Efficient and Explicit Coding for Interactive Communication

Ran Gelles, Ankur Moitra and Amit Sahai

***Abstract*—We revisit the problem of reliable interactive communication over a noisy channel, and obtain the first fully explicit (randomized) efficient constant-rate emulation procedure for reliable interactive communication. Our protocol works for any discrete memoryless noisy channel with constant capacity, and fails with exponentially small probability in the total length of the protocol.**

Following a work by Schulman [Schulman 1993] our simulation uses a tree-code, yet as opposed to the non-constructive *absolute tree-code* used by Schulman, we introduce a relaxation in the notion of goodness for a tree code and define a *potent tree code*. This relaxation allows us to construct an explicit emulation procedure for any two-party protocol. Our results also extend to the case of interactive multiparty communication.

We show that a randomly generated tree code (with suitable constant alphabet size) is an efficiently decodable potent tree code with overwhelming probability. Furthermore we are able to partially derandomize this result by means of epsilon-biased distributions using only $O(N)$ random bits, where N is the depth of the tree.

I. Introduction

In this work, we study the fundamental problem of reliable *interactive* communication over a noisy channel. The famous coding theorem of Shannon [1] from 1948 shows how to transmit any message over a noisy channel with optimal rate such that the probability of error is exponentially small in the length of the message. However, if we consider an interactive protocol where individual messages may be very short (say, just a single bit), even if the entire protocol itself is very long, Shannon's theorem does not suffice.

In a breakthrough sequence of papers published in 1992 and 1993 [2], [3], Schulman attacked this problem and gave a *non-constructive* proof of the existence of a general method to emulate any two-party interactive protocol such that: (1) the emulation protocol only takes a constant-factor longer than the original protocol, and (2) if the emulation protocol is executed over a noisy channel, then the probability that the emulation protocol fails to return the same answer as the original protocol is exponentially small in the total length of the protocol. These results hold for any discrete memoryless channel with constant capacity, including the important case of a binary symmetric channel[1] with some constant crossover probability less than $\frac{1}{2}$.

Unfortunately, Schulman's 1992 emulation procedure [2] either required a nonstandard model in which parties already share a large amount of randomness before they communicate, where the amount of shared randomness is quadratic in the length of the protocol to be emulated[2], or required inefficient encoding and decoding. On the other hand, Schulman's 1993 emulation procedure [3] is non-constructive in that it relies on the existence of *absolute tree codes*[3]. The only known proofs of the existence of absolute tree codes are non-constructive, and finding an explicit construction remains an important open problem. Indeed picking a tree code uniformly at random almost surely results in a bad tree code.

Our Results. In this work, we revisit the problem of reliable interactive communication, and give the first fully *explicit* emulation procedure for reliable interactive communication over noisy channels with a constant communication overhead. Our results hold for any discrete memoryless channel with constant capacity (including the binary symmetric channel), and our protocol achieves failure probability that is exponentially small in the length of the original communication protocol[4]. To obtain this result, we do the following:

- We introduce a new notion of goodness for a tree code, and define the notion of a *potent tree code*. We believe that this notion is of independent interest.
- We prove the correctness of an explicit emulation procedure based on any potent tree code. (This replaces the need for absolute tree codes in the work of Schulman [3].) This procedure is efficient given a black box for efficiently decoding the potent tree code.
- We show that a randomly generated tree code (with suitable constant alphabet size) is a potent tree code with overwhelming probability. Furthermore, we show that a randomly generated tree code (when combined with a good ordinary error-correcting code) can be efficiently

Manuscript received January 00, 0000; revised January 00, 0000; accepted January 00, 0000. This work was supported in part by a Fannie and John Hertz Foundation Fellowship, a DARPA/ONR PROCEED award, NSF grants 1118096, 1065276, 0916574 and 0830803, a Xerox Foundation Award, a Google Faculty Research Award, an equipment grant from Intel, and an Okawa Foundation Research Grant.

R. Gelles and A. Sahai are with University of California, Los angeles, USA (e-mail: gelles@cs.ucla.edu, sahai@cs.ucla.edu). A. Moitra is with the Institute for Advanced Study, New Jersey, USA (e-mail: moitra@ias.edu).

This material is based upon work supported by the Defense Advanced Research Projects Agency through the U.S. Office of Naval Research under Contract N00014-11-1-0389. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

[1]The binary symmetric channel with crossover probability p is one that faithfully transmits a bit with probability $1-p$, and toggles the bit with probability p.

[2]Note that in Schulman's 1992 shared randomness protocol (called the "public coin" protocol in that paper [2]), if the parties communicated the shared randomness to each other, this would result in an inverse polynomial rate for the protocol overall. Schulman obtained his main result in the standard model (called the "private coin" model there) by applying an inefficient transformation, that destroys explicitness.

[3]We note, with apology, that what we are calling an "absolute tree code" is what Schulman calls a "tree code." We make this change of terminology because we will introduce an alternative relaxed notion of goodness for a tree code that will lead to our notion of a "potent tree code."

[4]Here we assume that we know the length of the protocol in advance.

decoded with respect to any discrete memoryless channel with constant capacity with overwhelming probability.

- Finally, we are able to partially derandomize the above result by means of epsilon-biased distributions. Specifically, using highly explicit[5] constructions of small bias sample spaces [4], we give a highly explicit description of what we call *small bias tree codes* of depth N using only $O(N)$ random bits. We show that such small bias tree codes are not only potent with overwhelming probability, but that the efficient decoding procedure above still works for any discrete memoryless channel with constant capacity.

With the above work done, our result is immediate: Since only $O(N)$ random bits are needed for choosing a small bias tree code, these random bits can be chosen once and for all, encoded using an ordinary block error-correcting code, and sent to the other party. Then a deterministic procedure can be used to finish the protocol.

We then present an alternative explicit randomized construction of potent tree codes that achieves better potency, but with somewhat higher probability of failure. Our construction is first based on the observation that the above application of epsilon-biased sample spaces can be applied to partially derandomizing random *linear* tree codes, achieving the same level of potency as small bias tree codes. We can then compose such codes with explicit constructions of *weak* tree codes [5] which guarantee good distance for all length $c \log N$ length divergent paths. This composition (which works by simply concatenating symbols) achieves potency for sub-constant ϵ with probability of failure exponentially small in ϵN.

We use this improved construction to extend our result to the case of any number of parties. Our explicit emulation procedure will have a $O(\log m)$ slowdown for m parties (regardless of the length of the protocol). To obtain our result, we adapt the (non-explicit) emulation procedure based on absolute tree codes given by Rajagopalan and Schulman [6], that achieved the same asymptotic slowdown of $O(\log m)$.

Also, another result we obtain relates to the recent work of Braverman and Rao [7]. They give a new simulation procedure, again based on absolute tree codes, which uses a constant-sized alphabet and succeeds against an adversarial channel as long as the fraction of errors is at most $1/4-\epsilon$ (the simulation tolerates a $1/8-\epsilon$ error fraction when using a binary alphabet). These results improve over the result of Schulman which can only tolerate a fraction of errors that is below $1/240$. We further demonstrate the applicability of potent tree codes by showing that the same result can be obtained once again by replacing an absolute tree code with a potent tree. However, like all previous work on the adversarial error case, we cannot efficiently perform the decoding steps needed in order to run the simulation procedure.

Our approach. We begin our investigation by asking the question: What properties does a tree code need in order to be useful for emulating protocols over noisy channels? (Without loss of generality, assume that protocols only exchange one bit at a time from each party.) For the purpose of this paper, a tree code is simply any deterministic on-line encoding procedure in which each symbol from the input alphabet Σ is (immediately) encoded with a single symbol from the output alphabet S, but the encoding of future input symbols can depend on all the input symbols seen so far. As such, any such deterministic encoding can be seen as a complete $|\Sigma|$-ary tree with each edge labeled with a single symbol of the output alphabet S.

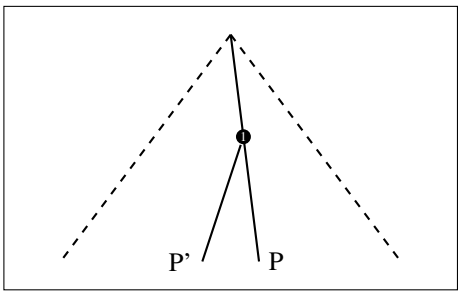

Fig. 1. A very bad tree code with two long paths that violate the hamming distance condition.

The usefulness of some kind of tree code for protocol emulation seems immediate, since each party must encode the bit it needs to send, before knowing what other bits it needs to send later (which it will not know until it receives messages from the other party). Let us associate every path from the root to a node in the tree code with the concatenation of output symbols along that path. Then, at first glance, it may appear that all we need from the tree code is for "long-enough" divergent paths to have large relative Hamming distance. That is, suppose that the tree code illustrated in Figure 1 has the property that the relative Hamming distance between the path from node 1 to P and the path from node 1 to P' is very small, even though each of those paths is long. This would certainly be problematic since the protocol execution corresponding to each path could be confused for the other. As long as all long divergent paths had high Hamming distance, however, it seems plausible that eventually the protocol emulation should be able to avoid the wrong paths. Also, it is important to note that with suitable parameters, a randomly generated tree code would guarantee that all long divergent paths have high relative Hamming distance with overwhelming probability.

However, this intuition does not seem to suffice, because while the protocol emulation is proceeding down an *incorrect* path, one party is sending the *wrong* messages – based on wrong interpretations of the other party's communication. After a party realizes that it has made a mistake, it must then be able to "backtrack" and correct the record going forward. The problem is that even short divergent paths with small relative Hamming distance can cause problems. Consider the tree code illustrated in Figure 2. In this figure suppose the path along the nodes 1, 2, and 3 is the "correct" path, but that the short divergent paths from 1 to A, 2 to B, and 3 to C all have small relative Hamming distance to the corresponding portions of the correct path. Then in the protocol emulation, because of the bad Hamming distance properties, the emulation may initially incorrectly proceed to node A, and then realize it made a mistake. But instead of correcting to a node on the correct path, it might correct to the node A' and proceed down the

[5]By a "highly explicit" object, we mean that the ith bit of the object should be computable in time polylogarithmic in the size of the object.

path to B. Then it may correct to B', and so on. Because the protocol emulation keeps making mistakes, it may never make progress towards simulating the original protocol.

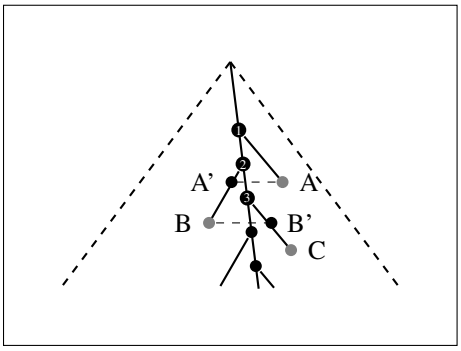

Fig. 2. A bad tree code with multiple short paths that violate the hamming distance.

Schulman [3] dealt with this problem by simply insisting that *all* divergent paths have large relative Hamming distance in his definition of an absolute tree code. This would prevent all such problems, and guarantee that any errors in emulation could be blamed on channel errors. The downside of this approach is that randomly generated tree codes would have short divergent paths with small (even zero) relative Hamming distance with overwhelming probability, and thus would not be absolute tree codes.

Our main observation is that this requirement goes too far. If a tree code has the property that for every path from root to leaf, there are only a few small divergent branches with low relative Hamming distance (as illustrated in Figure 3), then the emulation protocol will be able to recover from these few errors without any problems. We call such tree codes *potent tree codes* since they are sufficiently powerful to enable efficient and reliable interactive communication over a noisy channel.

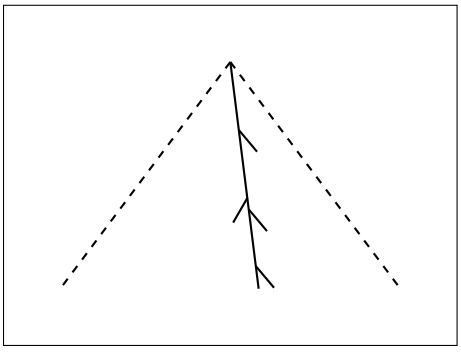

Fig. 3. A potent tree code with small number of bad paths.

More precisely, let ϵ and α be two parameters from the interval $[0, 1]$. Define a path from node u to a descendant node v (of length ℓ) to be α-*bad* if there exists a path from u to another descendant node w (also of length ℓ) such that u is the *first* common ancestor or v and w, and the Hamming distance between the u-v path and the u-w path is less than $\alpha\ell$. (Note that the u-v path and the u-w path must diverge at u, since u is the first common ancestor of v and w.) Then an (ϵ, α)-*potent tree code* of depth N is such that for every path Q from root to leaf, the number of nodes in the union of all α-bad paths contained in Q is at most ϵN.

We show that randomly generated tree codes (with suitable constant alphabet sizes) are potent tree codes with overwhelming probability. As hinted above, because every root-leaf path has good properties, a potent tree code will work for emulating *any* (adversarially chosen) interactive protocol. With some additional randomization, we show that within such emulations, decoding of a randomly generated potent tree code can be done efficiently even for an adversarially chosen protocol.

Naturalness of our definition. We can elucidate the relationship between potent tree codes and Schulman's absolute tree codes through an analogy with ordinary error correcting codes. Here, potent tree codes with $\epsilon = 0$ correspond to maximum distance separable (MDS) codes, yet just as MDS codes are powerful and useful objects, but not necessary for most applications, so too we can regard Schulman's absolute tree codes as powerful and useful, but not necessary for important applications like reliable interactive communication where potent tree codes suffice.

Other Related Work. Peczarski [8] provides a randomized way for constructing absolute tree codes. The construction succeeds with probability $1 - \epsilon$ using alphabet with size proportional to ϵ^{-1}. Therefore, using Peczarski's method to construct an absolute tree code with exponentially small failure probability ϵ, yields a polynomial slowdown; or a sub-linear but super-logarithmic slowdown if ϵ is negligible (in the length of the simulated protocol). Braverman [9] defines a "product" operator on tree-codes which yields a deterministic construction of an absolute tree codes which takes sub-exponential time. Other methods for constructing an absolute tree code are reported by Schulman [5], yet they require polynomial-size alphabet (in the depth of the tree), resulting in a logarithmic slowdown using Schulman's emulation [3]. Schulman [5] also provides methods for constructing tree codes with weaker properties such as satisfying the Hamming distance property for only a logarithmic depth (which yields a failure probability that is inverse-polynomial). Ostrovsky, Rabani, and Schulman [10] consider a relaxed problem of communication for control of polynomially bounded systems, and gave explicit constructions of codes suitable for that setting.

II. Preliminaries

We begin with several definitions that we use later. Unless otherwise mentioned, we use base 2 for all logarithms. Our model of communication assumes that some noisy discrete memoryless channel affects all communication between parties. A representative example of such a channel is the binary symmetric channel:

Definition 1. *A binary symmetric channel (BSC) with error probability p_{BSC} is a binary channel $\{0,1\} \to \{0,1\}$ such that each input bit is independently flipped with probability p_{BSC}.*

This channel is memoryless because conditioned on any bit in the input stream, the corresponding output bit is independent of all other bits in the input.

Shannon's coding theorem asserts the existence of an error-correcting code that reduces the error probability (for a single message) to be exponentially small, while increasing the amount of transmitted information by only a constant factor. We will not define the notion of a channel capacity formally, but for a binary symmetric channel with $p_{BSC} < 1/2$, the corresponding channel capacity C is strictly greater than zero.

Lemma II.1 (Shannon Coding Theorem [1])**.** *For any discrete memoryless channel T with capacity C, an alphabet S and any $\xi > 0$, there exists a code $enc : S \to \{0,1\}^n$ and $dec : \{0,1\}^n \to S$ with $n = O(\frac{1}{C}\xi \log |S|)$ such that*

$$\Pr\left[dec(T(enc(m))) \neq m\right] < 2^{-\Omega(\xi \log |S|)}.$$

This coding theorem will not be sufficient for coding in the context of *interactive* communication, since it assumes the entire messages is known to the encoding procedure. We require an encoding scheme in which the prefix of a message can be encoded independently of later bits in the message. The main structure we use is a *tree code*, introduced by Schulman [3], [11].

Definition 2. *The Hamming distance $\Delta(\sigma, \sigma')$ of two strings $\sigma = \sigma_1 \dots \sigma_m$ and $\sigma' = \sigma'_1 \dots \sigma'_m$ of length m over an alphabet S, is the number of positions i such that $\sigma_i \neq \sigma'_i$.*

A *tree code* is a (usually regular) tree for which every arc i in the tree is assigned a label σ_i over some fixed alphabet S. Denote with $w(s)$ the label of the arc between a node s and its parent, and with $W(s)$ the concatenation of the labels along the path from root to s. We associate a message with a root-to-leaf path, encoded as the symbols along the path. In a typical application, one requires a tree code to have good "distance" properties—divergent paths must be far apart in Hamming distance. We call these tree codes (as introduced by Schulman [11]), *absolute tree codes*:

Definition 3 (Tree Codes [11])**.** *An **absolute** d-ary tree code over an alphabet S, of distance parameter α and depth N, is a d-ary tree code such that for every two distinct nodes s and r at the same depth,*

$$\Delta(W(s), W(r)) \geq \alpha l,$$

where l is the distance from s and r to their least common ancestor.

It is shown in [11] that for every distance parameter $\alpha \in (0,1)$, there exists an absolute d-ary tree code of infinite depth, labeled using $|S| \leq 2\lfloor (2d)^{\frac{1}{1-\alpha}} \rfloor - 1$ symbols. Although these tree codes are known to exist, finding an efficient, explicit construction remains an open question.

Tree codes can be used to communicate a node u between the users, by sending the labels $W(u)$. Decoding a transmission means recovering the node at the end of the route defined by the received string of labels. In order to reduce the error probability of the label transmission, each label is separately coded using a standard error-correcting code. Note that the incremental communication cost of specifying a node v that is a child of u, after already transmitting the string $W(u)$ is just the cost of communicating the symbol $w(v)$. Our goal in most applications is to choose S to be constant-sized.

III. Potent Tree Codes

A. Potent Tree Codes and Their Properties

We now formally define the set of *potent trees* and its complement, the set of *bad trees*. The latter contains trees that are not useful for our purpose: at least one of their paths is composed of "too many" sub-paths that do not satisfy the distance condition, i.e., the total length of these sub-paths is at least ε fraction of the tree depth N, for some fixed constant $\varepsilon > 0$.

Definition 4. *Let u, v be nodes at the same depth h of a tree-code, and let w be their least common ancestor, located at depth $h - \ell$. We call the nodes u and v α-**bad nodes** (of length ℓ) if $\Delta(W(u), W(v)) < \alpha\ell$. Also, we call the path (of length ℓ) between w and u an α-**bad path** (similarly, the path between w and v would also be a bad path). Additionally, we call the interval $[h - \ell, h]$ an α-**bad interval** (of length ℓ).*

Definition 5. *An (ε, α)-**bad tree** is a tree of depth N that has a path Q for which the union of α-bad intervals corresponding to the α-bad subpaths of Q has total length at least εN. If the tree is not (ε, α)-**bad tree**, then we will call it an (ε, α)-**potent tree code**.*

We stress that a bad tree is not necessarily bad in *all* of its paths, since the existence of a single bad path is sufficient.

Suppose we randomly pick each label of the tree – call this construction a Random Tree Code (RTC). A RTC is a potent tree except with probability exponentially small in the depth of the tree (see details in [12]). The drawback of such a construction is that its description is exponential. However, we observe that requiring the entire tree to be random can be replaced with requiring any two paths along the tree to be *independent*. Using a method of Alon, Goldreich, Håstad and Peralta [4] we are able to construct a tree in which any two paths are *almost independent* – and we call such a code a *Small-Biased Tree Code* (SBTC). Moreover, such a tree has an efficient description and the randomness required to seed the construction is proportional to the depth of the tree (and hence the total communication required by the original protocol).

B. Small-Biased Random Trees as Potent Trees

In order to agree on a RTC with alphabet S, the users need to communicate (or pre-share) $O(d^N \log |S|)$ random bits. Surprisingly, we can reduce the description size to $O(N \log |S|)$ and still have a potent code with overwhelming probability. To accomplish this, we make use of Alon et al.'s construction of a sample space with an efficient description that is ϵ-biased [4].

Definition 6 (ϵ-biased sample space [13], [4])**.** *A sample space X on n bits is said to be ϵ-biased with respect to linear tests if for every sample $x_1 \cdots x_n$ and every string $\alpha_1 \cdots \alpha_n \in \{0,1\}^n \setminus \{0\}^n$, the random variable $y = \sum_{i=1}^n \alpha_i x_i \mod 2$ satisfies $|\Pr[y = 0] - \Pr[y = 1]| \leq \epsilon$.*

We use [4, Construction 2] to achieve a sample space $\mathbf{B}_n$ on n bits which is ϵ-biased with respect to linear tests. Let p be an odd prime such that $p > (n/\epsilon)^2$, and let $\chi_p(x)$ be the quadratic character of x (mod p). Let $\mathbf{B}_n$ be the sample space described by the following construction. A point in the sample space is described by a number $x \in [0, 1, \ldots, p-1]$, which corresponds to the n-bit string $r(x) = r_0(x)r_1(x)\cdots r_{n-1}(x)$ where $r_i(x) = \frac{1-\chi_p(x+i)}{2}$.

Proposition III.1 ([4], Proposition 2)**.** *The sample space $\mathbf{B}_n$ is $\frac{n-1}{\sqrt{p}} + \frac{n}{p}$-biased with respect to linear tests.*

We use this to construct a d-ary tree code of depth N with labels over an alphabet S. Without loss of generality we assume that $|S|$ is a power of 2, and describe the tree as the $d^N \log|S|$-bit string constructed by concatenating all the tree's labels in some fixed ordering. Since each n-bit sample describes a tree-code, we are sometimes negligent with the distinction between these two objects.

Definition 7. *A d-ary* Small-Biased Tree Code *(SBTC) of depth N, is a tree described by a sample from the sample space $\mathbf{B}_n$ with $n = d^N \log|S|$, $\epsilon = 1/2^{cN\log|S|}$ for some constant c which we choose later.*

We note that small-bias trees have several properties which are very useful for our needs. Specifically, every set of labels are almost independent.

Definition 8 (almost k-wise independence [4])**.** *A sample space on n bits is (ϵ, k)-**independent** if for any k positions $i_1 < i_2 < \cdots < i_k$ and k-bit string ξ,*

$$|\Pr[x_{i_1}x_{i_2}\cdots x_{i_k} = \xi] - 2^{-k}| \le \epsilon$$

Due to a lemma by Vazirani [14] (see also Corollary 1 in [4]), if a sample space is ϵ-biased with respect to linear tests, then for every k, the sample space is $((1-2^{-k})\epsilon, k)$-independent. Thus, $\mathbf{B}_n$ is (ϵ, k)-independent, for any k.

Corollary III.2. *Let $\mathcal{T}$ be a d-ary SBTC of depth N, then any $1 \le k \le d^N$ labels of $\mathcal{T}$ are almost independent, that is, any $k\log|S|$ bits of $\mathcal{T}$'s description are $(2^{-cN\log|S|}, k)$-independent.*

Finally, let us argue that such a construction is efficient (i.e., highly explicit). Let $p = O((n/\epsilon)^2)$ and assume a constant alphabet $|S| = O(1)$. Each sample x takes $\log p = O(N)$ bits, and each $r_i(x)$ can be computed by $poly(N)$ operations.

We now prove a main property about SBTCs. Except with negligible probability, a SBTC is potent.

Theorem III.3. *Suppose $\varepsilon, \alpha \in (0,1)$. Except with probability $2^{-\Omega(N)}$, a d-ary SBTC of depth N over alphabet $|S| > (2d)^{(2+2/\varepsilon)/(1-\alpha)}$ is (ε, α)-potent.*

Proof: We show that the probability of a d-ary SBTC of depth N to be (ε, α)-bad is exponentially small for a sufficiently large constant size alphabet S.

For a fixed node v, we bound the probability that v is α-bad of length l, i.e., the probability that there exists a node u at the same depth as v which imposes a bad interval of length l. Denote by $W_l(u)$ the last l labels of $W(u)$. Since the tree is $(1/2^{cN\log|S|}, 2l\log|S|)$-independent, then $W_l(u)$ and $W_l(v)$ are almost independent.

Lemma III.4. *For any two nodes v, u at the same depth with a common ancestor l levels away,*

$$\Pr[\Delta(W(u), W(v)) = j] \le \binom{l}{l-j}\left(\frac{1}{|S|}\right)^{l-j} + 2^{-\Omega(N)}.$$

Proof: Note that $W(u)$ and $W(v)$ are identical except for the last l labels. Using the fact that the labels are almost independent we can bound the probability $\Pr[\Delta(W(u), W(v)) = j] \le (2^{-2l\log|S|} + 2^{-cN\log|S|}) \times 2^{2l\log|S|}\binom{l}{l-j}\left(\frac{1}{|S|}\right)^{l-j}\left(\frac{|S|-1}{|S|}\right)^{j}$. Choosing $c > 3$ completes the proof. For the ease of notation, in the following we use $2\binom{l}{l-j}\left(\frac{1}{|S|}\right)^{l-j}$ as an upper bound of the above probability. ∎

The above lemma leads to the following bound on the probability that two (fixed) nodes at the same depth are α-bad.

Corollary III.5. $\Pr[\Delta(W(v), W(u)) \le \alpha l] \le 2\frac{2^l}{|S|^{(1-\alpha)l}}$.

For a fixed node v, the probability that there exists a node $u \ne v$ with least common ancestor l level away such that v and u do not satisfy the distance requirement, is bounded by $\sum_u 2\frac{2^l}{|S|^{(1-\alpha)l}} \le 2(2d/|S|^{1-\alpha})^l$, using a union bound.

Assume that the tree is bad, that is, there exists a path from root to some node z with bad intervals of total length εN. Due to the following Lemma III.6 there must exist *disjoint* bad intervals of total length at least $\varepsilon N/2$. Note that there are at most $\sum_{j=\varepsilon N/2}^{N}\binom{N}{j} \le 2^N$ ways to distribute these disjoint intervals along the path from root to z.

Lemma III.6 ([11])**.** *Let $\ell_1, \ell_2, \ldots, \ell_n$ be intervals on $\mathbb{N}$, of total length X. Then there exists a set of indices $I \subseteq \{1, 2, \ldots, n\}$ such that the intervals indexed by I are disjoint, and their total length is at least $X/2$. That is, for any $i, j \in I$, $\ell_i \cap \ell_j = \emptyset$, and $\sum_{i\in I}|\ell_i| \ge X/2$.*

A proof is given in [11].

Consider again the path from root to z, and the disjoint bad intervals of total length at least $\varepsilon N/2$ along it. There are at most $2N$ labels involved (along both the path to z and the colliding paths). Since the intervals are disjoint, their probabilities to occur are almost independent as well, and the probability that a specific pattern of intervals happens is bounded by their product. Hence, the probability for a SBTC to be (ε, α)-bad is bounded by

$$\begin{aligned}
&\Pr[\text{ SBTC is } (\varepsilon,\alpha)\text{-bad }] \\
&\quad\le \sum_{z}\sum_{\substack{\ell_1,\ell_2,\ldots \text{ disjoint,}\\ \text{of length} \ge \varepsilon N/2}}\prod_i 2(2d/|S|^{1-\alpha})^{\ell_i} \\
&\quad\le d^N \cdot 2^N (4d/|S|^{1-\alpha})^{\sum_i \ell_i} \le (2d)^N (4d/|S|^{1-\alpha})^{\varepsilon N/2}
\end{aligned}$$

which is exponentially small in N for a constant alphabet size $|S| > (4d\cdot(2d)^{2/\varepsilon})^{1/(1-\alpha)}$. ∎

IV. Potent Trees Applications

A. Simulation with Adversarial Errors

In a recent paper [7] Braverman and Rao show how to simulate any 2-party protocol over an adversarial channel, as long as the fraction of errors is at most $1/4 - \epsilon_2$, for any constant $\epsilon_2 > 0$. Their simulation is also based on absolute tree codes.

We show that the analysis of Braverman and Rao can be repeated using a $(\epsilon_1, 1-\epsilon_2)$-potent tree instead of an absolute tree code, and withstand error rate of up to $1/4 - 2\epsilon_1 - \epsilon_2$. Intuitively, for every node which is not $(1-\epsilon_2)$-bad, the potent tree code behaves exactly like an absolute tree code (i.e., many channel errors are required for having a decoding error). On the other hand, for every possible path along the potent tree, there are at most $\epsilon_1 N$ nodes which are $(1-\epsilon_2)$-bad, that is, at most $\epsilon_1 N$ additional times in which the scheme differs from an absolute tree code (in each one of the directions of communication). This gives an algorithm that withstand up to $1/4 - (2\epsilon_1 + \epsilon_2)$ fraction of (adversarial) errors.

Theorem IV.1. *For any 2-party binary protocol π and any constant $\epsilon > 0$ there exist a protocol Π that simulates π over an adversarial channel in which the fraction of errors is at most $1/4 - \epsilon$, uses potent tree-codes with a constant-sized alphabet and imposes a constant slowdown.*

We re-iterate that like all previous work on the adversarial error case, we cannot efficiently perform the decoding steps needed in order to run the simulation procedure.

The proof follows the analysis of Braverman and Rao [7] in a straightforward way, assuming the tree code in use is $(\epsilon_1, 1-\epsilon_2)$-potent (that is, $\alpha = 1-\epsilon_2$). In [7] the users consider π as a binary tree $\mathcal{T}$. Each path in the tree describes a possible transcript of π, where odd levels describe party A's outputs and even levels describe B's outputs. The users use an absolute tree code to communicate the vertices of $\mathcal{T}$ according to their inputs.

Assume that at time t user A sends a_t and let a'_t be the label received at B's side (similarly, user B sends b_t, etc.). Upon receiving a'_t, user B decodes the received string $a'_1, \ldots, a'_t$ and obtains a possible transcript of π, from which he can compute his next step in π. This process is repeated for $R = \lceil T/\epsilon_2 \rceil$ times.

Let $D(a'_1, \ldots, a'_t)$ denote a set of vertices in $\mathcal{T}$ described by decoding the received string. We denote with $m(i)$ the largest number such that the first $m(i)$ symbols of $D(a'_1, \ldots, a'_i)$ are equal to $a_1, \ldots, a_{m(i)}$ and the first $m(i)$ symbols of $D(b'_1, \ldots, b'_i)$ are equal to $b_1, \ldots, b_{m(i)}$.

Define $\mathcal{N}(i, j)$ to be the number of transmission errors in the $[i, j]$ interval of the simulation (for both users). In the analysis of [7], a lower bound on the number of errors is obtained assuming that simulation fails. We now show that using a $(\epsilon_1, 1-\epsilon_2)$-potent tree, the lower bound changes by at most ϵ_1.

The analysis of [7] begins by considering a simpler simulation in which the alphabet size might be polynomial, and then extends the result to a constant alphabet size in a straightforward way. In order to ease the proof, we show that the theorem holds for the simple protocol with polynomial alphabet. Extending the result to the constant-alphabet protocol is immediate.

Proof: (**Theorem IV.1**.) We redefine the quantity $\mathcal{N}$ to allow us consider possible errors caused by the tree in addition to channel errors. Let $\mathcal{N}(i, j, d)$ be the number of communication errors between rounds i and j, assuming that the total length of bad intervals along the paths $a_i, \ldots, a_j$ and $b_i, \ldots, b_j$ in the potent tree, is at most d.

Lemma IV.2 (replacing lemma 4 of [7])**.** $\mathcal{N}(m(i)+1, i, d) \geq (1-\epsilon_2)(i - m(i))/2 - d$

Proof: Without loss of generality, we assume that the $(m(i)+1)$-th symbol in $D(a'_1, \ldots, a'_i)$ differs from $a_{m(i)+1}$. Consider two cases. If the node a_i is not α-bad, then the only way to get a decoding error of magnitude $l = i - m(i)$ is if at least $\alpha l/2 = (1-\epsilon_2)(i-m(i))/2$ communication errors have happened (this is identical to [7]).

In the second case, the node a_i is α-bad. If $i - m(i) \leq d$ the lemma is trivial. Otherwise, a_i must be an α-bad node of maximal length at most d. $\Delta(a_1 \cdots a_i, a'_1 \cdots a'_i) \geq \alpha(i - m(i))$ and again such a decoding error implies at least $(1-\epsilon_2)(i-m(i))/2$ communication errors. ∎

The quantity $t(i)$ is defined by [7] as the earliest round j such that both users announced the first i edges of $\mathcal{T}$ within their transmissions. The following Lemma is stated in [7].

Lemma IV.3 (Lemma 5 of [7])**.** *For $i \geq 0, k \geq 1$, if $i + 1 < t(k)$, then $m(i) < t(k-1)$*

The proof of this lemma is independent of the tree code in use, and thus it is valid for simulation with potent tree as well.

Last, we show the following lower bound on the number of errors.

Lemma IV.4 (replacing lemma 6 of [7])**.** *For $i \geq -1, k \geq 0$, if $i + 1 < t(k)$, then $\mathcal{N}(1, i, d) \geq (i - k + 1)(1-\epsilon_2)/2 - d$*

Proof: We prove by induction. $\mathcal{N}(1, i, d) = \mathcal{N}(1, m(i), x) + \mathcal{N}(m(i)+1, i, d-x)$ assuming that the total length of the imposed bad-intervals between rounds 1 and $m(i)$ (that is, along the paths $a_1, \ldots, a_{m(i)}$ and $b_1, \ldots, b_{m(i)}$) is exactly x, $0 \leq x \leq d$. Lemma IV.2 guarantees that $\mathcal{N}(m(i)+1, i, d-x) \geq (1-\epsilon_2)(i-m(i))/2 - (d-x)$. By Lemma IV.3, $m(i) < t(k-1)$ and we can use the induction hypothesis on the first part, which gives

$$\mathcal{N}(1, m(i)-1, x) \geq ((m(i)-1) - (k-1) + 1)(1-\epsilon_2)/2 - x \ .$$

Summing these two bounds proves the lemma. ∎

Note that the in the case of an absolute tree code, $d = 0$, which gives exactly Lemma 6 of [7]. With a potent tree, $d \leq 2\epsilon_1 N$ which reduces the maximal error rate by $2\epsilon_1$.

In a similar way Lemma 8 of [7] can be adapted to potent trees, which completes the proof of Theorem IV.1, by setting $\epsilon \geq \epsilon_1/2 + \epsilon_2$. ∎

B. Efficient Simulation with Random Errors

In the case of a noisy channel, our simulation (based on potent tree-codes) can also be implemented efficiently

and fails only with exponentially-small probability. In 1992 Schulman proposed an efficient randomized scheme that solves this problem [2] in an *alternative* non-standard model which requires a quadratic number of shared randomness between parties ahead of time[6]. By using potent trees (realized via SBTCs), we improve the result of Schulman and obtain a linear communication (i.e., a constant dilation) which includes the communication required to agree on the same SBTC. The scheme we obtain is efficient and constructive. We then extend our proof to any multiparty protocol following the analysis of Rajagopalan and Schulman [6], again, by replacing the absolute tree code with a potent tree.

Our simulation follows the method of Schulman [3], [11] (See model and full details in the Appendix). We prove the following results.

Theorem IV.5. *There exists a randomized simulation that runs in expected polynomial time and simulates a 2-party protocol π of length T over a BSC. The length of the simulation is always at most $O(T)$, and the simulation succeeds with probability $1-2^{-\Omega(T)}$ over the channel errors and the choice of the SBTC.*

We extend our result to the m-party case [6], and show the following.

Theorem IV.6. *There exists a constructible and efficient simulation that computes any n-party protocol π of length T using a BSC for communication. The simulation succeeds with probability $2^{-\Omega(T)}$, and imposes a dilation of $O(m)$.*

See proof in the Appendix.

Rajagopalan and Schulman [6] give a dilation of $O(\log(r+1))$ where $r \leq m$ is the maximal level of connectivity. In the following Section V we give a construction of more "potent" potent trees, and achieve an improved $O(\log(r+1))$ dilation too, yet the failure probability increases to $2^{-\Omega(T/m)}$.

V. Greater Potency and Improved Multiparty Protocols

In this section we give a construction of a d-ary (ε,α)-potent tree of depth N which, for a constant α and an arbitrary ε, requires a constant-size alphabet and fails with probability $2^{-\Omega(\varepsilon N)}$.

Theorem V.1. *For a constant $\alpha \in (0,1)$ and arbitrary ε, there exists an efficient and explicit construction of a d-ary (ε,α)-potent tree-code of depth N over a constant-size alphabet S, such that the labels of any two divergent paths of length $k \leq N$ are almost k-independent. The construction succeeds with probability at least $1-2^{-\Omega(\varepsilon N)}$.*

If we set $\varepsilon = O(1/m)$, then this construction along with the analysis of Theorem IV.6 immediately yields the following result.

[6]The trivial way to convert this protocol to the standard model without shared randomness would be to have one user send this shared randomness to the other. However, no efficient derandomization is known so far, although Schulman gave an inefficient method to reduce the number of bits required to linear in the depth, sacrificing efficiency of the simulation.

Theorem V.2. *There exists a constructible and efficient simulation that computes any m-party protocol π (in which the maximum connectivity is r) of length T over a BSC. The simulation achieves dilation $O(\log(r+1))$ and takes $O(T)$ rounds. The simulation succeeds with probability at least $1-2^{-\Omega(T/m)}$.*

Proof: (**Theorem V.1**) Our construction consists of two parts. In the first part we build a "weak" tree-code $\mathcal{T}_1$ in which each two paths of length at most $\log N$ satisfy the distance property. Such a tree can easily be constructed using an efficient deterministic method by Schulman [5]: we find an absolute tree code of depth $2\log N$ over some constant size alphabet S_1, and then, starting at depth $\log N$ we overlap another copy of the absolute tree code at every depth that is a multiple of $\log N$, so that each label of $\mathcal{T}_1$ is a concatenation of the labels of the two overlapping absolute tree code.

In the second part we construct a linear-SBTC $\mathcal{T}_2$ in the following way. Consider the small-biased random[7] lower-triangular matrix $G_{N\times N}$ over some finite field $\mathbb{F}$, say of characteristic at least d. We will set our second alphabet $S_2 = \mathbb{F}$. The labels assigned to the path $a_1, a_2, a_3, \ldots, a_N$, where $a_i \in \{0,1,\ldots,d-1\}$, are given by $G\cdot(a_1,a_2,\ldots,a_N)^{\mathsf{T}}$. It is easy to verify that each label does not depend on the path beneath, which makes this construction a valid tree code.

We get our final tree code $\mathcal{T}$ over $S = S_1 \times S_1 \times S_2$, by concatenating, for each arc, the label of the matching arc in $\mathcal{T}_1$ and in $\mathcal{T}_2$. Note the following properties: any two divergent paths in $\mathcal{T}$ which do not satisfy the distance property must be of length at least $\log N$. Moreover, due to the linearity of $\mathcal{T}_2$, it is clear that there exists an α-bad path at depth d of length ℓ if and only if the path $\bar{0} = (0,0,\ldots,0)$ has an α-bad path contained within it at the same depth d and of the same total length ℓ. This means that in order to bound the probability of $\mathcal{T}$ to be α-bad we only need to analyze bad paths of length at least $\log N$ where the common ancestor lies on the $\bar{0}$ path. Last, note that for any two divergent paths of length $k \leq N$ in $\mathcal{T}_2$, the labels are almost k-independent (however, in contrast to the SBTC, this is not the case for labels which are not on two divergent paths).

We now show that a constant alphabet size is sufficient for $\mathcal{T}$ to be (ε,α)-potent with probability $1-2^{-\Omega(\varepsilon N)}$. Assume that $\mathcal{T}$ is α-bad, that is, the $\bar{0}$ path have bad sub-paths of total length at least εN (and thus, this path is α-bad both in $\mathcal{T}_1$ and $\mathcal{T}_2$. We focus on $\mathcal{T}_2$ for the rest of this analysis). Using Lemma III.6, there exists bad sub-paths $l_1, l_2, \ldots, l_n$ of total length $\sum_i l_i > \varepsilon N/2$. Note that for any i, we have that $l_i > \log N$, thus we can consider only $n \leq \varepsilon N/2\log N$. Trivially, there are at most $N^{2n} \leq 2^{\varepsilon N}$ ways to distribute the intervals along the $\bar{0}$ path. As analyzed above (see Corollary III.5), the probability that a node u, that imposes the fixed i^{th} interval of length l_i, exists is at most $2(2d/|S_2|^{1-\alpha})^{l_i}$.

$$\Pr[\mathcal{T} \text{ is } \alpha\text{-bad }] \leq \sum_{\substack{l_1,l_2,\ldots \text{ disjoint,}\\ \text{of length}\geq\varepsilon N/2}} \prod_i 2(2d/|S_2|^{1-\alpha})^{l_i}$$

[7]That is, the $O(N^2\log|\mathbb{F}|)$ bits required to define G are drawn from a small-biased sample space B_n with bias $2^{-O(N\log|\mathbb{F}|)}$.

$$\leq 2^{\varepsilon N} 2^n (2d/|S_2|^{1-\alpha})^{\sum_i l_i} \leq (16d/|S|^{1-\alpha})^{\varepsilon N/2},$$

thus, for a constant alphabet $|S_2| > (16d)^{1/(1-\alpha)}$, the claim holds. ∎

ACKNOWLEDGMENTS

We would like to thank Leonard Schulman and Anant Sahai for many useful discussions during this research. We also thank Madhu Sudan, David Zuckerman, and Venkatesan Guruswami for several helpful conversations. We would like to thank Alan Roytman for miscellaneous remarks.

REFERENCES

[1] C. E. Shannon, "A mathematical theory of communication," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 5, no. 1, pp. 3–55, 2001, originally appeared in *Bell System Tech. J.* 27:379–423, 623–656, 1948.

[2] L. J. Schulman, "Communication on noisy channels: a coding theorem for computation," *Foundations of Computer Science, Annual IEEE Symposium on*, vol. 0, pp. 724–733, 1992.

[3] ——, "Deterministic coding for interactive communication," in *STOC '93: Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*. New York, NY, USA: ACM, 1993, pp. 747–756.

[4] N. Alon, O. Goldreich, J. Håstad, and R. Peralta, "Simple constructions of almost k-wise independent random variables," *Random Structures & Algorithms*, vol. 3, no. 3, pp. 289–304, 1992.

[5] L. J. Schulman, "Postscript to "coding for interactive communication"," 2003, schulman's homepage. http://www.cs.caltech.edu/~schulman/Papers/intercodingpostscript.txt.

[6] S. Rajagopalan and L. Schulman, "A coding theorem for distributed computation," in *STOC '94: Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*. New York, NY, USA: ACM, 1994, pp. 790–799.

[7] M. Braverman and A. Rao, "Towards coding for maximum errors in interactive communication," in *Proceedings of the 43rd annual ACM symposium on Theory of computing*, ser. STOC '11. New York, NY, USA: ACM, 2011, pp. 159–166.

[8] M. Peczarski, "An improvement of the tree code construction," *Information Processing Letters*, vol. 99, no. 3, pp. 92–95, 2006.

[9] M. Braverman, "Towards deterministic tree code constructions," Electronic Colloquium on Computational Complexity (ECCC) TR11-064, 2011.

[10] R. Ostrovsky, Y. Rabani, and L. J. Schulman, "Error-correcting codes for automatic control," in *FOCS*, 2005, pp. 309–316.

[11] L. J. Schulman, "Coding for interactive communication," *IEEE Transactions on Information Theory*, vol. 42, no. 6, pp. 1745–1756, 1996.

[12] R. Gelles and A. Sahai, "Potent Tree Codes and their applications: Coding for Interactive Communication, revisited," Apr. 2011, arXiv:1104.0739 (cs.DS).

[13] J. Naor and M. Naor, "Small-bias probability spaces: efficient constructions and applications," in *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, ser. STOC '90. New York, NY, USA: ACM, 1990, pp. 213–223.

[14] U. V. Vazirani, "Randomness, Adversaries and Computation," Ph.D. dissertation, EECS, UC Berkeley, 1986.

[15] J. M. Wozencraft, "Sequential decoding for reliable communication," Massachusetts Institute of Technology, Tech. Rep. 325, 1957.

[16] B. Reiffen, "Sequential encoding and decoding for the discrete memoryless channel," Research Laboratory of Electronics. Massachusetts Institute of Technology, Tech. Rep. 374, 1960.

[17] R. M. Fano, "A heuristic discussion of probabilistic decoding," *IEEE Transactions on Information Theory*, vol. 9, no. 2, pp. 64–74, 1963.

APPENDIX

A. Interactive Protocol Over Noisy Channels

We consider a distributed computation of a fixed function f, performed by two users who (separately) hold the inputs. Let π be a 2-party distributed protocol which on inputs x_A, x_B, both parties output the value $f(x_A, x_B)$. In each round, A and B send a single message to each other, based on their input and messages previously received. The protocol π assumes an ideal communication channel which contains no errors. Under these assumptions, π takes T rounds of communication to output the correct answer, where one round means both users simultaneously send each other a message.

In a more realistic model, the channel between A and B may be noisy, so that each message needs to be encoded in order to identify and correct possible errors. Shannon's *Coding Theorem* [1] (see Lemma II.1) shows that an exponentially small decoding error in the length of the message $|m|$ can be achieved, if the message is encoded into a code word of length $c|m|$, for some constant c determined by the channel *capacity*. However, if we use a standard Shannon code to encode multiple messages, then the probability of having at least a single decoding error is proportional to the number of messages sent, rather than arbitrarily small. In this paper we explore the worst case scenario of the above tradeoff between the number of messages and their size. Namely, we assume that a total amount of T bits of information is divided into T messages of a single bit each. Our aim is to send $O(T)$ bits over the channel and obtain an exponentially small failure probability. We assume that in each round the users send only a single bit, which is the worst case in terms of the communication complexity. Indeed, if many bits are grouped into a single message they can be coded in a more efficient way.

Let us formalize this model of interactive communication and the associated protocol π. During each round, each user $i \in \{A, B\}$ sends one bit according to its input x_i and the messages received so far. Let $\pi_i(x_i, \emptyset)$ denote the first bit sent by user i, and let $\pi(x, \emptyset) \in \{00, 01, 10, 11\}$ be the two bits transmitted in the first round by A and B respectively, where $x = x_A x_B$. Generally, let $m_1, \ldots, m_t$ be the first t two-bit messages exchanged during the protocol, then the information sent in round $t+1$ is defined by $\pi(x, m_1 \ldots m_t)$.

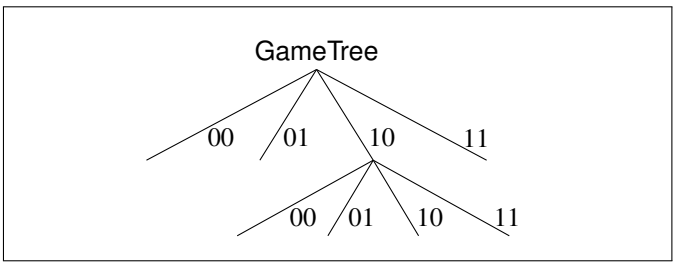

Fig. 4. An illustration of the GameTree.

The computation (over a noiseless channel) can be described as a single route $\gamma_{\pi,x}$ along the GameTree, a 4-ary tree of depth T (see Figure 4). The path $\gamma_{\pi,x}$ begins at the root of the tree and the t^{th} edge is determined by the 2 bits exchanged in the t^{th} round, i.e., the first edge in the path is $\pi(x, \emptyset)$, the second is $\pi(x, \pi(x, \emptyset))$, etc. Also, for a vertex $v \in$ GameTree, let $\pi_i(x_i, v)$ be the bit transmitted by user i at some round $t+1 = depth(v)+1$ if the information received in the previous t rounds is the labels along the path from root to v.

B. Simulating π Over a Noisy Channel

Our goal is to simulate a run of π over a noisy channel. In order to do so, we use the method of Schulman [11]. The idea behind the simulation is the following. Each user keeps a record of (his belief of) the current progress of π, described as a pebble on one of the GameTree nodes.

Each round, according to the transmissions received so far, the user makes a guess for the position of the other user's pebble, and infers how his own pebble should move. The user sends a message that describes the way he moves his pebble (out of six possible movements corresponding to the 4 child nodes, 'H' to keep the pebble at the same place or 'B' to back up to the parent node) and his output bit according to his current position on the GameTree. Each one of these 12 options represents a child in a 12-ary tree denoted as the StateTree (Figure 5).

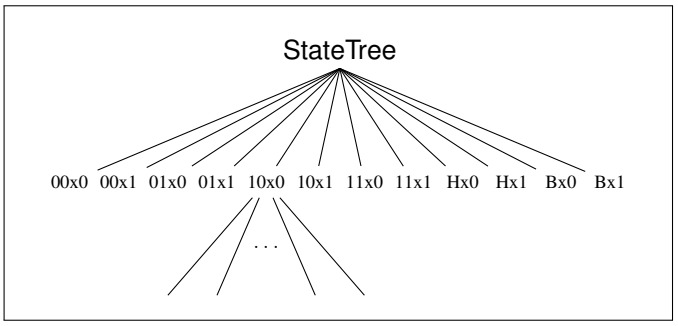

Fig. 5. An illustration of the StateTree.

The user communicates[8] the label assigned to the edge in the StateTree that describes his move. The *state* of the user is the current node on the StateTree, starting from its root, and changing according to the edge communicated. The protocol is given in Figure 6 (described for user A; the protocol for B is identical).

Informally speaking, the simulation works since the least common ancestor of both users' pebbles always lie along the path $\gamma_{\pi,x}$. If both users take the correct guess for the other user's pebble position, they simulate π correctly and their pebbles move along $\gamma_{\pi,x}$. Otherwise, their pebbles diverge, yet the common ancestor remains on $\gamma_{\pi,x}$. On the following rounds, when the users acknowledge an inconsistency in the pebbles' positions, they move their pebbles backwards until the pebbles reach their common ancestor, and the protocol continues. Repeating the above process for $N = O(T)$ rounds is sufficient for simulating π with exponentially small error probability (over the channel errors). We refer the reader to [11] for a detailed description of the protocol and its analysis.

We now replace the (non-constructive) absolute tree code originally used by Schulman by a potent tree, and show that the simulation still succeeds with overwhelming probability. Moreover, if we are given an oracle to a tree code decoding procedure, the obtained protocol is efficient.

[8]We imply here using a (standard) error-correcting code in order to send the label over the noisy channel, with constant slowdown (as given by Lemma II.1). Throughout the paper, any transmission of a label is to be understood in this manner.

[9]For the simulation to be well defined, we must extend π to N rounds. We assume that in each of the $N - T$ spare rounds, π outputs 0 for each user and every input.

Begin with own pebble at the root of GameTree and own state S_A at the StateTree root's child labeled $H \times \pi_A(x_A, \emptyset)$. Repeat the following N times[9]:

1) Send $w(S_A)$ to user B.
2) Given the sequence of messages Z received so far from user B, guess the current state g of B as the node that minimizes $\Delta(W(g), Z)$. From g, infer B's assumed pebble position, pebble(g), as well as B's assumed message $b = \pi_B(x_B, \text{pebble}(g))$
3) Set own pebble's movement and new state according to the current position v of your pebble:
   a) If $v = \text{pebble}(g)$ then move pebble according to the pair of bits $(\pi_A(x_A, v), b)$ to a state v'. The new state is S_A's child labeled with the arc $(\pi_A(x_A, v), b) \times \pi_A(x_A, v')$.
   b) If v is a strict ancestor of pebble(g): own movement is H, and the next state is along the arc $H \times \pi_A(x_A, v)$.
   c) Otherwise, move pebble backwards. New state is along the arc $B \times \pi_A(x_A, v')$ where v' is the parent of v.

Fig. 6. Interactive protocol Sim_π for noisy channels [11].

Theorem A.1. *Given a $(\frac{1}{10}, \alpha)$-potent tree code with a constant-size alphabet $|S|$, an oracle for a decoding procedure of that tree code, and a protocol π of length T, the protocol Sim_π (Figure 6) efficiently simulates π, takes $N = O(T)$ rounds and succeeds with probability $1 - 2^{-\Omega(T)}$ over the channel errors, assuming an error correcting code with (label) error probability $p < 2^{-42/\alpha}$.*

In Section C we show a decoding procedure that is efficient on average, given that the tree is SBTC. This immediately leads to the following Theorem.

Theorem A.2. *There exists an efficient simulation that computes any distributed 2-party protocol π of length T, using a BSC for communication and a pre-shared SBTC. The simulation imposes a constant slowdown, and succeeds with probability $1 - 2^{-\Omega(T)}$ over the channel errors and the choice of the SBTC.*

We now give the proof idea for Theorem A.1 and later complete the formal proof. We begin by defining a good move: a move that advances the simulation of π in one step, and a bad move: an erroneous step in the simulation that requires us to back up and re-simulate that step. We show that any bad move is associated with a decoding error, i.e., recovering a wrong node u, due to channel errors or tree defects. This allows us to bound the number of bad moves by bounding the probability for channel errors and tree defects. The latter is exponentially small due to the properties of Shannon codes and the properties of potent trees.

Recall the following properties of the simulation Sim_π.

Lemma A.3. *The least common ancestor of the two pebbles lies on $\gamma_{\pi,x}$.*

Lemma A.4. *Let v_A and v_B be the positions of the two pebbles in the* GameTree *at some time t, and let $\bar{v}$ denote the least common ancestor of v_A and v_B. Define the* mark *of the protocol as the depth of $\bar{v}$ minus the distance from $\bar{v}$ to the further of v_A and v_B.*

If during a specific round, both users guess the other's state correctly (a good *move), the mark increases by 1. Otherwise (a* bad *move), the mark decreases by at most 3.*

Proofs for both the above lemmas are given in [11].

Our goal is to show that the probability of having more than cN bad rounds is exponentially small. By setting $c = 1/5$ and $N = 5T$ we guarantee that at the end of the calculation the mark will be (at least) T. Since the common ancestor of the pebbles always lies along the path $\gamma_{\pi,x}$, a mark of value T indicates that the common ancestor has reached depth T, and π was successfully simulated.

For a bad round at time t, we assume that (at least) one of the users takes a wrong guess of the (other user's) current state. Suppose that the least common ancestor of the right state and the wrongly guessed state in the StateTree, are distanced l levels away (i.e., an error of magnitude l). Define the *error interval* (of length l) corresponding to the erroneous guess as $[t-l, t]$.

We now show that given a potent tree, Sim_π simulates π over a noisy channel with overwhelming probability.

Proof: **(Theorem A.1)**. Suppose the parties share a $(\frac{1}{10}, \alpha)$-potent tree code[10], for some $0 < \alpha < 1$. Assume that a specific run of a simulation failed, and thus it must be that more than $N/5$ errors have occurred.

Note that the (failed) simulation defines a path from the root of the StateTree to one of its leaves. This path contains bad intervals of total length at most $N/10$. We assume a worst case scenario in which each α-bad node causes a bad move in the simulation. In that case, the probability to have $N/10$ additional bad moves in the remaining nodes, is exponentially small.

Consider a specific bad move at time t caused by erroneously decoding a node which is not α-bad. Namely, the user guesses a wrong node r instead of the real transmitted node s. For an error of magnitude l_i, $W(s)$ and $W(r)$ are identical from the root to the least common ancestor of r and s at level $t-l$. Since the decoding is done by minimizing the Hamming distance, making such a wrong guess is independent of transmissions prior to round $t-l$. It follows that such an error (of magnitude l) can happen only if at least $\alpha l/2$ channel errors have occurred during the last l rounds. Due to the same reason, it is easy to see that decoding errors of which the error intervals are disjoint, are independent.

We consider again the bad moves at nodes which are not α-bad. Each such a bad move (i.e., a decoding error) imposes an error interval of length $l_i > 1$, such that the length of the union of these intervals is at least $N/10$. Each such an error happens with probability at most $\sum_{j=\alpha l_i/2}^{l_i} \binom{l_i}{j} p^j \leq 2^{l_i} p^{\alpha l_i/2}$. Due to Lemma III.6 we can find a set of disjoint intervals of length at least $N/20$. Since the intervals are disjoint, these errors are independent, and their probability to jointly occur is bounded by

$$\prod_i 2^{l_i} p^{\alpha l_i/2} = (2p^{\alpha/2})^l.$$

We conclude the proof by bounding the probability for having any possible error pattern of total length at least $N/20$ along the bad moves associated with nodes which are not α-bad, by using a union bound over all the $\sum_{j=N/20}^{N} \binom{N}{j} \leq 2^N$ possible error patterns, for each one of the users. The probability is bounded by

$$\sum_{\text{user } U} \sum_{\substack{\text{pattern of} \\ l \geq N/20 \text{ errors}}} (2p^{\alpha/2})^l \leq 2 \cdot 2^N (2p^{\alpha/2})^{N/20},$$

which is $2^{-\Omega(N)} = 2^{-\Omega(T)}$ for $p < 2^{-42/\alpha}$. ∎

C. Efficient Decoding

A decoding process outputs a node u (at depth t) that minimizes the Hamming distance $\Delta(W(u), \mathbf{r})$, where $\mathbf{r} = r_1 r_2 \cdots r_t$ is the received string of labels. Although the above Theorem A.1 is proven assuming an oracle to tree-code decoding procedure, this requirement is too strong for our needs. Since we count any node which is α-bad as an error (even when no error has occurred), it suffices to have an oracle that decodes correctly given that the (transmitted) node is not α-bad.

We follow techniques employed by Schulman [11] (which are based on ideas from [15], [16], [17]), and show an efficient decoding that succeeds if the node is not α-bad. While the decoding process of [11] is based on the fact that the underlying tree is an absolute tree code, in our case the tree code is a SBTC.[11]

The decoding procedure is the following. For a fixed time t, let g_{t-1} be the current guess of the other user's state, and denote the nodes along the path from root to g_{t-1} as $g_1, g_2, \ldots, g_{t-1}$. Set g_t to be the child node of g_{t-1} along the arc labeled with r_t, if such exists (break ties arbitrarily). Otherwise, set g_t as an arbitrary child of g_{t-1}.

Recall that $W_m(u)$ denotes the m-suffix of $W(u)$, i.e., the last m labels along the path from the tree's root to u. We look at the earliest time i such that $\Delta(r_i r_{i+1} \cdots r_t, W_{t-i+1}(g_t)) \geq \alpha(t-i)/2$. For that specific i, exhaustively search the subtree of g_i and output the node u (at depth t) that minimizes the Hamming distance $\Delta(r_1 r_2 \cdots r_t, W(u))$.

Note that when g_t is an α-bad node of maximal length l, any path from root to some other node g'_t, where the least common ancestor of g_t and g'_t is located $l' > l$ levels away, must have a Hamming distance $\Delta(W_{l'}(g_t), W_{l'}(g'_t)) \geq \alpha l'$. Therefore, if all the suffixes of length $l' > l$ satisfy $\Delta(r_{t-l'+1} \cdots r_t, W_{l'}(g_t)) < \alpha l'/2$, we are guaranteed to find the node minimizing the Hamming distance within the subtree of g_{t-l}. On the other hand, it is possible that the decoding procedure outputs a node u which is a descendant of g_{t-l},

[10] Theorem III.3 guarantees that as long as $|S| > (2d)^{22/(1-\alpha)}$, only exponentially-small fraction of the SBTCs are $(\frac{1}{10}, \alpha)$-bad. Therefore, for obtaining a potent tree with overwhelming probability, we require $\log|S| \geq 101$.

[11] A similar proof works also for a RTC, see [12].

yet does not minimize the Hamming distance. This happens when the decoding procedure explores a smaller subtree, i.e., $i > t - l$.

The following proposition bounds the probability for a decoding error of magnitude l.

Proposition A.5. *Assume a SBTC is used to communicate the string $W(v)$ over a BSC. Using the efficient decoding procedure (with some constant $\alpha \in (0,1)$), the probability for a specific user to make a decoding error of magnitude l is bounded by $2\left(\frac{4d}{|S|}\right)^l + 2\left(\frac{2d}{|S|^{1-\alpha}}\right)^l$, assuming an error correction code with (label) error probability $p < |S|^{-2}$.*

Proof: A decoding error of magnitude l occurs if the decoding process outputs a node $u \neq v$, such that the common ancestor of u, v is l levels away. Such an error can happen due to one of the following reasons:

(i) For the received string $\mathbf{r} = r_1 r_2 \dots r_l$ it holds that $\Delta(\mathbf{r}, W(u)) \leq \Delta(\mathbf{r}, W(v))$. This happens when the Hamming distance $\Delta(W(u), W(v))$ is $j = 0, 1, \dots, l$ and more than $j/2$ channel errors occurred.

(ii) The decoding process did not return the node that minimizes the Hamming distance.

Note that we only need to consider the paths from root to u and to v and thus use the $2N$-wise independence of the tree's labels. Recall that the probability to have a specific set of $l < 2N$ labels is $2^{-cN \log |S|}$ away from uniform with $c = O(1)$, and that the probability for a given Hamming distance between $W(u)$ and $W(v)$ is bounded by Lemma III.4. Let $p < |S|^{-2}$ be the maximal label error of the channel, and for $i \in \mathbb{N}$ let $\mathcal{E}_i$ be the event that *at least* i channel (symbol) errors have occurred. Using a union bound for every possible node u, the probability of part (i) is bounded by

$$\begin{aligned}
&\Pr[\text{ Error of magnitude } l \text{ }] \\
&\leq \sum_u \sum_{j=0}^{l} \Pr[\Delta(W(v), W(u)) = j] \Pr[\mathcal{E}_{j/2}] \\
&\leq d^l \sum_{j=0}^{l} 2\binom{l}{l-j} \left(\frac{1}{|S|}\right)^{l-j} \sum_{k=j/2}^{l} \binom{l}{k} p^k (1-p)^{l-k} \\
&\leq 2 \cdot d^l \sum_{j=0}^{l} \binom{l}{l-j} \sum_{k=j/2}^{l} \binom{l}{k} |S|^{j-l} |S|^{-2k} \\
&\leq 2 \cdot d^l \cdot 2^l \cdot |S|^{-l} ,
\end{aligned}$$

which is exponentially small in l as long as $|S| > 4d$.

For part (ii), note that the decoding process does not output the node that minimizes the Hamming distance if $l > t - i$, for i determined by the decoding procedure. For the output node u, $\Delta(r_{t-l+1} \cdots r_t, W_l(u)) < \alpha l/2$. However, since u does not minimize the Hamming distance, there must exist a node z of distance at most l, such that $\Delta(W_l(z), r_{t-l+1} \cdots r_t) \leq \Delta(r_{t-l+1} \cdots r_t, W_l(u))$. By the triangle inequality, $\Delta(W_l(z), W_l(u)) \leq \alpha l$. Using the union bound for any possible z and any possible Hamming distance up to αl, we bound the probability of this event by

$$d^l \sum_{j=0}^{\alpha l} 2\binom{l}{l-j} |S|^{-(l-j)} \leq 2(2d)^l |S|^{-l(1-\alpha)} .$$

A union bound on the two cases completes this proof. ∎

We stress that the above decoding process always outputs the correct node (i.e., the node that minimizes the Hamming distance), if the transmitted node is not α-bad. For that reason, the proof of Theorem A.1 is still valid, since it only requires the decoding procedure to succeed when the node is not α-bad (and assumes that the simulation has a bad move in each node which is a bad node).

We now show that this procedure is efficient in expectation. Let $L(t)$ be the depth of the subtree explored at time t. The decoding process takes $O\big(\sum_{t=1}^{N} d^{L(t)}\big)$ steps (this dominates terms of $O(L(t))$ required to maintain the guess, etc).

For a time t, if $L(t) = l$ then $\Delta(r_{t-l+1} \cdots r_t, W_l(g_t)) \geq \alpha l/2$ yet for $l' > l$, $\Delta(r_{t-l'+1} \cdots r_t, W_{l'}(g_t)) < \alpha l'/2$, thus $\Delta(r_{t-l+1} \cdots r_t, W_l(g_t)) = \lceil \alpha l/2 \rceil$. Let the transmitted sequence of labels be $W(v)$ for some node v of depth t. A Hamming distance of exactly $\lceil \alpha l/2 \rceil$ happens with probability at most

$$\begin{aligned}
&\leq \sum_{j=0}^{l} \Pr[\Delta(W_l(g_t), W_l(v) = j] \Pr[\mathcal{E}_{|\lceil \alpha l/2 \rceil - j|}] \\
&\leq \sum_{j=0}^{l} 2\binom{l}{l-j} \left(\frac{1}{|S|}\right)^{l-j} \sum_{k=|\lceil \alpha l/2 \rceil - j|}^{l} \binom{l}{k} p^k (1-p)^{l-k},
\end{aligned}$$

which is bounded by $2^{2l+1} |S|^{-l(1-\alpha/2)}$ for $p < |S|^{-2}$.

With a sufficiently large yet constant alphabet, e.g., $|S| > (8d)^{1/(1-\alpha/2)}$, we bound the probability that $L(t)$ equals l to be $2^{-\gamma l} < d^{-l}$. The expected running time is then given by $O\Big(\sum_{t=1}^{N} E\Big[d^{L(t)}\Big]\Big)$, which equals

$$O\Big(\sum_{t=1}^{N} \sum_{l=0}^{t} \Big[2^{-\gamma l} d^l\Big]\Big) = O\Big(\sum_{t=1}^{N} \frac{2^\gamma}{2^\gamma - d}\Big) = O(N).$$

We repeat the simulation step for $N = O(T)$ times, and the computation is efficient in expectation. Finally, we mention that [11] presents a data structure which allows us to perform the above decoding procedure with overhead $O(L(t))$.

D. Simulating m-Party Protocols

In this section we extend our result to support a simulation of a protocol π with any number m of users. This is done by incorporating the tools described in the previous sections with the method of simulating an m-party protocol over a disturbed channel developed by Rajagopalan and Schulman [6]. The paper [6] shows that a scheme for simulating multiparty protocol over a disturbed channel *exists*, yet the question of its efficient implementation has been open since 1994. The Scheme presented there obtains a communication dilation of $O(\log(r+1))$ where r is the maximal connectivity degree, that is, the maximal number of parties connected to a specific user.

Rajagopalan and Schulman describe how to adapt the 2-party simulation of [11] to an arbitrary number of users. The key idea is to replace the 12-ary StateTree with a ternary tree (that is, $d = 3$), where each node has three child nodes marked with $\{0, 1, bkp\}$. The values 0 and 1 indicate the output bit of the user in the simulated round, and bkp indicates that the last simulated round is suspected to be invalid and should be deleted and re-simulated. The simulation (described here for a specific user i) is completely defined by the following process. Each round, the user uses all the previous communications to infer the current simulated round of π and sends his output bit to user j (by communicating the label assigned with arc to child 0 or 1 respectively, in the ternary StateTree shared between users i and j). If the user finds an inconsistency, he transmits bkp which denotes deleting the last received (undeleted) bit and rolling the protocol π one step back. The user shares such a ternary tree with each of the r parties connected to him, and is allowed to output a different bit to each party. Yet, when the user decides to roll back he outputs bkp on each of the outgoing links. Inconsistency is defined as one of the two following cases: (1) the current decoded transcript of the StateTree disagrees with the bits sent so far, or (2) the user received bkp from one of his neighbors. We refer the reader to [6] for a complete description and analysis of this scheme.

One can easily check that the bulk of the analysis performed in [6] applies for the case of replacing the absolute ternary tree code with a ternary SBTC (or RTC). The analysis is composed of two parts. The first part shows that if after t rounds the scheme simulates step $t - l$ of π then at least $l/2$ errors have occurred in decoding the correct tree-node during the *history cone* of the user at time t (i.e., all the transmissions that affect the user state at time t). The other part bounds the probability of having a constant fraction of errors (out of the number of rounds). While the first part is completely independent of the fact that we replace the absolute tree code with a SBTC, in order to complete the proof, we must adapt the second part to the usage of SBTC. This is done by Lemma A.8 below.

Let us formally describe these two parts. We begin by defining the notion of the history cone [6]. Let (p, t) denote a user p at time t.

Definition 9 ([6])**.** *(p, t) and (p', t') are **time-like** if messages sent by user p at time t has an affect on the computation of user p' at time t' (or vice versa).*

That is, (p, t) and (p, t') are always time-like, and (p, t) and $(q, t + 1)$ are time-like if p and q are neighbors.

Definition 10 ([6])**.** *A t **time-like path** is a sequence $\{(p_i, i) \mid 1 \le i \le t\}$ such that any two elements in the path are time-like (i.e., for every i, p_i and p_{i+1} are either neighbors or the same party).*

The proof of [6] follows from the next two lemmas

Lemma A.6 (Lemma (5.1.1) of [6])**.** *If a user p at time t has successfully simulated only the first $t - l$ rounds of π, then there is a t time-like sequence that ends at (p, t) and includes at least $l/2$ tree-decoding errors.*

Lemma A.7 (Lemma (5.1.2) of [6])**.** *Using error correcting codes with dilation $O(\log(r + 1))$, the probability that any fixed t time-like path has more than $t/4$ tree-decoding errors, is less than $\frac{1}{(2(r+1))^t}$.*

The proof of the multiparty case is given by setting $t = N = 2T$. The first lemma states that if the simulation failed (the first $N/2$ rounds of π are not valid for some user) then there must exist one user who has $N/4$ errors along one of his N time-like sequences. The probability of this event is bounded by the Lemma A.7 to be less than $\frac{1}{(2(r+1))^t}$ summed over all the $N(r + 1)^N$ possible time sequences, which is bounded by $N2^{-N}$.

While the above Lemma A.6 holds regardless of the tree in use, we prove a variant of the above Lemma A.7 for the case of using a potent tree. Moreover, although Lemma A.7 holds for any time $1 \le t \le N$, only $t = N$ is required for completing the proof for the multiparty case, which we prove in the following lemma.

Lemma A.8. *Suppose each two users share a $(\frac{1}{16m}, \alpha)$-potent tree, for some $\alpha \in (0, 1)$. If an error correcting code with label error probability p is used, then for any fixed N time-like path, the probability that there are more than $N/4$ tree-decoding errors is bounded by $(2^{17} p^{\alpha/2})^{N/16}$, over the errors of the channel.*

Proof: We assume an oracle for the decoding process, which can easily be replaced by the *efficient decoding* procedure given in Section C, if we use a SBTC. Assume that at least $N/4$ errors have occurred in a specific N time-like path. Fix a specific user i and assume that the errors of this user are included in error intervals of total length l_i. By Lemma III.6, there exist disjoint error intervals of total length at least $l_i/2$. Recall that each error interval of length ℓ corresponds to an error of magnitude ℓ, and recall that in each tree, at most $N/16m$ of the nodes are α-bad. Thus, at least $k_i \equiv \max\{0, l_i/2 - N/16m\}$ of the errors of user i in the N time-like path occur in nodes which are not on an α-bad interval. These errors can only be originated due to channel errors[12], and since the intervals are disjoint, they are independent. As above (see proof of Theorem A.1), the probability of having errors that correspond to these (fixed) disjoint error intervals is bounded by $2^{k_i} p^{\alpha k_i/2}$. Clearly, tree-decoding errors of a specific user are independent of the communication (and channel errors) of other users. It follows that the probability for all the users to have a total amount of $N/4$ errors matching the fixed intervals pattern is bounded by $(2p^{\alpha/2})^{\sum_i k_i}$. With $\sum_i l_i > N/4$ and at most n users, this probability is bounded by $(2p^{\alpha/2})^{N/8 - m(N/16m)} = (2p^{\alpha/2})^{N/16}$.

Using a union bound we sum the probability over any number $j \ge N/4$ of errors and over any one of the $\binom{N}{j}$ different ways to distribute j errors along the fixed time-like path. The probability that there are at least $N/4$ errors in this

[12]This claim also applies to the efficient decoding procedure, as it always returns the node that minimized the Hamming distance, if it is not α-bad. See the proof of Theorem A.1 and discussion in Section C.

fixed N time-like path is bounded by

$$\sum_{j=N/4}^{N} \binom{N}{j} (2p^{\alpha/2})^{j/2-N/16} \leq (2^{17}p^{\alpha/2})^{N/16} \ .$$

■

For $p < (5(r+1))^{-32/\alpha}$, this probability is at most $\frac{1}{(2(r+1))^N}$.

Corollary A.9. *Suppose each two users share a* SBTC[13] *with* $|S| \geq ((2d)^{32m+2})^{1/(1-\alpha)}$ *for some* $\alpha \in (0,1)$*, and use an error correcting code with (label) error probability less than* $p \leq (5(r+1))^{-32/\alpha}$*. Then, except with probability* $2^{-\Omega(N)}$ *over the choice of the* SBTC*, for any fixed* N *time-like path, the probability that there are more than* $N/4$ *tree-decoding errors is less than* $\frac{1}{(2(r+1))^N}$ *over the channel errors.*

That is, with $|S| \geq ((2d)^{32m+2})^{1/(1-\alpha)}$ the SBTC is $(\frac{1}{16m}, \alpha)$-potent, with overwhelming probability, due to Theorem III.3. Each label in an alphabet of size $|S|$ requires $\log|S| = O(m)$ bits. Due to Lemma II.1, we can use an error correcting code such that each transmission is $O(m)$ and the label error probability is less than the required $(5(r+1))^{-32/\alpha}$. Specifically, for efficient decoding we require $p < |S|^{-2}$, which can be done with code of length $O(m)$ as well. The above lemma replaces Lemma 5.1.2 of [6], and leads to the following theorem.

Theorem A.10. *There exists a constructible and efficient simulation that computes any* m*-party protocol* π *of length* T *using a BSC for communication and a pre-shared* SBTC*. The simulation succeeds with probability* $1 - 2^{-\Omega(T)}$*, and imposes a dilation of* $O(m)$*.*

[13]The same tree can be used by all users.