, the $\rho = \rho(e')$ path is guaranteed to cross a non-destination path after node i. As above, follow the $\rho = \rho(e')$ path until reaching a branch node k at which the symbol is combined with one influenced by node j. If the comparison at node k succeeds, it gives a constraint on $\tilde{X}_{e'}$. If the comparison fails, then the forwarding pattern will change such that the $\rho = \rho(e')$ path will fail to reach the destination, so we're done.
- $i \in \mathcal{W}_1 \cup \mathcal{W}_2$, $j \in \mathcal{N}_{\text{in}}(D)$: Assume node i is the traitor. By construction, since one output edge of j goes directly into the destination, the other must be on a non-destination path. Hence, j only controls one

symbol at the destination, so we again need to place only one constraint on node i. Let $e \in \mathcal{E}_{\text{out}}(i)$ be such that $\phi(e) \neq \phi(e')$ for all $e' \in \mathcal{E}_{\text{out}}(j)$. This is always possible, since the two output edges of i have different ϕ values, and since one output edge of j goes directly to the destination, only one of the output edges of j has a ϕ value. Let $v = \phi(e)$. Follow the path from e through $\Lambda(v)$ until reaching the non-destination symbol at node k with input edges f_1, f_2. Assume $\tilde{X}_{f_1}$ is influenced by $\tilde{X}_e$ and $\tilde{X}_{f_2}$ is a non-destination symbol. The comparison between these two symbols must succeed, because node j cannot influence either $\tilde{X}_{f_1}$ or $\tilde{X}_{f_2}$. This places the necessary constraint on $\tilde{X}_e$.

- $i, j \in \mathcal{W}_3 \cup \mathcal{V}_{2,1}$: Nodes i, j each control at most one symbol available at the destination, so either one, in order to make it appear as if the other could be the traitor, cannot corrupt anything.

F. Proof of Achievability of $M-4$ and $M-3$

Now we prove statement (1) of Theorem 3, that rate $M-4$ is always achievable if the network satisfies IN-OUT and 2-OUT. The source generates M symbols from an $(M, M-4)$ MDS code. (Any MDS code will do, including a linear one.) These M symbols are routed along M edge-independent paths from source to destination. These paths exist since the max-flow is M. Since no node has more than 2 output edges, no traitor could affect more than 2 of the symbols. A $(M, M-4)$ MDS code can correct any 2 errors, so the destination can always decode.

We now briefly sketch the proof of statement (2) of Theorem 3, that rate $M-3$ is achievable if the cut-set bound is at least $M-3$. The proof is far less complicated than the proof for the $M-2$ case, but it makes use of many of the same ingredients. First use Lemma 2 to transform the network into one satisfying 2-IN. Then use the cut-set bound to conclude that the set of 2-to-2 nodes that cannot reach any 2-to-1 node forms a path. (This is most easily seen using Lemma 7 in Appendix XII, a consequence of the cut-set bound.) For each 2-to-1 node v, set $\mathbf{p}_v$ to an arbitrary path satisfying $\text{tail}(\mathbf{p}_v) = S$ and $\text{head}(\mathbf{p}_v) = v$. Next perform a similar edge labeling as in Lemma 3, defining label function:

$$\phi : E \setminus \mathcal{E}_{\text{in}}(D) \to \mathcal{V}_{2,1} \cup \{\text{null}\}. \tag{146}$$

That is, some edge labels may be null. Property (A) from Lemma 3 must hold, but property (B) is replaced with

B′ For every 2-to-2 node that can reach at least one 2-to-1 node, at least one of its output edges must have a non-null label.

Internal nodes operate in the same way based on the edge labels as above, where symbols are always forwarded along edges with null labels. The decoding process is the same. Proving an analogous version of Lemma 6 requires only finding a single constraint on i or j, and then applying Corollary 2. This is always possible since by property (B′), either i or j is guaranteed to have a label on an output edge, unless they are both in the single path with no reachable 2-to-1 nodes, in which case they influence the same symbol reaching the destination.

X. Relationship Between Polytope Codes and Linear Codes

In this section we show that linear codes can usually be converted into Polytope Codes with no change in rate. Since Polytope Codes require large values of the blocklength n and the polytope size k, there would be little reason in practice to do this, but we provide the results to give further insights into the nature of Polytope Codes. First we prove a proposition on the rates achieved by linear codes with adversaries. Then we construct a sub-class of Polytope Codes that are similar to traditional linear codes, and prove an equivalent proposition about this class. Finally we comment on these results.

A. Rate of Linear Codes

Consider a linear code over a finite field $\mathbb{F}$ where each edge holds a value on $\mathbb{F}$ for some blocklength n. Let $w \in \mathbb{F}^R$ be the message for rate R. For each edge e, we denote the value sent along e by $x_e \in \mathbb{F}$. We define a linear code by

- for each edge $e \in \mathcal{E}_{\text{out}}(S)$, an encoding operator
$$K_{\to e} \in \mathbb{F}^{1\times R} \tag{147}$$
- and for each pair of edges e, f with $\text{head}(e) = \text{tail}(f)$, a coding operator
$$K_{e\to f} \in \mathbb{F}. \tag{148}$$

The encoding operation at the source is done by setting $x_e = K_{\to e} w$ for each $e \in \mathcal{E}_{\text{out}}(s)$. Then, for each internal node i, if i is honest, it transmits for each $f \in \mathcal{E}_{\text{out}}(i)$

$$x_f = \sum_{e\in\mathcal{E}_{\text{in}}(i)} K_{e\to f} x_e. \tag{149}$$

These operators do not specify the decoding at the destination, but Proposition 3, stated below, gives the maximum achievable rate for any linear code with a given set of coefficients $K_{\to e}, K_{e\to f}$. We then proceed to show a similar result for Polytope Codes.

Before stating the proposition, we make the following definitions. For a subset of edges $F \subset E$, let x_F be the vector of all values set along edges in F. For each node i, let $x_i := x_{\mathcal{E}_{\text{out}}(i)}$, and let $y_i := x_{\mathcal{E}_{\text{in}}(i)}$. In particular, x_S is the vector of values sent from the source, and y_D is the vector of values received at the destination. Let $K_{\to S}$ be the vector of $K_{\to e}$ for all $e \in \mathcal{E}_{\text{out}}(S)$. For a path $\mathbf{p}$ with edges $(e_1, \ldots, e_L)$, let

$$K_{\mathbf{p}} := \prod_{l=1}^{L-1} K_{e_l\to e_{l+1}}. \tag{150}$$

For any edges two edges e and f with $\text{head}(e) \neq \text{tail}(f)$, define

$$K_{e\to f} := \sum_{\text{paths } \mathbf{p} \text{ with } \text{tail}(\mathbf{p})=e, \text{head}(\mathbf{p})=f} K_{\mathbf{p}}. \tag{151}$$

For sets of edges $F_1, F_2 \subset E$, let $K_{F_1\to F_2} \in \mathbb{F}^{|F_2|\times|F_1|}$ be a matrix with $(K_{F_1\to F_2})_{f_1,f_2} = K_{f_1\to f_2}$. For notational convenience, for nodes i, j, also define $K_{i\to j} := K_{\mathcal{E}_{\text{out}}(i)\to\mathcal{E}_{\text{in}}(j)}$ and

for $F \subset E$, $K_{F\to j} := K_{F\to\mathcal{E}_{\text{in}}(j)}$. Note that if all nodes are honest

$$y_D = K_{S\to D}K_{\to S}w. \tag{152}$$

It follows that with no traitors, the message can be decoded without error at the destination if and only if $R \leq \text{rank}(K_{S\to D}K_{\to S})$. The following Proposition generalizes this fact to the adversary problem.

Proposition 3: Fix $\mathcal{F} \subset 2^E$. Suppose the adversary may control any set of links $F \in \mathcal{F}$.[8] For a linear code given by coding operators $(K_{\to e} : e \in \mathcal{E}_{\text{out}}(S))$ and $(K_{e\to f} : \text{head}(e) = \text{tail}(f))$, the message can be decoded reliably if and only if

$$R \leq \min_{F_1,F_2\in\mathcal{F}} \text{rank}([K_{S\to D}K_{\to S}\ K_{F_1\to D}\ K_{F_2\to D}]) - \text{rank}([K_{F_1\to D}\ K_{F_2\to D}]). \tag{153}$$

Proof: If the adversary controls the links in $F \in \mathcal{F}$, then the values received by the destination are

$$y_D = K_{S\to D}K_{\to S}w + K_{F\to D}z_F \tag{154}$$

where $z_F \in \mathbb{F}^{|F|}$ is the difference between what the adversary sends on its controlled links and what it would have been sent on those links with no adversary.

First we assume a decoder that can decode the message reliably, and we show (153) holds. Let $F_1, F_2 \subset \mathcal{F}$ be minimizers in (153). Let G be a matrix whose columns form a basis for the subspace of $\text{span}(K_{S\to D}K_{\to S})$ that is orthogonal to $\text{span}([K_{F_1\to D}\ K_{F_2\to D}])$. Hence G has full column rank, and

$$\text{rank}(G) = \text{rank}([K_{S\to D}K_{\to S}\ K_{F_1\to D}\ K_{F_2\to D}]) - \text{rank}([K_{F_1\to D}\ K_{F_2\to D}]). \tag{155}$$

We need to prove that $R \leq \text{rank}(G)$. Moreover,

$$\text{span}(K_{S\to D}K_{\to S}) \subset \text{span}([G\ K_{F_1\to D}\ K_{F_2\to D}]). \tag{156}$$

Therefore there exists matrices H_1, H_2, H_3 such that

$$K_{S\to D}K_{\to S} = GH_1 + K_{F_1\to D}H_2 + K_{F_2\to D}H_3. \tag{157}$$

Suppose by contradiction that (153) does not hold; *i.e.* $R > \text{rank}(G)$. Over all messages $w \in \mathbb{F}^R$, GH_1w can take on at most $|\mathbb{F}|^{\text{rank}(G)}$ values, so there must be two messages w_1, w_2 such that $GH_1w_1 = GH_1w_2$. If the message is w_1, the set of adversarial edges is F_1, and $z_{F_1} = H_2(w_2 - w_1)$, then by (154) and (157)

$$y_D = \big(GH_1 + K_{F_1\to D}H_2 + K_{F_2\to D}H_3\big)w_1 + K_{F_1\to D}H_2(w_2 - w_1) \tag{158}$$

$$= GH_1w_1 + K_{F_1\to D}H_2w_2 + K_{F_2\to D}H_3w_1. \tag{159}$$

If the message is w_2, the set of adversarial edges is F_2, and $Z_{F_2} = H_2(w_2 - w_1)$, then precisely the same y_D appears. Therefore the destination cannot distinguish w_1 from w_2, and no matter what it does it will make an error on one of them.

Now we prove that there exists a reliable decoder if (153) holds. This decoder is as follows: given y_D, find the unique $w \in \mathbb{F}^R$, $F \in \mathcal{F}$, and $z_F \in \mathbb{F}^{|F|}$ such that (154) holds. An error will only occur if there is no unique w satisfying (154). That is, for two messages $w_1 \neq w_2$, two potential adversary locations $F_1, F_2 \in \mathcal{F}$ (we allow $F_1 = F_2$), and two traitor actions z_{F_1}, z_{F_2}

$$K_{S\to D}K_{\to S}w_1 + K_{F_1\to D}z_{F_1} = K_{S\to D}K_{\to S}w_2 + K_{F_2\to D}z_{F_2}. \tag{160}$$

Using the same decomposition as in (157), we have

$$GH_1(w_1 - w_2) + K_{F_1\to D}\big(H_2(w_1 - w_2) + z_{F_1}\big) + K_{F_2\to D}\big(H_3(w_1 - w_2) - z_{F_2}\big) = 0. \tag{161}$$

Since the columns of G are orthogonal to the columns of $[K_{F_1\to D}\ K_{F_2\to D}]$, this implies

$$GH_1(w_1 - w_2) = 0. \tag{162}$$

However by construction $\text{rank}(GH_1) = \text{rank}(G) \geq R$. Hence (162) cannot hold if $w_1 \neq w_2$. ∎

B. Rate of Linear-Like Polytope Codes

We construct a Polytope Code in a manner analogous to the finite field linear code discussed above. Unlike the Polytope Codes in Sec. VI, VII, and IX, this code involves no comparisons, and hence no changes in internal node behavior or decoding strategy based on these comparisons. This essentially gives up the major advantage of Polytope Codes, but we do this only to show that Polytope Codes can impersonate linear codes in most cases. Given a message dimension r,[9] coding operations are given by

- for each edge $e \in \mathcal{E}_{\text{out}}(S)$, an encoding operator
$$K_{\to e} \in \mathbb{Z}^{1\times r} \tag{163}$$
- and for each pair of edges e, f with $\text{head}(e) = \text{tail}(f)$, a coding operator
$$K_{e\to f} \in \mathbb{Z}. \tag{164}$$

We define random variables $X_e \in \{-k, \ldots, k\}$ for each edge e and message variable $W \in \{-k, \ldots, k\}^r$ to be uniform over the polytope given by

$$X_e = K_{\to e}W \quad \text{for } e \in \mathcal{E}_{\text{out}}(S) \tag{165}$$

$$X_f = \sum_{e\in\mathcal{E}_{\text{in}}(i)} K_{e\to f}X_e \quad \text{for } i \in V \setminus \{S, D\},\ f \in \mathcal{E}_{\text{out}}(i). \tag{166}$$

As described in Sec. VIII, we fix a blocklength n, and then each element of the type class given by this distribution is associated with a message. Each edge holds a sequence in $\{-k, \ldots, k\}^n$ taken from the appropriate element of the type class. Define matrices $K_{\to S} \in \mathbb{Z}^{|\mathcal{E}_{\text{out}}(S)|\times r}$ and $K_{F\to j} \in \mathbb{Z}^{|\mathcal{E}_{\text{in}}(j)|\times|F|}$ for $F \subset E$ and $j \in V$ analogously to those above. The following proposition gives the rate for the Polytope Code.

Proposition 4: Fix $\mathcal{F} \subset 2^E$. Suppose the adversary may control any set of links $F \in \mathcal{F}$. For a Polytope Code given by

[8] This is the most general adversary problem. Particularizing it to any set of s nodes requires setting $\mathcal{F} = \big\{\bigcup_{i=1}^{s}\mathcal{E}_{\text{out}}(v_i) : v_1, \ldots, v_s \in V \setminus \{S, D\}\big\}$.

[9] We distinguish between the rate R and the dimension r of the message vector. For a finite field code, these are identical, but for a Polytope Code, we can only achieve rate r for large blocklength n and polytope size k.

coding operators ($K_{\to e} : e \in \mathcal{E}_{\text{out}}(S)$) and ($K_{e\to f} : \text{head}(e) = \text{tail}(f)$), for any $\epsilon > 0$, there exist sufficiently large n and k such that the rate satisfies $R \geq r - \epsilon$. The message can be decoded reliably if and only if

$$r \leq \min_{F_1,F_2\in\mathcal{F}} \text{rank}([K_{S\to D}K_{\to S}\ K_{F_1\to D}\ K_{F_2\to D}]) - \text{rank}([K_{F_1\to D}\ K_{F_2\to D}]). \quad (167)$$

Proof: Using Proposition 1 and the fact that the type class is close to $2^{nH(W)}$ for sufficiently large n gives that the code achieves rate $r - \epsilon$. The reliability condition (167) follows from an identical argument as Proposition 3. ■

C. Remarks on Propositions 3 and 4

The conditions (153) and (167) are identical except that in (153) the matrices are over the finite field $\mathbb{F}$, and in (167) the matrices are over the integers. Hence, whether a Polytope Code can perform as well as a given linear code depends on whether there exists integer-valued matrices with the same rank as certain $\mathbb{F}$-valued matrices. This is a difficult question to answer in general, but we provide some commentary.

Suppose for now that $\mathbb{F}$ has prime order q (*i.e.* arithmetic over $\mathbb{F}$ is simply modulo q), and consider a matrix A with values in $\mathbb{F}$. Let $\tilde{A}$ be any integer-valued matrix such that $\tilde{A}_{i,j}$ is congruent to $A_{i,j}$ modulo q. We claim that $\text{rank}(\tilde{A}) \geq \text{rank}(A)$. Let $\mathcal{S}$ be a maximal set of linearly independent columns of $\tilde{A}$; hence $|\mathcal{S}| = \text{rank}(\tilde{A})$. Let $\tilde{A}_{\mathcal{S}}$ be the matrix composed of these columns, and $\tilde{A}_{\mathcal{S}^c}$ be the matrix composed of the remaining columns. (Let $A_{\mathcal{S}}$ and $A_{\mathcal{S}^c}$ denote the associated columns from A.) The columns of $\tilde{A}_{\mathcal{S}^c}$ are linearly dependent on the columns of $\tilde{A}_{\mathcal{S}}$; *i.e.* there exists an integer-valued matrix $\tilde{G}$ and an integer $\tilde{v}$ such that $\tilde{A}_{\mathcal{S}^c} = \tilde{A}_{\mathcal{S}}\tilde{G}v^{-1}$. If we take G to be $\mathbb{F}$-valued where $G_{i,j}$ is $\tilde{G}_{i,j}$ modulo q, and v to be $\tilde{v}$ modulo q, then

$$A_{\mathcal{S}^c}v - A_{\mathcal{S}}G = (\tilde{A}_{\mathcal{S}^c}\tilde{v} - \tilde{A}_{\mathcal{S}}\tilde{G} \bmod q) = (0 \bmod q) = 0. \quad (168)$$

Hence $A_{\mathcal{S}^c} = A_{\mathcal{S}}Fv^{-1}$, so there are at most $|\mathcal{S}| = \text{rank}(\tilde{A})$ linearly independent columns of A.

Now suppose $\mathbb{F}$ has order q^d for prime q and $d > 1$. Such a field is defined by an $d-1$st-order polynomial $p(z)$ over the variable z. Elements of the field are given by $d-1$st-order polynomials, and arithmetic on $\mathbb{F}$ is described by the relations $q = 0$ and $z^d + p(z) = 0$. The most straightforward way to convert a linear code over $\mathbb{F}$ into a Polytope Code is to take polytopes not over the integers but over the ring $\mathbb{Z}[\alpha]$ (*i.e.* the ring composed of the integers and real number α, and closed under addition and multiplication), where α is a real number satisfying $\alpha^n + p(\alpha) = 0$. In particular, we can replace (34) with a polytope of the form

$$\{\mathbf{x} = \mathbf{c}_0 + \mathbf{c}_1\alpha + \cdots + \mathbf{c}_{d-1}\alpha^{d-1} : F\mathbf{x} = 0, \mathbf{c}_i \in \mathbb{Z}^m, \|\mathbf{c}_i\|_\infty \leq k \text{ for } i = 0,\ldots,d-1\}. \quad (169)$$

No basic properties of the Polytope Code change: since this set is contained in $\mathbb{R}^m$, Theorem 4 applies. Given an $\mathbb{F}$-valued matrix A, let $\tilde{A}$ be any $\mathbb{Z}[\alpha]$-valued matrix such that if

$$A_{i,j} = c_0 + c_1 z + c_2 z^2 + \ldots + c_{n-1}z^{n-1} \quad (170)$$

where the coefficients c_l are elements of the field of order q, then

$$\tilde{A}_{i,j} = \tilde{c}_0 + \tilde{c}_1\alpha + \tilde{c}_2\alpha^2 + \ldots + \tilde{c}_{n-1}\alpha^{n-1} \quad (171)$$

where $\tilde{c}_l$ is congruent to c_l modulo q. A similar argument as above can be used to show that $\text{rank}(\tilde{A}) \geq \text{rank}(A)$.

The above analysis shows that given a finite field code defined by coding operators $K_{\to e}, K_{e\to f}$, we may construct a Polytope Code defined by $\tilde{K}_{\to e}, \tilde{K}_{e\to f}$ using the transformation described above, and each rank in (167) will not decrease relative to (153). However, since the second terms in (153) and (167) are negative, it is possible that the resulting Polytope Code rate from (167) will be strictly less than the linear code rate from (153). For the no-adversary problem, the negative term vanishes, and so in that setting a Polytope Code always does at least as well as a linear code. In fact, it is conceivable that there are multi-source no-adversary problems for which Polytope Codes outperform linear codes. However, we believe that in many useful cases, the above transformations give $\text{rank}(\tilde{A}) = \text{rank}(A)$. Indeed, $\text{rank}(\tilde{A}) > \text{rank}(A)$ is an event associated with finite fields of small order, while typically linear network codes require field orders above a certain threshold to achieve high rates. In this more generic scenario where the transformation does not alter any rank, the rate achieved by the Polytope Code is identical to that of the linear code, with or without adversaries.

XI. Looseness of the Cut-Set Bound

So far, the only available upper bound on achievable rates has been the cut-set bound. We have conjectured that for planar graphs this bound is tight, but that still leaves open the question of whether there is a tighter upper bound for non-planar graphs. It was conjectured in [10] that there is such a tighter bound, and here we prove this conjecture to be true. We do this in two parts. First, consider the problem that the traitor nodes are constrained to be only from a certain subset of nodes. That is, a special subset of nodes are designated as potential traitors, and the code need only guard against adversarial control of any s of those nodes. We refer to this as the *limited-node* problem.[10] Certainly the limited-node problem subsumes the all-node problem, since we may simply take the set of potential traitors to be all nodes. Furthermore, it subsumes the unequal-edges problem studied in [10], because given an instance of the unequal-edge problem, an equivalent all-node problem can be constructed as follows: create a new network with every edge replaced by a pair of edges of equal capacity with a node between them. Then limit the traitors to be only these interior nodes.

We will show in Section XI-A that the all-node problem actually subsumes the limited-node problem, and therefore also the unequal-edge problem. Then in Section XI-B we give an example of a limited-node network for which there is an active upper bound on capacity other than the cut-set. This proves that, even for the all-node problem, the cut-set bound is not tight in general. Transforming the example in

[10]We studied an instance of the limited-node problem on the Caterpillar network in Sec. VI.

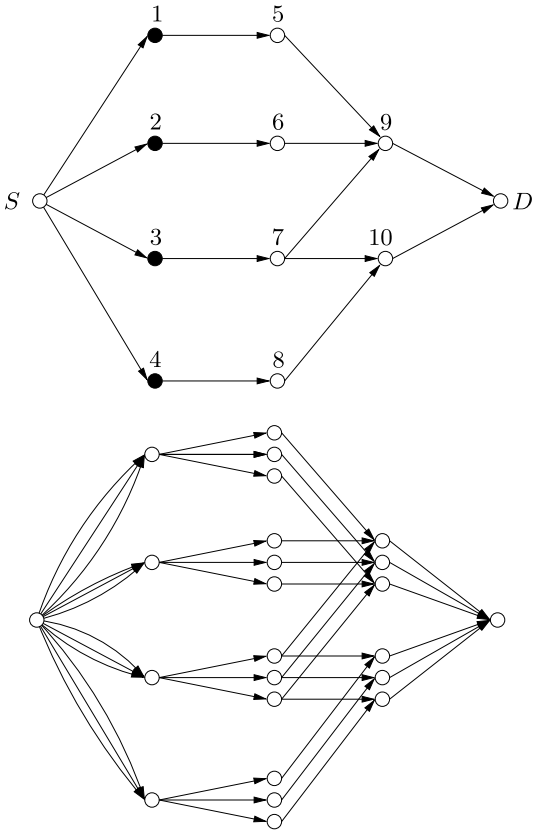

Fig. 12. The Ant network. Its capacity is strictly less than the cut-set bound. The limited-node network is shown above, and the equivalent all-node problem with 3 copies is shown below.

Section XI-B into an unequal-edge problem is not hard; this therefore confirms the conjecture in [10].

A. Equivalence of Limited-Node and All-Node

Let (V, E) be a network under a limited-node adversarial attack, where there may be at most s traitors constrained to be in $U \subseteq V$, and let C be its capacity. We construct a sequence of all-node problems, such that finding the capacity of these problems is enough to find that of the original limited-node problem. Let $(V^{(M)}, E^{(M)})$ be a network as follows. First make M copies of (V, E). That is, for each $i \in V$, put $i^{(1)}, \ldots, i^{(M)}$ into $V^{(M)}$, and for each edge $(i, j) \in E$, put $(i^{(1)}, j^{(1)}), \ldots, (i^{(M)}, j^{(M)})$ into $E^{(M)}$. Then, for each $i \in U$, merge $i^{(1)}, \ldots, i^{(M)}$ into a single node i^*, transferring all edges that were previously connected to any of $i^{(1)}, \ldots, i^{(M)}$ to i^*. Let $C^{(M)}$ be the all-node capacity of $(V^{(M)}, E^{(M)})$ with s traitors. This construction is illustrated in Fig. 12, where we show a limited-node network (V, E) and the all-node network $(V^{(M)}, E^{(M)})$ with $M = 3$. For large M, the all-node problem will be such that for any $i \notin U$, the adversary has no reason to control one of the respective nodes because it commands such a small fraction of the information flow through the network. That is, we may assume that the traitors will only ever be nodes in U. This is stated explicitly in the following theorem.

Theorem 5: For any M, $C^{(M)}$ is related to C by

$$\frac{1}{M}C^{(M)} \leq C \leq \frac{1}{M-2s}C^{(M)}. \tag{172}$$

Moreover,

$$C = \lim_{M\to\infty} \frac{1}{M}C^{(M)} \tag{173}$$

and if $C^{(M)}$ can be computed to arbitrary precision for any M in finite time, then so can C.

Proof: We first show that $\frac{1}{M}C^{(M)} \leq C$. Take any code on $(V^{(M)}, E^{(M)})$ achieving rate R when any s nodes may be traitors. We use this to construct a code on (V, E), achieving rate R/M when any s nodes in U may be traitors. We do this by first increasing the blocklength by a factor of M, but maintaining the same number of messages, thereby reducing the achieved rate by a factor of M. Now, since each edge in (V, E) corresponds to M edges in $(V^{(M)}, E^{(M)})$, we may place every value transmitted on an edge in the $(V^{(M)}, E^{(M)})$ code to be transmitted on the equivalent edge in the (V, E) code. That is, all functions executed by $i^{(1)}, \ldots, i^{(M)}$ are now executed by i. The original code could certainly handle any s traitor nodes in U. Hence the new code can handle any s nodes in U, since the actions performed by these nodes have not changed from $(V^{(M)}, E^{(M)})$ to (V, E). Therefore, the new code on (V, E) achieves rate R/M for the limited-node problem.

Now we show that $C \leq \frac{1}{M-2s}C^{(M)}$. Take any code on (V, E) achieving rate R. We will construct a code on $(V^{(M)}, E^{(M)})$ achieving rate $(M - 2s)R$. This direction is slightly more difficult because the new code needs to handle a greater variety of traitors. The code on $(V^{(M)}, E^{(M)})$ is composed of an outer code and M copies of the (V, E) code running in parallel. The outer code is an $(M, M - 2s)$ MDS code with coded output values $w_1, \ldots, w_M$. These values form the messages for the inner codes. Since we use an MDS code, if $w_1, \ldots, w_M$ are reconstructed at the destination such that no more than s are corrupted, the errors can be entirely corrected. The jth copy of the (V, E) code is performed by i^* for $i \in U$, and by $i^{(j)}$ for $i \notin U$. That is, nodes in U are each involved in all M copies of the code, while nodes not in U are involved in only one. Because the (V, E) code is assumed to defeat any attack on only nodes in U, if for some j, no nodes $i^{(j)}$ for $i \notin U$ are traitors, then the message w_j will be recovered correctly at the destination. Therefore, one of the w_j could be corrupted only if $i^{(j)}$ is a traitor for some $i \notin U$. Since there are at most s traitors, at most s of the $w_1, \ldots, w_M$ will be corrupted, so the outer code corrects the errors.

From (172), (173) is immediate. We can easily identify M large enough to compute C to any desired precision. ∎

The significance of Theorem 5 is that if we could calculate the capacity of any all-node problem, we could use (172) to calculate the capacity of any limited-node problem.[11] Furthermore, it is easy to see that for large M the cut-set bound of $(V^{(M)}, E^{(M)})$ is simply M times the cut-set bound of (V, E). Hence a limited-node example with capacity less than the cut-set bound—such as the one in Section XI-B—leads directly to an all-node example with capacity less than the cut-set bound.

[11] In [18], which studied the eavesdropping problem, a similar construction was used to convert a problem with restricted wiretap sets into one with nonuniform wiretap sets, by splitting a link into several parallel links each with smaller capacity.

B. Capacity Less Than Cut-Set: The Ant Network

Consider the Ant network shown in Figure 12. There is at most one traitor, but it may only be one of nodes 1–4. The cut-set bound is easily seen to be 2, but in fact the capacity is 1.5.[12]

Proposition 5: The capacity of the Ant network is no more than 1.5.

Proof: Suppose we are given a code achieving rate R for the Ant network. For $i = 1, 2, 3, 4$, let X_i be the random variable representing the value on the output edge of node i. Let Y be the value on edge $(9, D)$ and let Z be the value on $(10, D)$. Let p be the honest distribution on these variables, and define the following alternative distributions:

$$q_3 := p(x_1x_2x_4)p(x_3)p(y|x_1x_2x_3)p(z|x_3x_4), \quad (174)$$

$$q_4 := p(x_1x_2x_3)p(x_4)p(y|x_1x_2x_3)p(z|x_3x_4). \quad (175)$$

Suppose node 3 is the traitor. It may generate a completely independent version of X_3 according to $p(x_3)$, and send it along edge $(3, 7)$. All other nodes behave honestly, so this action results in the distribution q_3. In order for the destination to decode successfully, information about the message must get through from the honest edges at the start of the network, X_1, X_2, X_4, to what is received at the destination, Y, Z. Hence

$$nR \leq I_{q_3}(X_1X_2X_4; YZ) \quad (176)$$

where n is the blocklength (*i.e.* each edge carries a value in $[2^n]$). We apply standard information-theoretic relations to find

$$nR \leq I_{q_3}(X_1X_2X_4; Z) + I_{q_3}(X_1X_2X_4; Y|Z) \quad (177)$$

$$\leq I_{q_3}(X_4; Z) + I(X_1X_2; Z|X_4) + n \quad (178)$$

$$= I_{q_3}(X_4; Z) + n \quad (179)$$

where in (178) we have used that the capacity of $(9, D)$ is 1, and in (179) that $X_1X_2 - X_4 - Z$ is a Markov chain according to q_3. Using a similar argument in which node 4 is the traitor and it acts in a way to produce q_4, we may write

$$R \leq I_{q_4}(X_3; Z) + n. \quad (180)$$

Note that

$$q_3(x_3x_4z) = q_4(x_3x_4z). \quad (181)$$

In particular, the mutual informations in (179) and (180) can both be written with respect to the same distribution. Therefore,

$$2nR \leq I_{q_3}(X_4; Z) + I_{q_3}(X_3; Z) + 2n \quad (182)$$

$$= I_{q_3}(X_3X_4; Z) + I_{q_3}(X_3; X_4) - I_{q_3}(X_3; X_4|Z) + 2n \quad (183)$$

$$\leq I_{q_3}(X_3X_4; Z) + 2n \quad (184)$$

$$\leq 3n \quad (185)$$

where (184) follows from the positivity of conditional mutual information and that X_3, X_4 are independent according to q_3, and (185) follows because the capacity of $(10, D)$ is 1. It follows from (185) that $R \leq 1.5$. ∎

[12]We only prove the converse for this example. Achievability is left as an exercise for the reader.

The key property of the attack considered in the proof of Proposition 5 is (181): that the two distributions representing the two possible situations were equal, so it is impossible to determine at node 10 whether node 3 or 4 is the traitor. This condition is analogous to the notion of *symmetrizability* for the arbitrarily varying channel [19]. A channel is symmetrizable if there exists a distribution on the channel state that makes two different channel inputs indistinguishable, and hence it is impossible to decode with high probability. This is precisely the condition in (181).

Observe that all inequalities used in Proposition 5 were so-called Shannon-type information inequalities (*i.e.* based on the submodularity of entropy). For the no-adversary problem, there is a straightforward procedure to write down all the Shannon-type inequalities relevant to a particular network coding problem, which in principle can be used to find an upper bound [20]. This upper bound is more general than any cut-set upper bound, and in some multi-source problems—for example in [21]—it yields a tighter bound than any cut-set bound. The Ant network illustrates that a similar phenomenon occurs in the adversarial problem even for unicast. As the adversary problem seems to have much in common with the multi-source no-adversary problem, it would be worthwhile to formulate the tightest possible upper bound using only Shannon-type inequalities. However, it is yet unclear what the "complete" list of Shannon type inequalities would be for the adversary problem. The above argument demonstrates one method of finding them, but whether there are fundamentally different methods to find inequalities that could still be called Shannon-type, or even how to compile all inequalities using this method, is unclear. Moreover, it was shown in [22] that in the no-adversary problem, there can be active non-Shannon-type inequalities. It is therefore conceivable that with adversaries, non-Shannon-type inequalities could be relevant even for a single source.

XII. Conclusion

The main contribution of this paper has been to introduce the theory of Polytope Codes. As far as we know, they are the best known coding strategy to defeat generalized adversary attacks on network coding. However, it remains difficult to calculate the best possible rate they can achieve for a given network. We have proved that they achieve the cut-set bound, and hence the capacity, for some networks, including a class of planar networks, and we conjecture that this holds for all planar networks. One would obviously hope to find the capacity of all networks. We have shown that achieving the cut-set bound is not always possible, meaning there remains significant work to do on upper bounds as well as achievable schemes. Whether Polytope Codes can achieve capacity on all networks remains an important open question.

Appendix A
Proof of Bound on Linear Capacity for the Cockroach Network

We show that no linear code for the Cockroach network can achieve a rate higher than 4/3. Fix any linear code: in the

notation of Sec. X, this means fixing $K_{\to e}$ for $e \in \mathcal{E}_{\text{out}}(S)$ and $K_{e\to f}$ for $\text{head}(e) = \text{tail}(f)$. While Proposition 3 applies to scalar linear codes—that is, each edge holds a value from a finite field $\mathbb{F}$—it is easy to generalize it to vector linear codes, in which each edge holds a value from the finite vector field $\mathbb{F}^n$. We simply reinterpret (153) where each linear operator acts on $\mathbb{F}^n$ instead of $\mathbb{F}$. Then particularizing (153) to the case of a single traitor node gives

$$nR \le \min_{i,j\in V\setminus S,D} \text{rank}([K_{S\to D}K_{\to S}\ K_{i\to D}\ K_{j\to D}]) - \text{rank}([K_{i\to D}\ K_{j\to D}]). \quad (186)$$

We may loosen (186) to

$$nR \le \min_{i,j\in V\setminus S,D} \text{rank}([K_{S\to D}\ K_{i\to D}\ K_{j\to D}]) - \text{rank}([K_{i\to D}\ K_{j\to D}]). \quad (187)$$

Define k_1 to be the dimension of the subspace of $\text{span}(K_{(1,4)\to(4,D)})$ orthogonal to $\text{span}(K_{(2,4)\to(4,D)})$, and *vice versa* for k_2. That is,

$$k_1 := \text{rank}([K_{(1,4)\to(4,D)}\ K_{(2,4)\to(4,D)}]) - \text{rank}(K_{(2,4)\to(4,D)}), \quad (188)$$

$$k_2 := \text{rank}([K_{(1,4)\to(4,D)}\ K_{(2,4)\to(4,D)}]) - \text{rank}(K_{(1,4)\to(4,D)}). \quad (189)$$

Note that $k_1 + k_2 \le n$. Similarly define

$$l_1 := \text{rank}([K_{(2,5)\to(5,D)}\ K_{(3,5)\to(5,D)}]) - \text{rank}(K_{(3,5)\to(5,D)}), \quad (190)$$

$$l_2 := \text{rank}([K_{(2,5)\to(5,D)}\ K_{(3,5)\to(5,D)}]) - \text{rank}(K_{(2,5)\to(5,D)}). \quad (191)$$

Also $l_1 + l_2 \le n$. Particularizing (187) with $i = 2$ and $j = 3$ gives

$$\begin{aligned} nR &\le \text{rank}([K_{S\to D}\ K_{1\to D}\ K_{2\to D}]) - \text{rank}([K_{1\to D}\ K_{2\to D}])) \\ &\le \text{rank}(K_{S\to(1,D)}) + \text{rank}([K_{(1,4)\to(4,D)}\ K_{(2,4)\to(4,D)}]) - \text{rank}(K_{(2,4)\to(4,D)}) \\ &\le n + k_1. \end{aligned} \quad (192)$$

A similar analysis with $i = 1$ and $j = 2$ gives

$$nR \le l_2 + n. \quad (193)$$

Finally, setting $i = 1$ and $j = 3$ gives

$$\begin{aligned} nR &\le \text{rank}([K_{S\to D}\ K_{1\to D}\ K_{3\to D}]) - \text{rank}([K_{1\to D}\ K_{3\to D}]) \\ &\le \text{rank}([K_{(1,4)\to(4,D)}\ K_{(2,4)\to(4,D)}]) - \text{rank}(K_{(1,4)\to(4,D)}) \\ &\quad + \text{rank}([K_{(2,5)\to(5,D)}\ K_{(3,5)\to(5,D)}]) - \text{rank}(K_{(3,5)\to(5,D)}) \\ &= k_2 + l_1. \end{aligned} \quad (194)$$

Summing (192), (193), and (194) gives

$$3nR \le 2n + (k_1 + k_2) + (l_1 + l_2) \le 4n. \quad (195)$$

Hence $R \le 4/3$.

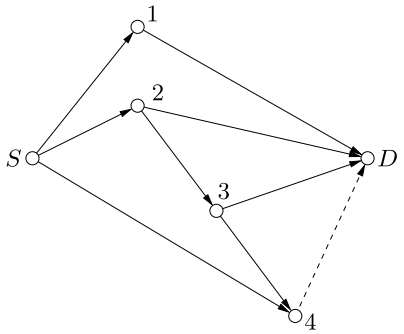

Fig. 13. The Beetle Network. All edges have unit-capacity except the dashed edge, which has zero capacity.

APPENDIX B
ILLUSTRATING THE EFFECT OF A BACKWARD EDGE: THE BEETLE NETWORK

The Beetle Network, shown in Figure 13, has two interesting properties in the presence of one traitor node. First, there is a cut with a backwards edge for which the value of the right hand side of (1) is strictly less than capacity. This illustrates the need for the condition in Theorem 1 that the cut has no backward edges. Second, the Beetle Network has a zero capacity edge[13] with a positive effect on the capacity. That is, the capacity of this network is 1, but if the zero-capacity edge $(4, D)$ were removed, then the capacity would drop to 0, as can easily be verified by Theorem 1. The reason for this is that, as we have seen, comparison operations can increase capacity, so we can use the zero-capacity edge to hold a comparison bit.

We may apply Theorem 1 with $A = \{S, 1, 2, 3, 4\}$ and $T = \{1, 2\}$ to conclude that the capacity is no more than 1. We will shortly present a code to achieve rate 1. Now consider the cut $A = \{S, 1, 2, 4\}$. For this cut $(3, 4)$ is a backwards edge, so we cannot apply Theorem 1. Note that if we set $T = \{1, 2\}$, the right hand side of (1) would evaluate to 0, strictly less than capacity.

We now present a simple linear code with a comparison for the Beetle Network achieving rate 1. Each unit-capacity edge carries a copy of the message w. That is, the source sends w along all three of its output links, and nodes 1, 2, and 3 each receive one copy of w and forward it along all of their output links. Node 4 receives a copy of w from the source and and one from node 3. It compares them and sends to the destination one of the symbols $=$ or $\neq$ depending on whether the two copies agreed. Because w may be a vector of arbitrary length, sending this single bit along edge $(4, D)$ takes arbitrarily little rate, so we do not exceed the edge capacity.

The decoding procedure is as follows. Let w_1, w_2, and w_3 be the values of w received at the destination from nodes 1, 2, and 3 respectively. There are two cases:

1) $w_2 \neq w_3$ or node 4 sends $\neq$. Then the destination decodes from w_1.

[13] A zero capacity edge is one with arbitrarily small capacity. Although elsewhere we have assumed all edges have unit capacity, here it is more convenient to consider an edge with zero capacity. It is equivalent to replace the zero capacity edge with a unit-capacity edge, and replace all other edges with an arbitrary number of parallel unit-capacity edges.

2) $w_2 = w_3$ and node 4 sends $=$. Then the destination decodes from w_2.

Now we show that the destination never makes an error. In case (1), the traitor can only by node 2, 3, or 4. Hence w_1 is trustworthy. Now consider case (2). Certainly if the traitor is either node 1 or 4, then $w_2 = w_3 = w$. If the traitor is node 3, then $w_2 = w$, so we still decode correctly. If the traitor is node 2, then it must send the same value of w to both the destination and node 3, because node 3 simply forwards its copy to the destination, and we know $w_2 = w_3$. Furthermore, this value of w must be the true one, because otherwise node 4 would observe that the copy sent along edge $(3, 4)$ is different from that sent from the source, so it would transmit $\neq$ to the destination. Since it did not, node 2 cannot have altered any of its output values. Therefore the destination always decodes correctly.

Appendix C
Tighter Cut-Set Upper Bound

Theorem 6: Consider a cut $A \subseteq V$ with $S \in A$ and $D \notin A$. Let E_A be the set of edges that cross the cut. For two not necessarily disjoint sets of possible traitors T_1, T_2, let E_1 and E_2 be the subset of edges in E_A that originate at nodes in T_1 and T_2 respectively. Let $\tilde{E}$ be the set of edges in $E_1 \cap E_2$ in addition to all edges $e \in E_1 \cup E_2$ for which there is no path that flows through e followed by any edge in $E_A \setminus E_1 \setminus E_2$. The following upper bound holds on the capacity of the network:

$$C \leq \sum_{e \in E_A \setminus \tilde{E}} c_e. \tag{196}$$

Proof: Suppose (1) were not true for some A, T_1, and T_2. Then there would exist a code achieving a rate R such that

$$R > \sum_{e \in E_A \setminus \tilde{E}} c_e. \tag{197}$$

We will consider two possibilities, one when T_1 are the traitors and they alter the values on $E_1 \cap \tilde{E}$, and one when T_2 are the traitors and they alter the values on $E_2 \cap \tilde{E}$. We will show that by (197), it is possible for the traitors to act in such a way that even though the messages at the source are different, all values sent across the cut are the same; therefore the destination will not be able to distinguish all messages. Note that traitors in T_1 or T_2 will only corrupt values on edges in $\tilde{E}$; that is, those edges controlled by either set of traitors, or those that could not influence $E_A \setminus E_1 \setminus E_2$.

Fix a value z representing one possible set of values that may be placed on the edges $E_1 \cap E_2$. By definition, no edges in $\tilde{E} \setminus (E_1 \cap E_2)$ are upstream of edges in $E_A \setminus \tilde{E}$. Since the traitors act honestly on all edges not in $\tilde{E}$, given z, the values on $E_A \setminus \tilde{E}$ are a function of the message, so by (197), there exist two messages w_a and w_b that cannot be distinguished just from $E_A \setminus \tilde{E}$. Call y the set of values on these edges under w_a or w_b.

Choose a coding order on the edges in $\tilde{E} \setminus (E_1 \cap E_2)$ written as

$$(l_1, l_2, \ldots, l_K) \tag{198}$$

where $K = |\tilde{E} \setminus (E_1 \cap E_2)|$, such that if there is a path through l_i followed by l_j, then $i < j$. Observe that $\tilde{E} \setminus (E_1 \cap E_2)$ can be divided into $\tilde{E} \setminus E_2$ and $\tilde{E} \setminus E_1$, and therefore the order in (198) must alternate between edges in the two sets. Maintaining the order in (198), we may group the edges by the two sets, rewriting (198) as

$$(\mathcal{U}_1, \mathcal{V}_1, \mathcal{U}_2, \mathcal{V}_2, \ldots, \mathcal{U}_{K'}, \mathcal{V}_{K'}) \tag{199}$$

where $\mathcal{U}_i \subset \tilde{E} \setminus E_2$ and $\mathcal{V}_i \subset \tilde{E} \setminus E_2$, and K' is the number of times the edges in (198) alternate between the two sets. Note that $\mathcal{U}_1$ or $\mathcal{V}_{K'}$ may be empty.

We now construct the manner in which the two possible sets of traitors, T_1 or T_2, may cause w_a and w_b to become indistinguishable. Suppose w_a is the message, T_1 are the traitors, they place z on $E_1 \cap E_2$, but behave honestly on all other edges. We denote the values on all edges crossing the cut as

$$(z, y, u_1^{(1)}, v_1^{(1)}, u_2^{(1)}, v_2^{(1)}, \ldots, u_{K'}^{(1)}, v_{K'}^{(1)}) \tag{200}$$

where z represents the values on $E_1 \cap E_2$, y the values on $E_A \setminus \tilde{E}$, and $u_i^{(1)}$ and $v_i^{(1)}$ are the values placed on $\mathcal{U}_i$ and $\mathcal{V}_i$ respectively. Note that only the u_i values are directly adjustable by the traitors, but they may affect elements later in the sequence.

Alternatively, if w_a is the message, T_2 are the traitors, and they place z on $E_1 \cap E_2$, but behave honestly elsewhere, the values across the cut are denoted

$$(z, y, u_1^{(2)}, v_1^{(2)}, u_2^{(2)}, v_2^{(2)}, \ldots, v_{K'}^{(2)}). \tag{201}$$

Here, only the v_i values may be changed directly by the traitors.

In the two scenarios, the traitors may alter their output values so that (200) and (201) are transformed to become identical. This can be done as follows. In (200), the traitors may replace $u_1^{(1)}$ with $u_1^{(2)}$. Downstream edges are either controlled by honest nodes, or they are controlled by traitors that continue, for now, to behave honestly. Hence, this change may affect the later edges in the sequence, but they do so in a way determined only by the code. This results in a set of values denoted by

$$(z, y, u_1^{(2)}, v_1^{(3)}, u_2^{(3)}, v_2^{(3)}, \ldots, u_{K'}^{(3)}, v_{K'}^{(3)}). \tag{202}$$

In (201), the traitors may now replace $v_1^{(2)}$ with $v_1^{(3)}$, resulting in

$$(z, y, u_1^{(2)}, v_1^{(3)}, u_2^{(4)}, v_2^{(4)}, \ldots, v_{K'}^{(4)}). \tag{203}$$

We may now return to (202) and replace $u_2^{(3)}$ with $u_2^{(4)}$, further changing downstream values. Continuing this process will cause the two sequences to become identical, thereby making w_a and w_b indistinguishable at the destination. ■

Appendix D
Proof of Lemma 1

Assume (V, E) satisfies 2-OUT, IN-OUT, and PLANAR, and the cut-set bound with $s = 1$ is $M - 2$. By Lemma 2, we may assume without loss of generality that (V, E) also satisfies 2-IN. To prove that the network satisfies PATHS,

we must construct non-destination paths $\mathbf{p}_v$ for each 2-to-1 node v satisfying (3). We show how to do so in Algorithm 1. Before proving algorithm correctness, we prove a lemma particularizing the cut-set bound of Theorem 1 to make it more directly applicable. Then we prove several claims resulting from the cut-set bound and the planarity of the network. Finally we prove algorithm correctness.

Lemma 7: Consider a network (V, E) satisfying IN-OUT. Let $s = 1$. For $i, j \in V$, let $d_{i,j}$ be the sum of $|\mathcal{E}_{\text{in}}(k)| - |\mathcal{E}_{\text{out}}(k)|$ for all nodes k reachable from either i or j, not including i or j.[14] Let c_i be the total number of output edges from node i, and let e_i be the number of output edges from node i that go directly to the destination. For any distinct pair of nodes $i_1, i_2 \in V$,

$$C \leq M - e_{i_1} - e_{i_2}. \tag{204}$$

Moreover, if there is no path between i_1 and i_2,

$$C \leq M + d_{i_1,i_2} - c_{i_1} - c_{i_2}. \tag{205}$$

Proof: Applying Theorem 1 with $A = V \setminus \{D\}$, $U = \{i_1, i_2\}$ immediately gives (204). To prove (205), we apply Theorem 1 with $U = \{i_1, i_2\}$, and

$$A = \{k \in V : k \text{ is not reachable from } i_1 \text{ or } i_2\} \cup \{i_1, i_2\}. \tag{206}$$

Observe that there are no backward edges for the cut A, because for any edge (j, k) with $j \in A^c$, j is reachable from i_1 or i_2, and hence k is as well; thus $k \in A^c$. Therefore we may apply Theorem 1. Since all output neighbors of i_1 and i_2 are not in A, each output edge of i_1 and i_2 crosses the cut. Hence (1) becomes

$$C \leq |\{e \in E : \text{tail}(e) \in A,\ \text{head}(e) \notin A\}| - c_{i_1} - c_{i_2}. \tag{207}$$

By IN-OUT, no node in the network has more output edges than input edges, so the difference between the first term in (207)—the number of edges crossing the cut—and M is exactly the sum of $|\mathcal{E}_{\text{in}}(k)| - |\mathcal{E}_{\text{out}}(k)|$ for all $k \in A^c$. Hence

$$|\{e \in E : \text{head}(e) \in A,\ \text{tail}(e) \notin A\}| - M = d_{i_1,i_2}. \tag{208}$$

Combining (207) with (208) gives (205). ∎

By assumption, the right hand sides of (204) and (205) are never less than $M - 2$. The following three claims result from this fact.

Claim 1: Every node in $\mathcal{N}_{\text{in}}(D)$ has exactly one output edge to D.

Proof: Since the cut-set bound is $M - 2$, by (204) we have that $e_{i_1} + e_{i_2} \leq 2$ for all $i_1, i_2 \in \mathcal{N}_{\text{in}}(D)$. Certainly any node in $\mathcal{N}_{\text{in}}(D)$ has at least one output edge to D, so $e_i \geq 1$ for all $i \in \mathcal{N}_{\text{in}}(D)$. Thus $e_i = 1$ for all $\mathcal{N}_{\text{in}}(D)$. ∎

Claim 2: Every node can reach at least one 2-to-1 node in $\mathcal{N}_{\text{in}}(D)$.

Proof: Given any node i, we may follow any path from i, avoiding D whenever possible. This eventually leads to a node i_1 all of whose output edges lead directly to D. By Claim 1, $e_{i_1} = 1$, so i_1 must have exactly one output edge; *i.e.* it is 2-to-1. ∎

[14]For any network satisfying IN-OUT, $|\mathcal{E}_{\text{in}}(k)| \geq |\mathcal{E}_{\text{out}}(k)|$, so the contribution from each node is nonnegative.

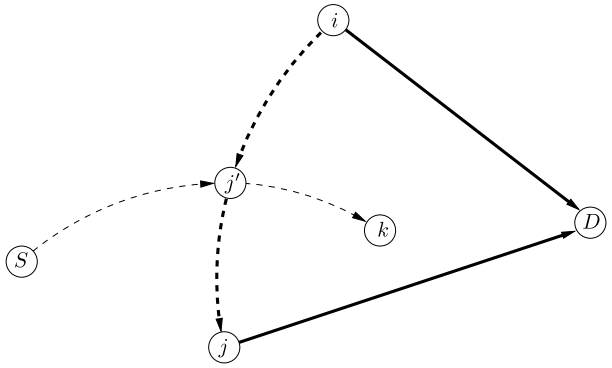

Fig. 14. A diagram of the planar embedding being used to prove Claim 4: in particular, that a node $k \in \mathcal{N}_{\text{in}}(D)$ on the interior of $\mathcal{C}_{i,j}$ is reachable from i. Solid lines are single edges; dashed lines represent paths made up of any number of edges (including zero). Thick lines correspond to edges in $\mathcal{C}_{i,j}$.

Claim 3: For each 2-to-1 node v, there exists a path $\bar{\Gamma}_v$ containing just the nodes $\Gamma_v \cup \{v\}$.

Proof: Recall that Γ_v is the set of nodes for which v is the only reachable node with more inputs than outputs. By 2-IN, the only nodes with more inputs than outputs are 2-to-1 nodes. Hence Γ_v is the set of nodes for which v is the only reachable 2-to-1 node. Other than v, all nodes in Γ_v are 2-to-2. Take any distinct $i_1, i_2 \in \Gamma_v \setminus \{v\}$. We have $d_{i_1,i_2} = 1$ and $c_{i_1} = c_{i_2} = 2$. If there were no path between i_1 and i_2 (in either direction), then we could apply (205), which would give $C \leq M - 3$, contradicting our assumption that the cut-set bound is $M - 2$. Hence there is a path between any two nodes in $\Gamma_v \setminus \{v\}$. Moreover, all nodes in Γ_v can reach v. Therefore there is a path $\bar{\Gamma}_v$ where $\text{tail}(\bar{\Gamma}_v) \in \Gamma_v$, $\text{head}(\bar{\Gamma}_v) = v$, and $\Gamma_v \subset \bar{\Gamma}_v$. Additionally, if any node $i \in \bar{\Gamma}_v$ were not in Γ_v, then i could reach a 2-to-1 node other than v; hence the previous node in $\bar{\Gamma}_v$ would also not be in Γ_v. This contradicts the construction that $\text{tail}(\bar{\Gamma}_v) \in \Gamma_v$. Thus $\bar{\Gamma}_v$ contains precisely the nodes $\Gamma_v \cup v$. ∎

Since (V, E) satisfies PLANAR, we may construct an embedding of the graph (V, E) in the plane such that S is on the exterior face. Such an embedding always exists for a planar graph [23]. For any set of edges forming an *undirected cycle*—that is, edges constituting a cycle on the underlying undirected graph—all nodes in the network are divided into three groups: those on the cycle, those in its interior, and those in its exterior. Any path from the exterior to the interior must cross the cycle at some node. We will use this principle several times in the proof, beginning with the following claim.

Claim 4: Suppose $i, j \in \mathcal{N}_{\text{in}}(D)$ where i can reach j. Let $\mathcal{C}_{i,j}$ be any undirected cycle composed of a path from i to j, in addition to the edges (i, D) and (j, D). (See Fig. 14.) If a node k is not exterior to $\mathcal{C}_{i,j}$, then it can be reached from i.

Proof: If k is on the cycle $\mathcal{C}_{i,j}$, then it must be on the path from i to j, so it can be reached by i. Now suppose k is in the interior of $\mathcal{C}_{i,j}$. Since S is in the exterior face of the graph, it must be exterior to the cycle $\mathcal{C}_{i,j}$. There exists some

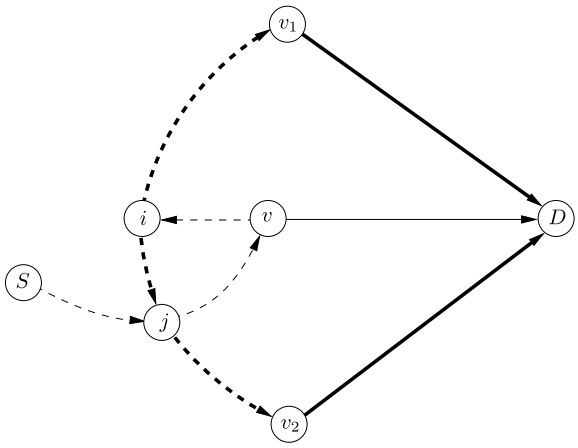

Fig. 15. A diagram of the planar embedding being used to prove that if node v can reach $v_1 = N_{\text{cw}}(v)$ and $v_2 = N_{\text{ccw}}(v)$, then v cannot be in the interior of the undirected cycle $\bar{\mathcal{C}}$ without there being a directed cycle. Solid lines are single edges; dashed lines represent paths made up of possibly many edges. Thick lines correspond to the undirected cycle $\bar{\mathcal{C}}$.

path from S to k, which crosses $\mathcal{C}_{i,j}$ at a node j'. Observe that j' is on the path from i to j, so it is reachable from i. Therefore i can reach j' and j' can reach k, so i can reach k. ■

In the planar embedding, we may travel around node D, noting the order in which nodes $\mathcal{N}_{\text{in}}(D)$ connect to it. In particular, for each $v \in \mathcal{N}_{\text{in}}(D)$, define $N_{\text{cw}}(v)$ to be the node in $\mathcal{N}_{\text{in}}(D)$ immediately clockwise of v, and $N_{\text{ccw}}(v)$ to be the node immediately counter-clockwise of v.

Claim 5: If $v \in \mathcal{N}_{\text{in}}(D)$ can reach both $N_{\text{cw}}(v)$ and $N_{\text{ccw}}(v)$, then v can reach every node in $\mathcal{N}_{\text{in}}(D)$. Moreover, there exists $u \in \{N_{\text{cw}}(v), N_{\text{ccw}}(v)\}$, and for each $k \in \mathcal{N}_{\text{in}}(D) \setminus \{v\}$ there exists an undirected cycle $\mathcal{C}_{v,k}$ composed of a path from v to k and the edges $(v, D), (k, D)$, such that:

1) All nodes in $\mathcal{N}_{\text{in}}(D)$ are not exterior to $\mathcal{C}_{v,u}$.
2) Node u is exterior to $\mathcal{C}_{v,k}$ for all $k \in \mathcal{N}_{\text{in}}(D) \setminus \{v, u\}$.

Proof: Define for convenience $v_1 := N_{\text{cw}}(v)$ and $v_2 := N_{\text{ccw}}(v)$. Assume v can reach both v_1 and v_2. Let i be a node that can be reached by v, can reach both v_1 and v_2, and can reach no other such node. Let $\mathbf{p}'$ be a path from v to i. Since one output edge from v goes directly to D, $\mathbf{p}'$ contains at least one edge; *i.e.* $i \neq v$. Let $\mathbf{p}_1$ be a path from i to v_1, and let $\mathbf{p}_2$ be a path from i to v_2. Note that $\mathbf{p}_1, \mathbf{p}_2$ share no edges. Let $\bar{\mathcal{C}}$ be the undirected cycle composed of $\mathbf{p}_1, \mathbf{p}_2$, and the edges $(v_1, D), (v_2, D)$.

We claim that v must be exterior to $\bar{\mathcal{C}}$, or else there is a cycle. (Recall we assume acyclic networks.) Note that every node in $\bar{\mathcal{C}}$ is reachable from i, and v can reach i, so v cannot be in $\bar{\mathcal{C}}$ without forming a cycle. Now suppose v is in the interior of $\bar{\mathcal{C}}$. Fig. 15 diagrams this construction. Any path from S to v must cross $\bar{\mathcal{C}}$ at a node j, reachable from i. Since j is on a path from S to v, j can reach v, so there is a cycle.

Let $\mathcal{C}_{v,v_1}$ be composed of the paths $\mathbf{p}', \mathbf{p}_1$ and the edges $(v, D), (v_1, D)$, and let $\mathcal{C}_{v,v_2}$ be composed of the paths $\mathbf{p}', \mathbf{p}_2$ and the edges $(v, D), (v_2, D)$. Of the three cycles $\bar{\mathcal{C}}, \mathcal{C}_{v,v_2}$, and $\mathcal{C}_{v,v_1}$, each pair share edges. In particular, since v is exterior to $\bar{\mathcal{C}}$, the interior of $\bar{\mathcal{C}}$ is contained in the interior of $\mathcal{C}_{v,u}$, where either $u = v_1$ or $u = v_2$. The two possibilities are illustrated in Fig. 16. The figure is slightly deceptive in that it may be that $i = v_1$ or $i = v_2$, but still it is the case that the interior of $\bar{\mathcal{C}}$ is contained in the interior of either $\mathcal{C}_{v,v_2}$ (as in subplot (a)) or $\mathcal{C}_{v,v_1}$ (as in subplot (b)). If an element of $\mathcal{N}_{\text{in}}(D)$ were exterior to $\mathcal{C}_{v,u}$, then it would be between v and u in the order that nodes connect to D. But this is impossible, since $u \in \{N_{\text{cw}}(v), N_{\text{ccw}}(v)\}$, so u and v are immediately adjacent in the connection order. Thus no element of $\mathcal{N}_{\text{in}}(D)$ is exterior to $\mathcal{C}_{v,u}$. By Claim 4, v can reach every node in $\mathcal{N}_{\text{in}}(D)$.

Fix any $k \in \mathcal{N}_{\text{in}}(D) \setminus \{v, u\}$. Since every node has two input edges, we may construct a path backwards from k, avoiding u whenever possible, until eventually reaching S. Thus there is a path with tail S and head k that avoids u. This path intersects $\mathcal{C}_{v,u}$ at a node j. This construction is shown in Fig. 17 for the case that $u = v_1$. Let $\mathcal{C}_{v,k}$ be composed of the path along $\mathcal{C}_{v,u}$ from v to j, followed by a path from j to k that avoids u, along with the edges $(v, D), (k, D)$. Since $\mathcal{C}_{v,k}$ does not include u, u is exterior to it. ■

Claim 6: There exist nodes $u_m \in \mathcal{N}_{\text{in}}(D)$ for $m \in [M]$ such that:

1) $\{u_1, \ldots, u_M\} = \mathcal{N}_{\text{in}}(D)$.
2) For any $m \in [M]$, if u_m can reach $u_{m'}$ for any $m' \neq m$, then u_m can reach either u_{m+1} or u_{m-1}, but not both.
3) Fix $m, m' \in [M]$ such that u_m can reach $u_{m'}$. Let

$$\mathcal{U} := \{u_{m''} : m < m'' < m' \text{ or } m' < m'' < m\}. \quad (209)$$

There exists an undirected cycle $\mathcal{C}_{u_m,u_{m'}}$ composed of a path from u_m to $u_{m'}$ and the edges $(u_m, u_{m'})$ such that all nodes in $\mathcal{U}$ are not exterior to $\mathcal{C}_{u_m,u_{m'}}$. Moreover, u_m can reach all nodes in $\mathcal{U}$.

Proof: Let u_1 be a node in $\mathcal{N}_{\text{in}}(D)$ that can be reached by no other nodes in $\mathcal{N}_{\text{in}}(D)$. (Such a node exists or else there would be a cycle.) Consider the case that u_1 can reach both $N_{\text{cw}}(u_1)$ and $N_{\text{ccw}}(u_1)$. Let $v := u_1$, and let $u \in \{N_{\text{cw}}(v), N_{\text{ccw}}(v)\}$ and $\mathcal{C}_{v,k}$ for all $k \in \mathcal{N}_{\text{in}}(D) \setminus \{v\}$ be as defined in Claim 5. If $u = N_{\text{cw}}(v)$, then set

$$u_m := N_{\text{ccw}}(u_{m-1}) \quad \text{for } m = 2, \ldots, M. \quad (210)$$

If $u = N_{\text{ccw}}(v)$, then set

$$u_m := N_{\text{cw}}(u_{m-1}) \quad \text{for } m = 2, \ldots, M. \quad (211)$$

By Claim 1, $|\mathcal{N}_{\text{in}}(D)| = |\mathcal{E}_{\text{in}}(D)| = M$. Thus both (210) and (211) satisfy statement (1) of the claim. Moreover, in either case $u_M = u$, so by Claim 5 all nodes in $\mathcal{N}_{\text{in}}(D)$ are not exterior to $\mathcal{C}_{u_1,u_M}$, and u_M is exterior to $\mathcal{C}_{u_1,u_m}$ for all $m \in \{2, \ldots, M-1\}$. For $m, m' \in [M]$ where $m > 1$ and u_m can reach $u_{m'}$, define $\mathcal{C}_{u_m,u_{m'}}$ based on an arbitrary path from u_m to $u_{m'}$.

Now consider the case that u_1 cannot reach both $N_{\text{cw}}(u_1)$ and $N_{\text{ccw}}(u_2)$. If u_1 cannot reach $N_{\text{cw}}(u_1)$, then set u_m as in (210). Otherwise set u_m as in (211). In either case, u_1 cannot reach u_M. For all $m, m' \in [M]$ where u_m can reach $u_{m'}$, define $\mathcal{C}_{u_m,u_{m'}}$ in an arbitrary manner.

Next we prove statement (3) of the claim. Fix $m, m' \in [M]$ where u_m can reach $u_{m'}$. Let $\bar{\mathcal{U}} := \mathcal{N}_{\text{in}}(D) \setminus \{u_m, u_{m'}\}$.

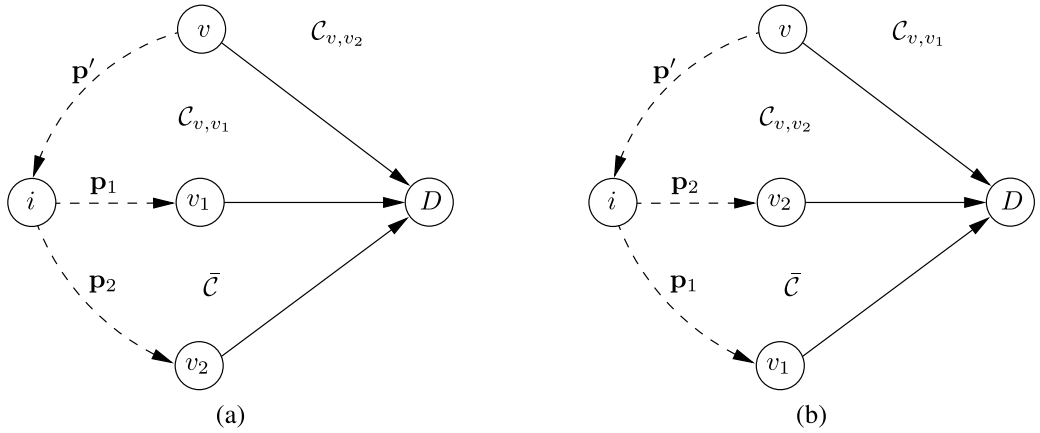

Fig. 16. Diagrams showing the two possible configurations given that v is exterior to $\bar{\mathcal{C}}$. Single edges are solid lines, and paths are dashed lines ($\mathbf{p}'$ contains at least one edge, but $\mathbf{p}_1$ and $\mathbf{p}_2$ may contain no edges). The three cycles $\mathcal{C}_{v,v_1}, \mathcal{C}_{v,v_2}, \bar{\mathcal{C}}$ are labeled, as well as the paths $\mathbf{p}', \mathbf{p}_1, \mathbf{p}_2$. Undirected cycle $\mathcal{C}_{v,v_1}$ is composed of $\mathbf{p}', \mathbf{p}_1, (v, D), (v_1, D)$; cycle $\mathcal{C}_{v,v_2}$ is composed of $\mathbf{p}', \mathbf{p}_2, (v, D), (v_2, D)$; cycle $\bar{\mathcal{C}}$ is composed of $\mathbf{p}_1, \mathbf{p}_2, (v_1, D), (v_2, D)$. In subplot (a) we set $u = v_2$, in subplot (b) we set $u = v_1$. Either way the interior of $\bar{\mathcal{C}}$ is contained in the interior of $\mathcal{C}_{v,u}$.

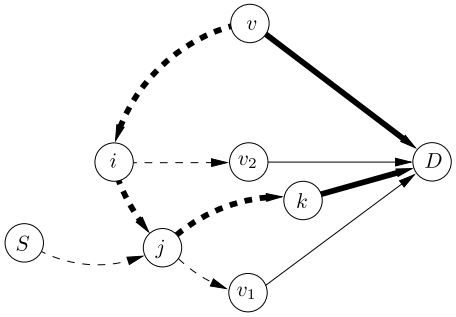

Fig. 17. Diagram showing the construction of the undirected cycle $\mathcal{C}_{v,k}$ for the case that $u = v_1$. Single edges are solid lines, and paths are dashed lines. Thick lines are those in $\mathcal{C}_{v,k}$. Node v_2 is exterior to $\mathcal{C}_{v,k}$ even if $j = k$.

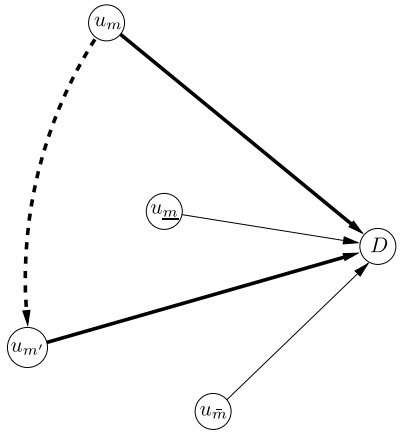

Fig. 18. A diagram of planarity being used to prove that if the order that nodes enter D is $(u_m, u_{\underline{m}}, u_{m'}, u_{\bar{m}})$, and $u_{\bar{m}}$ is exterior to $\mathcal{C}_{u_m,u_{m'}}$, then $u_{\underline{m}}$ cannot also be in exterior to $\mathcal{C}_{u_m,u_{m'}}$. Solid lines are single edges; dashed lines are paths; think lines are on the undirected cycle $\mathcal{C}_{u_m,u_{m'}}$.

To prove statement (3), it is enough to show there exists some $u_{\bar{m}} \in \bar{\mathcal{U}}$ that is exterior to $\mathcal{C}_{u_m,u_{m'}}$. To see why this is so, take any $u_{\underline{m}} \in \mathcal{U} \backslash \mathcal{C}_{u_m,u_{m'}}$ (*i.e.* u_{m_1} is either strictly in the interior or the exterior of $\mathcal{C}_{u_m,u_{m'}}$). By the definition of $\mathcal{U}$ in (209), $\underline{m}$ is strictly between m and m', and since $u_{\bar{m}} \in \bar{\mathcal{U}}$, $\bar{m}$ is not between m and m'. Thus, if we were to sort these four integers in increasing order, we would find some permutation or reflection of the order $(m, \underline{m}, m', \bar{m})$. By the constructions of $u_1, \ldots, u_M$ in (210)–(211), traveling around D in the planar embedding will encounter these nodes in either increasing or decreasing order in their subscript. Thus the four nodes are encountered in some permutation or reflection of $(u_m, u_{\underline{m}}, u_{m'}, u_{\bar{m}})$. As shown in Fig. 18, this order implies that if $u_{\bar{m}}$ is exterior to $\mathcal{C}_{u_m,u_{m'}}$, then $u_{\underline{m}}$ must be interior to $\mathcal{C}_{u_m,u_{m'}}$. Therefore, if there is some $u_{\bar{m}} \in \bar{\mathcal{U}}_2$ exterior to $\mathcal{C}_{u_m,u_{m'}}$, then no node in $\mathcal{U}$ is exterior to $\mathcal{C}_{u_m,u_{m'}}$. In particular, by Claim 4 all nodes in $\mathcal{U}$ can be reached from u_m.

First consider the case that $m > 1$. By construction u_m cannot reach u_1, so by Claim 4, u_1 is exterior to $\mathcal{C}_{u_m,u_{m'}}$. Moreover, $m' > 1$, so $u_1 \in \bar{\mathcal{U}}$. Therefore the above argument proves statement (3). Now suppose $m = 1$ and u_1 cannot reach both $N_{\text{cw}}(u_1)$ and $N_{\text{ccw}}(u_1)$. By construction u_1 cannot reach u_M, so by Claim 4 u_M is exterior to $\mathcal{C}_{u_1,u_{m'}}$. Since $u_M \in \bar{\mathcal{U}}$, again the above argument proves statement (3). The final case is that $m = 1$ and u_1 can reach both $N_{\text{cw}}(u_1)$ and $N_{\text{ccw}}(u_1)$. By Claim 5 all of $\mathcal{N}_{\text{in}}(D)$ are not exterior to $\mathcal{C}_{u_1,u_M}$, so if $m' = M$, then we are done. Otherwise, by Claim 5 node u_M is exterior to $\mathcal{C}_{u_1,u_{m'}}$. Since $u_M \in \bar{\mathcal{U}}$, the above argument proves statement (3).

Now we prove statement (2). If u_m can reach any other node $u_{m'}$, then either $m' = m \pm 1$, or $m \pm 1 \in \mathcal{U}$. Thus by the above argument, u_m can reach either u_{m+1} or u_{m-1}. Suppose it could reach both. In order for both nodes to exist, we need $1 < m < M$. The constructions in (210)–(211) imply $\{u_{m-1}, u_{m+1}\} = \{N_{\text{cw}}(u_m), N_{\text{ccw}}(u_m)\}$, so u_m can reach both $N_{\text{cw}}(u_m)$ and $N_{\text{ccw}}(u_m)$. By Claim 5, this means that u_m can reach all other nodes in $\mathcal{N}_{\text{in}}(D)$, including u_1, which contradicts the construction of u_1. This proves statement (2). ■

Algorithm 1 gives a method to construct edge-independent non-destination paths $\mathbf{p}_v$ for each 2-to-1 node v satisfying (3). The algorithm behaves as follows: There are of three main

Algorithm 1 Construct Non-Destination Paths to Satisfy PATHS

Input: Network (V, E) satisfying 2-IN and PLANAR.
Output: Edge-independent paths $\mathbf{p}_v$ for each 2-to-1 node v such that $\text{tail}(\mathbf{p}_v) = S$, $\text{head}(\mathbf{p}_v) = v$, $\Gamma_v \subset \mathbf{p}_v$, and $\mathcal{N}_{\text{in}}(D) \subset \bigcup_{v \in \mathcal{V}_{2,1}} \mathbf{p}_v$.

1: For all $v \in \mathcal{V}_{2,1}$, $\mathbf{p}_v \leftarrow \bar{\Gamma}_v$, where $\bar{\Gamma}_v$ is as defined in Claim 3.
2: Let $u_1, \ldots, u_M$ be as defined in Claim 6.
3: Let $\mathcal{R}_1, \ldots, \mathcal{R}_L$ be disjoint sets so $\bigcup_{l \in [L]} \mathcal{R}_l = \mathcal{N}_{\text{in}}(D)$ such that given $u_m \in \mathcal{R}_l$, $u_{m+1} \in \mathcal{R}_l$ if and only if there is a path between u_m and u_{m+1} in either direction.
4: **for** $l \in [L]$ **do**
5: *Claim 7:* There is exactly one 2-to-1 node v in $\mathcal{R}_l$.
6: Let $m \in [M]$ be such that $u_m = v$.
7: Let $m_1, m_2 \in [M]$ be such that

$$\mathcal{R}_l = \{u_{m_1}, u_{m_1+1}, \ldots, u_m, \ldots, u_{m_2-1}, u_{m_2}\}.$$

8: Let m_1' be the largest integer such that $m_1 \le m_1' < m$ and $u_{m_1'} \in \mathcal{R}_l \setminus \Gamma_v$. If m_1' exists, $i_1 \leftarrow u_{m_1'}$.
9: Let m_2' be the smallest integer such that $m < m_1' \le m_2$ and $u_{m_2'} \in \mathcal{R}_l \setminus \Gamma_v$. If m_2' exists, $i_2 \leftarrow u_{m_2'}$.
10: *Claim 8:* For $a \in \{1, 2\}$, there exist paths $\mathbf{q}_a$ such that $\text{tail}(\mathbf{q}_a) = u_{m_a}$, $\text{head}(\mathbf{q}_a) = u_{m_a'}$, and

$$\{u_{m_1}, u_{m_1+1}, \ldots, u_{m_1'}\} \subset \mathbf{q}_1,$$
$$\{u_{m_2}, u_{m_2-1}, \ldots, u_{m_2'}\} \subset \mathbf{q}_2.$$

11: Let $\mathcal{Q}_v$ be the set of 2-to-1 nodes v' such that the only node in $\mathcal{V}_{2,1} \cap \mathcal{N}_{\text{in}}(D)$ reachable from v' is v.
12: *Claim 9:* There exists distinct $v_1, v_2 \in \mathcal{Q}_v$ and paths $\mathbf{r}_1, \mathbf{r}_2$ such that for $a \in \{1, 2\}$, $\text{tail}(\mathbf{r}_a) = i_a$ and $\text{head}(\mathbf{r}_a) = \text{tail}(\mathbf{p}_{v_a})$.
13: For $a \in \{1, 2\}$, $\mathbf{p}_{v_a} \leftarrow (\mathbf{q}_a, \mathbf{r}_a, \mathbf{p}_{v_a})$.
14: **end for**
15: Let $\Psi : V \setminus \{S, D\} \to [|V| - 2]$ be a bijection such that if $(i, j) \in E$, then $\Psi(i) > \Psi(j)$.
16: **for** $q = 1$ to $|V| - 2$ **do**
17: $j \leftarrow \Psi^{-1}(q)$
18: **if** there is a node $i \in \mathcal{N}_{\text{in}}(j)$ such that (i, j) is on both $\mathbf{p}_{v_1}$ and $\mathbf{p}_{v_2}$ with $v_1 \neq v_2$ **then**
19: **for** $a \in \{1, 2\}$ **do**
20: **if** no nodes in $\mathbf{p}_{v_a} \cap \mathcal{N}_{\text{in}}(D) \setminus \bigcup_{v \in \mathcal{V}_{2,1} \setminus \{v_a\}} \mathbf{p}_v$ can reach i **then**
21: Replace $\mathbf{p}_{v_a}$ with its sub-path from j to v_a.
22: **continue**
23: **else**
24: Let $m_a \in [M]$ be such that u_{m_a} is the furthest downstream node in $\mathbf{p}_{v_a} \cap \mathcal{N}_{\text{in}}(D) \setminus \bigcup_{v \in \mathcal{V}_{2,1} \setminus \{v_a\}} \mathbf{p}_v$ that can reach i.
25: **end if**
26: **end for**
27: Let i' be the input neighbor of j other than i.
28: *Claim 10:* There exists a node k that can reach i', where k is on the sub-path of $\mathbf{p}_{v_a}$ between u_a and i for some $a \in \{1, 2\}$.
29: Replace the sub-path of $\mathbf{p}_{v_a}$ from k to j with a path from k to i' followed by (i', j).
30: **end if**
31: **end for**
32: **for** $v \in \mathcal{V}_{2,1}$ **do**
33: Let $\mathbf{s}_v$ be a path from S to $\text{tail}(\mathbf{p}_v)$ that is edge-independent from all paths $\mathbf{p}_{v'}$ for $v' \in \mathcal{V}_{2,1} \setminus \{v\}$.
34: $\mathbf{p}_v \leftarrow (\mathbf{s}_v, \mathbf{p}_v)$
35: **end for**

loops. In the first loop (lines 4–14), non-destination paths are constructed to satisfy (3) while ignoring the edge-independence constraint. In the second loop (lines 16–31), the paths are adjusted to become edge-independent while maintaining (3). In the third (lines 32–35), the paths are extended backwards so they begin at S.

Several claims are stated within the algorithm description. We prove these claims, then proceed to prove algorithm correctness.

Proof of Claim 7: Suppose $\mathcal{R}_l$ contains two 2-to-1 nodes $u_m, u_{m'}$ where $m < m'$. By construction of $\mathcal{R}_l$, it must also contain $u_{m+1}, u_{m+2}, \ldots, u_{m'-1}$. Since $u_m \in \mathcal{N}_{\text{in}}(D)$ has only one output edge, and that output edge lead directly to D, u_m cannot reach any other nodes in $\mathcal{N}_{\text{in}}(D)$. Thus, since $u_{m+1} \in \mathcal{R}_l$, there must be a path from u_{m+1} to u_m. By Claim 6, u_{m+1} cannot also reach u_{m+2}. Hence u_{m+2} must be able to reach u_{m+1}. Continuing this argument, $u_{m'-1}$ must be able to reach $u_{m'-2}$, and so $u_{m'}$ must be able to reach $u_{m'-1}$. But since $u_{m'}$ is also a 2-to-1 node in $\mathcal{N}_{\text{in}}(D)$, this is impossible. This proves that $\mathcal{R}_l$ contains at most one 2-to-1 node.

Now we show that $\mathcal{R}_l$ contains at least one 2-to-1 node. Take any $u_m \in \mathcal{R}_l$. If u_m is 2-to-1, we are done. Otherwise, by Claim 2, u_m can reach at least one 2-to-1 node in $\mathcal{N}_{\text{in}}(D)$. Hence by Claim 6, u_m can reach either u_{m+1} or u_{m-1}. Assume without loss of generality that it can reach u_{m+1}. Thus $u_{m+1} \in \mathcal{R}_l$. If u_{m+1} is 2-to-1, we are done. Otherwise, u_{m+1} can also reach a 2-to-1 node in $\mathcal{N}_{\text{in}}(D)$, so again by Claim 6 it must be able to reach u_{m+2}. This process cannot continue indefinitely, so it must terminate at a 2-to-1 node in $\mathcal{R}_l$. ∎

Proof of Claim 8: By the construction of $\mathcal{R}_l$, for all m' with $m_1 \le m' < m$, $u_{m'}$ can reach $u_{m'+1}$, and for all m' with $m < m' \le m_2$, $u_{m'}$ can reach $u_{m'-1}$. The existence of the paths $\mathbf{q}_1, \mathbf{q}_2$ follows. ∎

Proof of Claim 9: The sets $\mathcal{Q}_v$ for each $v \in \mathcal{V}_{2,1} \cap \mathcal{N}_{\text{in}}(D)$ are disjoint. Thus choosing $v_1, v_2 \in \mathcal{Q}_v$ guarantees that v_1, v_2 were not selected in line 12 of any previous iteration of the loop, and so before the current iteration of line 12, for all $v' \in \mathcal{Q}_v$ the path $\mathbf{p}_{v'}$ has not been modified since line 1; thus $\mathbf{p}_{v'} = \bar{\Gamma}_{v'}$. Note that $v \in \mathcal{Q}_v$. If $v' \in \mathcal{Q}_v \setminus \{v\}$, then v' can reach v, meaning $\Gamma_{v'} = \emptyset$, and so $\text{tail}(\mathbf{p}_{v'}) = \text{tail}(\bar{\Gamma}_{v'}) = v'$. In the following cases and sub-cases, we select v_1, v_2 and $\mathbf{r}_1, \mathbf{r}_2$:

1) *Neither i_1 nor i_2 exists*: There is nothing to do.
2) *i_1 exists, i_2 does not exist:*
   a) *There is a node $u_{m'} \in \mathcal{V}_{2,1} \cap \mathcal{N}_{\text{in}}(D) \setminus \{v\}$ that can be reached from i_1*: Since $i_1 = u_{m'_1}$ can reach $u_{m'_1+1}$, by Claim 6 $m' > m'_1$. Moreover, the only 2-to-1 node in $\mathcal{R}_l$ is v, so $u_{m'} \notin \mathcal{R}_l$. Thus $m' > m_2 \geq m > m'_1$. Hence by Claim 6 $v = u_m$ is not exterior to $\mathcal{C}_{i_1,u_{m'}}$. Moreover, no element of Γ_v can be exterior to $\mathcal{C}_{i_1,u_{m'}}$, or else it could reach $u_{m'}$ in addition to v. Hence, by Claim 4 i_1 can reach all elements of Γ_v, and in particular $\text{tail}(\bar{\Gamma}_v)$. Set $v_1 = v$ and $\mathbf{r}_1$ to be a path from i_1 to $\text{tail}(\bar{\Gamma}_v)$.
   b) *Node i_1 can reach no node in $\mathcal{V}_{2,1} \cap \mathcal{N}_{\text{in}}(D) \setminus \{v\}$*: Since i_1 can reach v but is not in Γ_v (by construction in line 8), it must be able to reach a 2-to-1 node other than v. Set v_1 to be this node. By assumption $v_1 \notin \mathcal{N}_{\text{in}}(D)$. Node v_1 can reach no node in $\mathcal{V}_{2,1} \cap \mathcal{N}_{\text{in}}(D) \setminus \{v\}$, or else i_1 could as well. Moreover, by Claim 2, v_1 can reach some 2-to-1 node in $\mathcal{N}_{\text{in}}(D)$, so v_1 must be able to reach v. Hence $v_1 \in \mathcal{Q}_v$. Since $v_1 \neq v$, $\text{tail}(\mathbf{p}_{v_1}) = v_1$, so we may set $\mathbf{r}_1$ to be a path from i_1 to v_1.
3) *i_1 does not exist and i_2 exists*: We may repeat the same two cases as above with i_2 in place of i_1.
4) *i_1, i_2 exist*:
   a) *There exist $v'_1, v'_2 \in \mathcal{V}_{2,1} \cap \mathcal{N}_{\text{in}}(D) \setminus \{v\}$ such that i_1 can reach v'_1 and i_2 can reach v'_2*: This cannot occur without a cycle. Indeed, if this were true then $v'_1, v'_2 \notin \mathcal{R}_l$, so by Claim 6 i_2 could reach i_1 and i_1 could reach i_2.
   b) *i_1 can reach a node in $\mathcal{V}_{2,1} \cap \mathcal{N}_{\text{in}}(D) \setminus \{v\}$, but i_2 cannot*: Let $u_{m'} \in \mathcal{V}_{2,1} \cap \mathcal{N}_{\text{in}}(D) \setminus \{v\}$ be a node reachable from i_1. As in case (2a), $u_{m'} \notin \mathcal{R}_l$, and so i_1 can reach $\text{tail}(\bar{\Gamma}_v)$. Set $v_1 = v$ and $\mathbf{r}_1$ to be a path from i_1 to $\text{tail}(\bar{\Gamma}_v)$. Set v_2 to a 2-to-1 node other than v that can be reached from i_2, and set $\mathbf{p}_2$ to be a path from i_2 to v_2. By the same argument as in case (2b), $\text{tail}(\mathbf{p}_{v_2}) = v_2$.
   c) *i_2 can reach a node in $\mathcal{V}_{2,1} \cap \mathcal{N}_{\text{in}}(D) \setminus \{v\}$, but i_1 cannot*: Repeat the previous case reversing the roles of i_1 and i_2.
   d) *Neither i_1 nor i_2 can reach a node in $\mathcal{V}_{2,1} \cap \mathcal{N}_{\text{in}}(D) \setminus \{v\}$*:
      i) *There exist distinct $v_1, v_2 \in \mathcal{V}_{2,1} \setminus \{v\}$ such that i_1 can reach v_1 and i_2 can reach v_2*: Set $\mathbf{r}_1$ to a path from i_1 to v_1, and set $\mathbf{r}_2$ to a path from i_2 to v_2. By the same argument in case (2b), $\text{tail}(\mathbf{p}_{v_a}) = v_a$ for $a = 1, 2$.
      ii) *There is only one node $\mathcal{V}_{2,1} \setminus \{v\}$ reachable by either i_1 or i_2*: Denote this unique node v'. By assumption v' cannot reach any node in $\mathcal{V}_{2,1} \cap \mathcal{N}_{\text{in}}(D) \setminus \{v\}$, but by Claim 2 it can reach a node in $\mathcal{V}_{2,1} \cap \mathcal{N}_{\text{in}}(D)$. Hence $v' \in \mathcal{Q}_v$. Let $\tilde{\mathcal{C}}_{i_1,i_2 \to v'}$ be the undirected cycle composed of a path from i_1 to v', a path from i_2 to v', and the edges (i_1, D), (i_2, D). This construction is diagrammed in Fig. 19. Node v' can reach v.

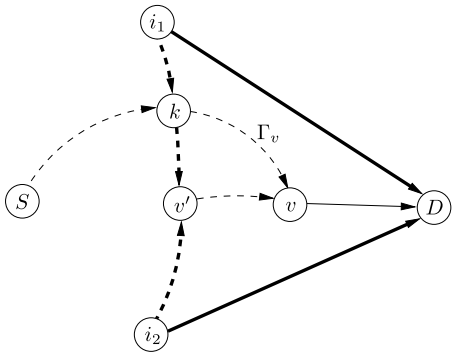

Fig. 19. Diagram of planarity being used to prove that there exists a path from either i_1 or i_2 to v through Γ_v. Solid lines are single edges; dashed lines represent paths. Thick lines correspond to the undirected cycle $\tilde{\mathcal{C}}_{i_1,i_2 \to v'}$. The set of nodes Γ_v are along the path indicated.

Thus v is in the interior of $\tilde{\mathcal{C}}_{i_1,i_2 \to v'}$. Now consider any path from S to v, passing through all nodes in Γ_v. This path must cross $\tilde{\mathcal{C}}_{i_1,i_2 \to v'}$ at a node k that is either on the path from i_1 to v' or the one from i_2 to v'. Suppose the former. Node k can reach both v' and v, so it cannot be in Γ_v. Set $v_1 = v$, and $\mathbf{r}_1$ to be composed of a path from i_1 to k, followed by a path from k to $\text{tail}(\bar{\Gamma}_v)$. Set $v_2 = v'$, and $\mathbf{r}_2$ to be a path from i_2 to v'. As in case (2b), $\text{tail}(\mathbf{p}_{v'}) = v'$. If k is on the path from i_2 to v', do the reverse. ■

Proof of Claim 10: Let u_m be the node such that $v_1 \in \mathcal{Q}_{u_m}$. First we show that either $m_1 < m < m_2$ or $m_2 < m < m_1$. Suppose otherwise: for example, suppose $m_1 < m_2 < m$. By the construction of the $\mathbf{p}_v$ paths in the first loop

$$\{u_{m_1}, u_{m_1+1}, \ldots, u_{m-1}\} \setminus \Gamma_{u_m} \subset \mathbf{p}_v. \tag{212}$$

(This does not change in the while loop.) Since both u_{m_1}, u_{m_2} can reach i, and i can reach both v_1, v_2, then $u_{m_1}, u_{m_2} \notin \Gamma_{u_m}$. Hence, by the assumption that $m_1 < m_2 < m$, (212) gives $u_{m_2} \in \mathbf{p}_{v_1}$, which contradicts the definition of m_2 in line 24. This precludes the ordering $m_1 < m_2 < m$. Analogous arguments can be used to preclude all orderings except $m_1 < m < m_2$ and $m_2 < m < m_1$. Assume the former without loss of generality.

Let $\tilde{\mathcal{C}}_{u_{m_1},u_{m_2} \to i}$ be the cycle composed of the sub-path of $\mathbf{p}_{v_1}$ from u_{m_1} to i, the sub-path of $\mathbf{p}_{v_2}$ from u_{m_2} to i, and the edges (u_{m_1}, D), (u_{m_2}, D). Since $m_1 < m < m_2$, u_m is in the interior of $\tilde{\mathcal{C}}_{u_{m_1},u_{m_2} \to i}$. Since j is on a path from i to u_m, j also in the interior. Hence i' is not in the exterior. Any path from the source to i' intersects $\tilde{\mathcal{C}}_{u_{m_1},u_{m_2} \to i}$ at a node k. (This is essentially the same as the construction in Fig. 19.) Node k can reach i', and it is on the sub-path of $\mathbf{p}_{v_a}$ between u_{m_a} and i for some $a \in \{1, 2\}$. ■

Proof of Correctness of Algorithm 1: In the first loop, the paths are constructed so that in line 13, $\text{head}(\mathbf{q}_a) = \text{tail}(\mathbf{r}_a)$ and $\text{head}(\mathbf{r}_a) = \text{tail}(\mathbf{p}_{v_a})$; hence the concatenation of paths is valid. Moreover, line 13 only extends $\mathbf{p}_{v_a}$, so after the

first loop

$$\Gamma_v \subset \mathbf{p}_v \text{ for } v \in \mathcal{V}_{2,1}. \tag{213}$$

The construction of $\mathbf{q}_1, \mathbf{q}_2$ is such that $\mathcal{R}_l \setminus \Gamma_v \subset \mathbf{q}_1 \cup \mathbf{q}_2$. Thus, after line 13, $\mathcal{R}_l \setminus \Gamma_v \subset \mathbf{p}_{v_1} \cup \mathbf{p}_{v_2}$. Since $\bigcup_{l\in[L]} \mathcal{R}_l = \mathcal{N}_{\text{in}}(D)$, after the first loop

$$\mathcal{N}_{\text{in}}(D) \subset \bigcup_{v\in\mathcal{V}_{2,1}} \mathbf{p}_v. \tag{214}$$

Note that (213)–(214) imply (3).

We now show if (213)–(214) hold at the start of given iteration of the second loop, they still hold at the end of it. In particular, the non-destination paths are only modified in lines 21 and 29, so we need to show that neither line affects (213)–(214). In any iteration of the loop, j can reach both v_1 and v_2, so j is neither in Γ_{v_1} nor Γ_{v_2}. Neither line 21 nor line 29 modify $\mathbf{p}_{v_a}$ downstream of j, so (213) remains after an iteration of the loop. Moreover, if in line 21 nodes in $\mathcal{N}_{\text{in}}(D)$ are dropped from $\mathbf{p}_{v_a}$, the condition in line 20 ensures that these dropped nodes are contained in another non-destination path, so (214) remains. In line 29, the sub-path of $\mathbf{p}_{v_a}$ between k and j contains no nodes in $\mathcal{N}_{\text{in}}(D)$ that are not also in another non-destination path, so again (214) remains.

Now we show that at the end of the second loop, all non-destination paths are edge-independent. In particular, we show that after the qth iteration, any input edge to $\Psi^{-1}(q')$ for $q' \le q$ is on at most one path. We prove this by induction. Certainly it is true before the first iteration. Now assume that at the start of the qth iteration, any input edge to $\Psi^{-1}(q')$ for $q' \le q-1$ is on at most one path. We show this remains true at the end of the qth iteration, and in addition any input edge to $\Psi^{-1}(q) = j$ is on at most one path. The construction of the Ψ bijection in line 15 ensures that all nodes downstream of j have been visited in earlier iterations of the loop, so initially each output edge from j is on at most one path. If j is 2-to-2, then no path may terminate at j, so at most two paths enter j. If j is 2-to-1, one path terminates at j, but only one path may exit j, so again at most two paths enter j. Thus, in either case, if initially (i, j) is on $\mathbf{p}_{v_1}$ and $\mathbf{p}_{v_2}$, no other paths enter j. In particular, (i', j) is on no paths. In any iteration of the loop, either line 21 or 29 will be executed. Either one leaves (i, j) and (i', j) on at most one path each. Moreover, both line 21 and 29 affect only nodes that can reach j. By the construction of Ψ in line 15, any node j' that can reach j has $\Psi(j') \ge \Psi(j)$. Thus, at the end of the iteration, any input edge to $\Psi^{-1}(q')$ for $q' \le q$ is on at most one non-destination path.

In the third loop, it is always possible to find a path $\mathbf{s}_v$ edge-independent from all other paths. This is because all paths terminate at 2-to-1 nodes, so if a node i has an output edge on no paths, then i has an input edge on no paths. Moreover, tail$(\mathbf{p}_v)$ must have an input edge on no paths. Thus we may travel backward from tail$(\mathbf{p}_v)$, avoiding any edge already on a path, and eventually reach S. After the third loop is complete, all paths begin at S, and remain edge-independent. ■

APPENDIX E
PROOF OF PROPOSITION 2

Fix three binary random variables X, Y, Z with distribution $p(xyz)$, not necessarily satisfying the conditions of the proposition. Let $\tilde{X}, \tilde{Y}, \tilde{Z}$ be random variables with distribution $q(xyz)$ satisfying

$$(X, Y) \sim (\tilde{X}, \tilde{Y}),\ (X, Z) \sim (\tilde{X}, \tilde{Z}),\ (Y, Z) \sim (\tilde{Y}, \tilde{Z}). \tag{215}$$

Let $r(xyz) = q(xyz) - p(xyz)$. For any $x, y \in \{0, 1\}$, we have $p(xy0) + p(xy1) = q(xy0) + q(xy1)$. Hence $r(xy0) = -r(xy1)$. Similarly, for any $x, z \in \{0, 1\}$, $r(x0z) = -r(x1z)$, and for any $y, z \in \{0, 1\}$, $r(0yz) = -r(1yz)$. Therefore

$$\begin{aligned} r(000) &= -r(001) = r(011) = -r(010) \\ &= r(110) = -r(111) = r(101) = -r(100). \end{aligned} \tag{216}$$

Thus, for some $\alpha \in \mathbb{R}$

$$r(000) = \alpha \quad r(001) = -\alpha \tag{217}$$
$$r(011) = \alpha \quad r(010) = -\alpha \tag{218}$$
$$r(101) = \alpha \quad r(100) = -\alpha \tag{219}$$
$$r(110) = \alpha \quad r(111) = -\alpha. \tag{220}$$

Let m_a and M_a be the minimum and maximum respectively of $\{p(000), p(011), p(101), p(110)\}$, and let m_b and M_b be the minimum and maximum respectively of $\{p(001), p(010), p(100), p(111)\}$. Since $q(xyz) = p(xyz) + r(xyz) \in [0, 1]$,

$$\max\{-m_a, M_b - 1\} \le \alpha \le \min\{1 - M_a, m_b\}. \tag{221}$$

Indeed, the complete set of distributions $q(xyz)$ satisfying (215) is given by the range (221). Condition 1 in the statement of the proposition holds if and only the only choice for α is zero. Thus condition 1 is equivalent to

$$\max\{-m_a, M_b - 1\} = \min\{1 - M_a, m_b\} = 0. \tag{222}$$

Note that condition 2 is equivalent to $m_a = m_b = 0$. This certainly implies (222). Now suppose $m_a > 0$. Then at least four triples have positive probability, so $p(xyz) < 1$ for all $x, y, z \in \{0, 1\}$, meaning $M_b < 1$. Thus $\max\{-m_a, M_b - 1\} < 0$, so (222) does not hold. The same argument follows for $m_b > 0$. Hence (222) implies $m_a = m_b = 0$.

REFERENCES

[1] R. Ahlswede, N. Cai, S. R. Li, and R. W. Yeung, "Network information flow," *IEEE Trans. Inf. Theory*, vol. 46, no. 4, pp. 1204–1216, Jul. 2000.

[2] L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 1, pp. 14–30, 1982.

[3] D. Dolev, "The Byzantine generals strike again," *J. Algorithms*, vol. 3, no. 1, pp. 14–30, 1982.

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

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

[6] S. Jaggi *et al.*, "Resilient network coding in the presence of Byzantine adversaries," *IEEE Trans. Inf. Theory*, vol. 54, no. 6, pp. 2596–2603, Jun. 2008.

[7] T. Ho, B. Leong, R. Koetter, M. Médard, M. Effros, and D. R. Korger, "Byzantine modification detection in multicast networks using randomized network coding," in *Proc. Int. Symp. Inf. Theory*, 2004.

[8] R. Koetter and F. R. Kschischang, "Coding for errors and erasures in random network coding," *IEEE Trans. Inf. Theory*, vol. 54, no. 8, pp. 3579–3591, Aug. 2008.
[9] O. Kosut, L. Tong, and D. Tse, "Nonlinear network coding is necessary to combat general Byzantine attacks," in *Proc. Allerton Conf. Commun., Control, Comput.*, Sep. 2009.
[10] S. Kim, T. Ho, M. Effros, and S. Avestimehr, "Network error correction with unequal link capacities," in *Proc. Allerton Conf. Commun., Control, Comput.*, Sep. 2009, pp. 1387–1394.
[11] M. Kim, M. Médard, J. Barros, and R. Koetter, "An algebraic watchdog for wireless network coding," in *Proc. Int. Symp. Inf. Theory*, Jul. 2009, pp. 1159–1163.
[12] G. Liang and N. H. Vaidya, "When watchdog meets coding," in *Proc. IEEE INFOCOM*, San Diego, CA, USA, Mar. 2010, pp. 1–9.
[13] I. Csiszár and J. Körner, *Information Theory: Coding Theorems for Discrete Memoryless Systems*. San Diego, CA, USA: Academic, 1981.
[14] R. Singleton, "Maximum distance Q-nary codes," *IEEE Trans. Inf. Theory*, vol. 10, no. 2, pp. 116–118, Apr. 1964.
[15] S. Kim, T. Ho, M. Effros, and S. Avestimehr, "New results on network error correction: Capacities and upper bounds," in *Proc. Inf. Theory Appl. Workshop*, 2010, pp. 1–10.
[16] S. Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi, and M. Medard, "Resilient network coding in the presence of Byzantine adversaries," in *Proc. IEEE 26th INFOCOM*, May 2007, pp. 616–624.
[17] M. Langberg, A. Sprintson, and J. Bruck, "The encoding complexity of network coding," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2386–2397, Jun. 2006.
[18] T. Cui, T. Ho, and J. Kliewer, "On secure network coding with nonuniform or restricted wiretap sets," *IEEE Trans. Inf. Theory*, vol. 59, no. 1, pp. 166–176, Jan. 2013.
[19] I. Csiszar and P. Narayan, "The capacity of the arbitrarily varying channel revisited: Positivity, constraints," *IEEE Trans. Inf. Theory*, vol. 34, no. 2, pp. 181–193, Mar. 1988.
[20] L. Song, R. Yeung, and N. Cai, "Zero-error network coding for acyclic networks," *IEEE Trans. Inf. Theory*, vol. 49, no. 12, pp. 3129–3139, Dec. 2003.
[21] N. J. A. Harvey, R. Kleinberg, and A. Lehman, "On the capacity of information networks," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2345–2364, Jun. 2006.
[22] R. Dougherty, C. Freiling, and K. Zeger, "Networks, matroids, and non-Shannon information inequalities," *IEEE Trans. Inf. Theory*, vol. 53, no. 6, pp. 1949–1969, Jun. 2007.
[23] F. Harary, *Graph Theory*. Reading, MA, USA: Addison-Wesley, 1972.

Oliver Kosut (S'06–M'10) received B.S. degrees in electrical engineering and mathematics from the Massachusetts Institute of Technology, Cambridge, MA in 2004 and a Ph.D. degree in electrical and computer engineering from Cornell University, Ithaca, NY in 2010.

He was a visiting student at University of California at Berkeley in 2008–2009. He was a Postdoctoral Research Associate in the Laboratory for Information and Decision Systems at MIT, Cambridge, MA, from 2010 to 2012. Since 2012, he has been an Assistant Professor at Arizona State University, Tempe, AZ. His research interests include network information theory, security, and power systems.

Lang Tong (S'87–M'91–SM'01–F'05) is the Irwin and Joan Jacobs Professor in Engineering at Cornell University, Ithaca, New York. He is also the Cornell site direction of the Power System Engineering Research Center (PSERC). Lang Tong's current research focuses on statistical inference, optimization, and economic problems in energy systems.

Lang Tong received the B.E. degree in Automation from Tsinghua University, Beijing, China, in 1985, and M.S. and Ph.D. degrees in electrical engineering in 1987 and 1991, respectively, from the University of Notre Dame, Notre Dame, Indiana. He was a Postdoctoral Research Affiliate at the Information Systems Laboratory, Stanford University in 1991. He was the 2001 Cor Wit Visiting Professor at the Delft University of Technology and had held visiting positions at Stanford University and the University of California at Berkeley.

Lang Tong received the 1993 Outstanding Young Author Award from the IEEE Circuits and Systems Society, the 2004 best paper award from IEEE Signal Processing Society, and the 2004 Leonard G. Abraham Prize Paper Award from the IEEE Communications Society. He is also a coauthor of seven student paper awards. He received Young Investigator Award from the Office of Naval Research. He was a Distinguished Lecturer of the IEEE Signal Processing Society.

David N. C. Tse (M'96–SM'07–F'09) received the B.A.Sc. degree in systems design engineering from University of Waterloo in 1989, and the M.S. and Ph.D. degrees in electrical engineering from Massachusetts Institute of Technology in 1991 and 1994 respectively. He is currently a professor at Stanford University. He received a 1967 NSERC graduate fellowship from the government of Canada in 1989, a NSF CAREER award in 1998, the Best Paper Awards at the Infocom 1998 and Infocom 2001 conferences, the Erlang Prize in 2000 from the INFORMS Applied Probability Society, the IEEE Communications and Information Theory Society Joint Paper Awards in 2001 and 2013, the Information Theory Society Paper Award in 2003, the 2009 Frederick Emmons Terman Award from the American Society for Engineering Education, a Gilbreth Lectureship from the National Academy of Engineering in 2012, the Signal Processing Society Best Paper Award in 2012 and the Stephen O. Rice Paper Award in 2013. He was an Associate Editor of the IEEE TRANSACTIONS ON INFORMATION THEORY from 2001 to 2003, the Technical Program co-chair in 2004 and the General co-chair in 2015 of the International Symposium on Information Theory. He is a coauthor, with Pramod Viswanath, of the text *Fundamentals of Wireless Communication*, which has been used in over 60 institutions around the world.