University of California
Los Angeles

On Interactive Computation over a Noisy Channel

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Computer Science

by

Ran Gelles

2014

Abstract of the Dissertation

On Interactive Computation over a Noisy Channel

by

Ran Gelles

Doctor of Philosophy in Computer Science

University of California, Los Angeles, 2014

Professor Amit Sahai, Co-chair

Professor Rafail Ostrovsky, Co-chair

When two remote parties wish to perform some computation given that their communication channel may be noisy, sophisticated coding schemes must be employed in order to resist a constant fraction of noise while increasing the communication by only a constant factor. The ultimate goal is to find computationally efficient schemes that resist the maximal possible amount of noise while transmitting as few bits as possible.

In this work we examine two properties of such interactive schemes: their computational efficiency and their noise resilience. First, we show how to obtain an efficient scheme that is resilient to random errors (where each bit is flipped with constant probability). Next, we consider the case of adversarial noise (where we only limit the total amount of flipped bits) and obtain a scheme that is resilient to noise rates up to 1/2, given simple setup assumptions. All our schemes feature constant communication overhead.

Similar techniques can also be applied to the problem of communicating a data stream over a noisy channel, for which we achieve an optimal and efficient scheme that resists noise rates less than one.

The dissertation of Ran Gelles is approved.

Eliezer M. Gafni

Suhas N. Diggavi

Rafail Ostrovsky, Committee Co-chair

Amit Sahai, Committee Co-chair

University of California, Los Angeles

2014

Table of Contents

List of Figures

Acknowledgments

First and foremost, I am extremely grateful to my advisors Rafail Ostrovsky and Amit Sahai.

Rafi is a great advisor — he is a machine-gun of ideas, and approaches to tackle problems. Although he is probably one of the more busy persons I know, Rafi was always able to find the time to discuss my half-baked ideas, (usually) dismiss them, and push me to think harder and further. Throughout my short career so far, I rarely took any important decision without first asking Rafi's opinion, whose insights and strategy always surprised me. Rafi's healthy attitude towards research, publishing, and life in academia inspired me a lot, and greatly shaped the researcher I have become.

Amit is a great advisor — he has the remarkable ability of seeing things crystal clear, and being able to explain them so they become clear in the listener's mind (even if for just a single moment of clarity). I still remember this time I heard Amit describing our joint work to some other people (after the work was done and submitted)—only then, listening to Amit's explanations, I suddenly *really* understood my work... I thank Amit for plenty of conversations we had about research, teaching, and life, and for his great patience and supportive advice.

I am truly grateful for the complete financial support I received from Rafi and Amit. The value such an extensive support had on my research is inestimable. In times of constant budget cuts, this should not be taken for granted.

I thank Mark Braverman for hosting me for a summer internship at Princeton University. Mark is a fountain of knowledge and one of the quickest thinkers I've ever met. The short time I spent at Princeton was a true inspiration for me, and much of it is due to Mark.

I thank Juan Garay for being my mentor and hosting me for a summer internship at AT&T Research. Juan has been a great role model, and I learned a lot from him. In addition to his great understanding of science, Juan has a good grip on life, which is simply admirable.

I thank Serge Fehr for hosting me for a summer internship at CWI, and for showing me how to deal with tumbling times in the most professional way. I would also like to thank the

other members of the CWI group (Harry, Niek, Nacho, Ronald, Robbert, and Christian) for an interesting, fruitful, and pleasant summer.

I would like to thank Tal Mor for hosting me and supporting my travels to Israel. Visiting the Technion has become something I was looking forward to every time I came back to Israel. Best put by the Eagles: "You can check-out any time you like, but you can never leave".

I would like to thank my co-authors, who made many deadlines possible: Shweta Agrawal, Vladimir Braverman, Michel Boyer, Harry Buhrman, Nishanth Chandran, Serge Fehr, Matthew Franklin, Juan Garay, Vipul Goyal, David Johnson, Aggelos Kiayias, Ankur Moitra, Tal Mor, Alan Roytman, Christian Schaffner, Leonard Schulman, Akshay Wadia, Kina Winoto, and Moti Yung. I learned a lot from each and every one of you, and had a great pleasure working with you.

I would like to thank my friends and colleagues from the crypto lab, the theory lab, the computer science department, and other places at UCLA—Shweta, Prabhanjan, Vova, Darrell, Nishanth, Chongwon, David, Sanjam, Shankar, Vipul, Divya, Abhishek, Bhavana, Dakshita, Gergely, Abishek, Chen-Kuei, Maji, Anat, Alejandra, Vanishree, Alan, Uri, Alessandra, Michael, Brian, Ivan, Akshay, Tomer, and Vassilis—you made my time at UCLA worth it, and I thank you all for that.

A special one goes to my dear friends Chongwon and Julia Cho, and their cute son Brent. The Cho's place was almost my second home: together we spent many weekends of BBQ'ing, drinking beers and fancy scotch (on special occasions we had the finest doubly-processed coffee), or just playing together (board games if it was late enough, or cars/lego/'pick-a-fruit'/body-drumming if it wasn't). I will deeply miss having you around!

Living in L.A. wouldn't be the same without Victor and Michal Nahmias, and I have my deepest gratitude to them and to Ariel and Vicky, and Tamar and Andrew. Although we've never met before I moved to L.A., they accepted me as part of their family from the very first moment, and always made me feel welcome and loved.

Last, I would like to thank my family. Although far away, they are always close at heart.

Vita

2002	Philip M. Merlin Prize for distinguished student in Computer Engineering
2003	B.Sc. in Computer Engineering (summa cum laude), Technion – Israel Institute of Technology, Haifa, Israel
2003–2009	Military Service, Israel Defense Forces
2009	M.Sc. in Computer Science, Technion – Israel Institute of Technology, Haifa, Israel
2013	UCLA Dissertation Year Fellowship

Publications

R. Gelles, A. Moitra, and A. Sahai, "Efficient Coding for Interactive Communication", *IEEE Transactions on Information Theory*, **60**(3), pages 1899–1913, 2014.
Preliminary version in *FOCS 2011: Proceedings of the IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 768–777, October 2011.

H. Buhrman, N. Chandran, S. Fehr, R. Gelles, V. Goyal, R. Ostrovsky, and C. Schaffner, "Position-Based Quantum Cryptography: Impossibility and Constructions", *SIAM Journal on Computing (SICOMP)*, **43**(1), pages 150–178, 2014.
Preliminary version in *CRYPTO 2011: Proceedings of the 31st Annual Cryptology Conference*, LNCS 6841, pages 429–446, August 2011.

R. Gelles, R. Ostrovsky, and A. Roytman, "Efficient Error-Correcting Codes for Sliding Windows", In *SOFSEM 2014: Proceedings of the 40th International Conference on Current Trends in Theory and Practice of Computer Science*, LNCS 8327, pages 258–268, January 2014.

R. Gelles, A. Sahai, and A. Wadia, "Private Interactive Communication across an Adversarial Channel", In *ITCS 2014: Proceedings of the 5th conference on Innovations in theoretical computer science*, pages 135–144, January 2014.

M. Franklin, R. Gelles, R. Ostrovsky, and L. J. Schulman, "Optimal Coding for Streaming Authentication and Interactive Communication", In *CRYPTO 2013: Proceedings of the 33rd Annual Cryptology Conference*, Part II, LNCS 8043, pages 258–276, August 2013.

V. Braverman, R. Gelles, and R. Ostrovsky, "How to Catch L_2-Heavy-Hitters on Sliding Windows", In *COCOON 2013: Proceedings of the 19th International Conference on Computing and Combinatorics* LNCS 7936, pages 638–650, June 2013.

S. Fehr, R. Gelles, and C. Schaffner, "Security and Composability of Randomness Expansion from Bell Inequalities", *Physical Review A*, **(87)**:012335, 2013.

R. Gelles and T. Mor, "On the Security of Interferometric Quantum Key Distribution", In *TPNC 2012: Proceedings of the 1st International Conference on Theory and Practice of Natural Computing*, LNCS 7505, pages 133–146, October 2012.

M. Boyer, R. Gelles, and T. Mor, "Attacks on Fixed Apparatus Quantum Key Distribution Schemes", In *TPNC 2012: Proceedings of the 1st International Conference on Theory and Practice of Natural Computing*, LNCS 7505, pages 97–107, October 2012.

R. Gelles, R. Ostrovsky, and K. Winoto, "Multiparty Proximity Testing with Dishonest Majority from Equality Testing", In *ICALP 2012: Proceedings of the 39th International Colloquium on Automata, Languages and Programming*, Part II, LNCS 7392, pages 537–548, July 2012.

M. Boyer, R. Gelles, and T. Mor, "Security of the Bennett-Brassard quantum key distribution protocol against collective attacks", *Algorithms*, **2**(2), pages 790-807, 2009.

M. Boyer, R. Gelles, D. Kenigsberg, and T. Mor, "Semiquantum Key Distribution", *Physical Review A*, **(79)**:032341, 2009.

CHAPTER 1

Introduction

Suppose Alice and Bob wish to communicate some information, however, their communication channel may be noisy. Shannon [Sha48], in a seminal paper, laid the mathematical foundations for analyzing communication of information over noisy channels, and initiated the study of coding schemes and error-correcting codes. The setting considered by Shannon (and generally, by most error-correcting codes) can be summarized in the following way: Alice holds a message x that she wishes to send to Bob. The message x is selected from a message space M according to some distribution $p(x)$. To overcome the channel noise, Alice first encodes her message using an *error-correction code*, and obtains a *codeword* $c = \mathsf{ECC}(x)$. The codeword is then communicated over the noisy channel, so that Bob receives $\tilde{c}$, a noisy version of the codeword. Bob decodes his received codeword and obtains $\tilde{x} = \mathsf{ECC}^{-1}(\tilde{c})$. The goal is to find an error-correction code with "good properties": e.g., *efficient* encoding and decoding procedures; small codeword size (as a function of the size of the message space); high *success probability*, $\Pr[\tilde{x} = x]$, etc.

In this work we consider two extensions of the above model. First, we consider coding in the *interactive communication* model. Here, rather than having one party who sends the information and other party who receives it, the communication is bi-directional. Specifically, we consider the case where Alice holds some input x, Bob holds an input y, and their desire is to jointly compute a function $f(x, y)$. To that end, they run a *protocol* which defines the messages they send to each other at each round. As before, the communication channel (in both direction) may be noisy.

Noise can have two flavors: it can be *random*, where each bit is flipped with some constant probability; or it can be *adversarial* where the only limit is on the total amount of bits that

can be flipped. In both noise models, the goal is to find noise-resilient protocols with "good properties": e.g., low communication overhead (with respect to the noiseless protocol); high success probability, etc.

The second way we extend the standard "single message transfer" model is by considering the *data stream* model. Here only Alice has information to communicate to Bob, however instead of a single (finite) message, Alice holds a *data stream*—an infinite sequence of symbols $\{x_1, x_2, \ldots\}$ such that Alice learns (or generates) x_i only at time i. The goal is to devise a coding scheme that allows Bob to *eventually* learn each element x_i of the data stream. As above, an important desire is to keep the communication overhead small.

We now expand about each of the two directions, and state our main results.

1.1 Interactive Coding Schemes

At a first look, it seems that the question of interactive communication can easily be solved using (standard) error-correction codes: the parties can simply compute the function $f(x, y)$ by running any protocol π that computes f, where each message sent by π is encoded by an error correction code prior to its transmission. However, this intuition is wrong. When each message is encoded separately, there is some (tiny) probability that the decoding will fail. However, the protocol for computing $f(x, y)$ may be very long, that is, communicate many messages. In this case, with a high probability at least one message will be decoded incorrectly, and the output of the protocol will be incorrect. To avoid this issue, one must use a very strong error-correcting code, in which the success-probability of the code depends on the length of the protocol π. Yet, roughly speaking, such codes will have a large codeword size, and the communication overhead of the noise-resilient protocol will be high.

The question of interactive communication was first considered by Schulman, in a breakthrough sequence of papers published in 1992 and 1993 [Sch92, Sch93, Sch96]. Schulman gave 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 (memoryless) 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.

Unfortunately, Schulman's 1992 emulation procedure [Sch92] either assumed the parties already share a large amount of randomness before they communicate, where the length of the shared random string is quadratic in the length of the protocol to be emulated or required inefficient encoding and decoding. On the other hand, Schulman's 1993 emulation procedure [Sch93, Sch96] relies on the existence of *tree codes*, a complex data-structure of which no efficient construction is known.

Efficient schemes for random noise. We revisit the problem of reliable interactive communication and give the first fully efficient emulation procedure for reliable interactive communication over noisy channels with a constant rate[1]. 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. To obtain this result, we do the following:

- We introduce a new relaxed notion for a tree code, namely, a *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 decoded with respect to any discrete memoryless channel with constant capacity with overwhelming probability.
- We prove the correctness of an efficient emulation procedure based on any potent tree code. (This replaces the need for (standard) tree codes in the work of Schulman [Sch96].)

[1]Constant rate emulation means that the communication of the noise-resilient protocol is at most constant times more than the communication of the noiseless protocol. See Section 2.3 for a definition.

- Finally, we are able to partially de-randomize the above result by means of epsilon-biased distributions. Specifically, using element-wise efficient[2] constructions of small bias sample spaces [AGHP92], we give a 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.

This immediately yields our first result,

Theorem 1.1. *For any two-party protocol π with communication T bits assuming a noiseless channel, there exists an efficient protocol Π with communication $O(T)$ bits, that emulates π assuming each message is sent over a memoryless noisy-channel. The emulation succeeds with probability $1 - 2^{-\Omega(T)}$.*

We also extend our result to the case of any number m of parties. This is done in a similar fashion to the above. We build on a an m-party emulation scheme by Rajagopalan and Schulman [RS94] which uses tree-codes. We replace the (non element-wise efficient) tree codes with a potent tree code, to obtain an efficient scheme,

Theorem 1.2. *For any m-party protocol π of T rounds in which the maximum connectivity is r, there exists an efficient protocol Π with $O(T)$ rounds and a slowdown of $O(\log(r+1))$ over π's communication (that is, the communication is higher by a factor of $O(\log(r+1))$, and Π emulates π over a memoryless noisy channel with probability at least $1 - 2^{-\Omega(T/m)}$*

Resilient schemes for adversarial noise, assuming shared randomness. Next, we consider interactive communication when the noise may be adversarial rather than random. Schulman [Sch96] showed a constant-rate encoding scheme that copes with a noise rate of up to 1/240 of the transmitted bits. Later, Braverman and Rao [BR11] showed how to deal with noise rates less than 1/4. In addition, Braverman and Rao showed that 1/4 is the highest error rate any interactive protocol can withstand, as long as the protocol defines whose turn

[2]By an "element-wise efficient" object, we mean that the i-th bit of the object should be computable in time polylogarithmic in the size of the object.

it is to speak at every round regardless of the observed noise. The fascinating open question left by the work of Braverman and Rao is whether other methods could circumvent the 1/4 bound.

We improve the bound obtained by [BR11] by allowing the parties to pre-share a random string unknown to the adversarial channel. Specifically, we show how to convert any interactive protocol (for noiseless channel) into a constant-rate protocol that withstands any adversarial noise level smaller than 1/2, given pre-shared randomness. We also show that for higher noise rates, no constant-rate interactive protocol exists for tasks that depend on inputs of both parties.

Theorem 1.3. *For any constant $\varepsilon > 0$ and for any function f and inputs x, y, there exists an interactive protocol with constant rate in the shared-randomness model, such that if the adversarial corruption rate is at most $\frac{1}{2} - \varepsilon$, both parties output $f(x, y)$ with overwhelming probability over the shared random string.*

We further observe that if communication is performed over an *erasure channel*, we can resist erasure rate of up to $1/2 - \varepsilon$ with constant dilation and *without the need for a shared randomness*. This is obtained by employing our analysis directly on the scheme of Braverman and Rao [BR11]. Our analysis thus implies that the scheme of [BR11] is optimal for the erasure channel model, since with erasure rate 1/2, the adversary can completely delete the outgoing communication of a single party.

Theorem 1.4. *For any constant $\varepsilon > 0$ and for any function f, there exists an interactive protocol over an erasure channel, that has a constant rate and resists adversarial erasure rate of up to $\frac{1}{2} - \varepsilon$.*

1.2 Coding Schemes for Data Streams

We then turn to investigate the question of transmitting data streams over an adversarially noisy channel. Within this framework we consider two related questions, namely, *error-correction* and *authentication* of data streams. Loosely speaking, in error-correction schemes,

the receiver decodes the correct message as long as the noise level is below some threshold (but possibly outputs a wrong message if the noise exceeds that threshold). In authentication schemes, the receiver's task is to indicate whether or not the received (decoded) message is indeed the one sent to him. To see the relation between these two tasks note that if the corruption level of an adversary is guaranteed to be lower than the threshold, any error-correction scheme guarantees that the receiver decodes the original message. However, while no constant-rate error-correction scheme can withstand a noise level higher than 1/2, this is not the case for authentication schemes that are capable of indicating a change in the message even when the adversary has a full control of the channel. On the other hand for the task of authentication, it is generally assumed that the parties pre-share a secret key.

Standard error-correction and authentication methods do not apply directly to the model of data streams. The straightforward method to perform error-correction (or authentication) of a data stream is to cut the stream into chunks and separately encode each chunk. The problem now is that while the adversary is limited to some *global* noise rate, there is no restriction on the noise level of any local part of the stream. Specifically, the adversary can corrupt a single chunk in its entirety (while not exceeding the global amount of allowed noise), and cause Bob to decode this chunk in a wrong way. Even if this event is noticed by Bob since the chunk fails the authentication, the information carried within this chunk is lost unless Bob requests a retransmission of that chunk, i.e., unless the communication is interactive. The same problem exists (with high probability) when the noise is random rather than adversarial, given that the stream is long enough or infinite.

A possible mitigation to the above is to increase the chunks' size. This, however, has an undesirable side effect—Bob needs to wait until receiving a complete chunk in order to decode and authenticate it. This means that the information received in the very recent bits is inaccessible until the chunk is completely received. Our goal is thus, to construct a constant-rate scheme that can withstand a constant fraction of errors (globally) and still guarantee the correct decoding and authenticity of the information received so far.

We begin by constructing a constant-rate coding scheme for data streams that withstands noise rates of less than 1/2. Informally, as long as the global noise rate up to some time n

does not exceed some parameter $c < 1/2$, a fraction of $1 - 2c$ of the stream sent up to time n can be recovered. For constant-rate schemes, it is clear that $c < 1/2$ is a hard limit and no scheme can succeed when the noise is higher. In order to achieve schemes that withstand higher noise rates we must relax the model and give the users more resources. Indeed, with the use of shared randomness (i.e., a shared secret key) we can break the $c = 1/2$ barrier. To emphasize the fact that the parties are allowed to share a secret key, we refer schemes in this model as *authentication schemes* rather than error-correction schemes, based on the relation of these two tasks mentioned above.

Our main result for this part is constructing a constant-rate authentication scheme for data streams over a noisy (adversarial) channel. For any constant fraction of noise c less than 1, our scheme succeeds in decoding at least a $(1 - c)$-fraction of the stream so far, with high probability. The decoded part is always the *prefix* of the stream. The decoded prefix is authenticated, meaning that there is only a negligible probability that the scheme outputs a wrong string. Furthermore, our scheme is *efficient*.

Theorem 1.5. *For any noise rate $0 \leq c < 1$ and small constant $\varepsilon > 0$, there exists an efficient constant-rate coding scheme for data streams in the shared-randomness model that, at any time n, the receiver decodes with high probability a prefix of length at least $(1-c)n-\varepsilon n$ of the stream sent so far.*

Our scheme does not make any (cryptographic) assumptions, other than pre-sharing a secret random string. The amount of randomness utilized by the scheme grows with the message length, and can be unbounded if the data stream is infinite.

In addition, we show that our scheme is optimal with regard to the length of the decoded prefix:

Theorem 1.6. *For any noise rate $0 \leq c < 1$ and $\varepsilon > 0$, Assume that the receiver at time n decodes a prefix of length $(1 - c)n + \varepsilon n$, then the decoding succeeds with probability at most $2^{-\Omega(\varepsilon n)}$ in the worst case.*

1.3 Related Work

Interactive Communication. As mentioned above, the question of performing interactive computation over a disturbed channel was initiated by Schulman [Sch92, Sch93, Sch96] who gave a scheme that can deal both with random and adversarial noise. For the random noise case, Schulman's scheme is efficient up to the construction of the tree code.

For adversarial noise, the Schulman's scheme was shown to resist a noise fraction of up to 1/240, yet this cannot be done efficiently. The work of Braverman and Rao [BR11] improves the noise resiliency to 1/4, which is the maximal possible in this setting. Braverman and Efremenko [BE14] further analyze the noise resilience of interactive schemes, and determine, for any noise rate (α, β) where α is the amount of noise in the channel from Alice to Bob and β is the noise in the other direction, whether or not interactive communication (with constant rate) is possible. Furthermore, they show a region of noise in which one can only achieve a *list-decoding* notion of interactive communication; that is, the parties will conclude the computation with a short list of possible outputs, rather than a single (unique) output. The notion of list decoding in interactive communication was also considered by Ghaffari, Haeupler, and Sudan [GHS13, GH13] who also show that a list-decoding notion for interactive-communication could resist noise rate up to 1/2.

As for efficiency in the case of adversarial noise, Brakerski and Kalai [BK12] give the first efficient scheme for interactive communication, which resists noise rate of up to 1/16. Brakerski and Naor [BN13] further improve the efficiency and obtain a quasi-linear scheme. With the help of list-decoding Ghaffari and Haeupler [GH13] achieve an efficient, constant rate[3] scheme that resists the maximal noise rate of 1/4.

The question of interactive communication was also examined in other models and settings. Brassard, Tapp, and Touchette [BTT13] examine interactive communication in the quantum setting and achieve a scheme that resists noise rate of up to 1/2. Another model that was considered is that of *adaptive protocols*. In those protocols, the parties may determine who is

[3]In fact, [GH13] gets only quasi-constant rate, but this could be turned into constant rate [Hae14], combining with the result of [BE14].

the next person to send a message in an online and independent manner, possibly according to the noise observed. Ghaffari, Haeupler and Sudan [GHS13] consider such a model and show that adaptivity allows resisting more noise. Specifically, they give a scheme with constant rate that resists 2/7, which is the maximal noise in their model. If the parties are to preshare a random string, the bound of 1/2 we present in this work can be improved to 2/3. Adaptive protocols were also considered by Agrawal, Gelles, and Sahai [AGS13], yet in a different model. E.g., while in [GHS13] the protocol have a constant length, in [AGS13] the length of the protocol is also adaptively determined according to noise. They consider several variants of adaptive models, and show that such adaptive protocols can resist noise rate up to 2/3 in a model where parties are allowed to "remain silent", and the noise is relative to the communication of the specific instance. They also analyze a model in which the adaptivity comes from changing the length of the protocol according to the observed noise. A different variant in [AGS13] considers interactive communication over *erasure channels* for which a protocol with constant rate that resists noise rate up to $1 - \varepsilon$ is given.

Another line of research considers performing interactive communication with the additional requirement of *privacy*, where the protocol doesn't leak more than the necessary information. Chung, Pass, and Telang [CPT13] considered protocols that leak *exactly* the same information as the noiseless protocol. They prove that there is no compiler that takes a protocol and generates a noise-resilient protocol with good parameters (i.e., constant rate) such that the compiled version exactly imitates the information revelation of the original protocol. Their work extends also to the computational setting where they show an interesting tradeoff between the noise susceptibility and the rate of any possible scheme. Gelles, Sahai and Wadia [GSW14] also consider the privacy of interactive communication schemes, and prove a strong impossibility result for performing information-theoretically *private* interactive communication, even when the fraction of noise is exponentially small.

Kol and Raz [KR13] analyze the capacity of binary symmetric channels in the interactive setting. They show that when the noise probability ε approaches zero, the capacity of the channel behaves like $1 - \Theta\left(\sqrt{H(\varepsilon)}\right)$. This implies a difference between the interactive and non-interactive case, as the latter has capacity $1 - H(\varepsilon)$, where H is the binary entropy function.

Tree Codes. See Section 3.1.

Coding in the shared-randomness model Coding schemes that assume the parties pre-share some randomness (also known as *Private Codes* [Lan04]) first appeared in [Sha58], and were greatly analyzed since. The main advantage of such codes is that they can deal with *adversarial noise*, rather than a random noise. Langberg [Lan04] considers private codes for adversarial channels that approach Shannon's bound and require only $O(\log n)$ randomness for block size n, as well as an $\Omega(\log n)$ lower bound for the needed randomness. The construction of Langberg also implies an efficient code with $O(n \log n)$ randomness. This result was improved to $n + o(n)$ randomness by Smith [Smi07]. Explicit constructions with $o(n)$ randomness are yet unknown (see [Smi07]).

Coding for Data Streams. There has been quite a lot of work on coding scheme for data stream, however most previous works assume an erasure channel [MS02, THYJ12, LH12, BKTA13] and do not apply to the more general case of arbitrary noise.

The works of Even, Goldreich and Micali [EGM90] and Gennaro and Rohatgi [GR97] consider authentication of data streams, however the focus of these schemes is not only to authenticate the message but also to prevent the sender from denying having signed the information. These constructions rely on cryptographic primitives such as one-time signatures. Another related line of research [PCTS00, MS01, GM01] pursues authentication of streams over *lossy* channels, usually in the multicast setting.

Gelles, Ostrovsky and Roytman [GOR14] extend the question of communicating a data stream over a noisy channel to the *sliding window* model, in which the receiver needs to be able to decode only the last "window" (last N symbols), rather than the entire stream.

CHAPTER 2

Preliminaries

2.1 Notations

We begin with several notations we use throughout. We denote the set $\{1, 2, \ldots, n\}$ by $[n]$, and for a finite set Σ we denote by $\Sigma^{\leq n}$ the set $\cup_{k=1}^{n} \Sigma^k$. The Hamming distance $\Delta(x, y)$ of two strings $x, y \in \Sigma^n$ is the number of indices i for which $x_i \neq y_i$. Throughout this manuscript $\log()$ denotes the binary logarithm (base 2) and $\ln()$ denotes the natural logarithm (base e).

2.2 Interactive Communication Protocols

We begin by defining the notion of an interactive-communication protocol that computs some function f, assuming an ideal (noiseless) communication channel. Such a protocol is usually referred to as the *noiseless protocol*.

Let $\mathcal{X}, \mathcal{Y}$, and $\mathcal{Z}$ be some finite sets. Consider a distributed computation of a fixed function $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$, performed by two users, Alice and Bob (A and B for short), where Alice holds the input $x \in \mathcal{X}$ and Bob holds the input $y \in \mathcal{Y}$.

Definition 2.1. *A two-party protocol π is two probabilistic algorithms, π_A, π_B run by party A, B, respectively. The protocol runs for T rounds (also called the length of the protocol). At each round $i = 1, \ldots, T$, both A and B send a single message to each other, denoted $a_i, b_i \in \Sigma$, where Σ is a finite set. Each such message is a function of the party's input and all the messages the party received so far. The protocol π is said to (perfectly) compute $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ if for any pair of inputs $x \in \mathcal{X}$, $y \in \mathcal{Y}$, both parties output $f(x, y)$.*

For simplicity, we will assume the protocol π is deterministic.

Definition 2.2. *The transcript of π on x, y, $\mathsf{trans}_\pi(x, y)$, is all the messages exchanged during a run of π on inputs x, y.*

Definition 2.3. *The communication complexity of a protocol π, $\mathsf{CC}(\pi)$, is the (maximal) amount of bits communicated during an instance of π.*

Note that the protocol communicates the same amount of bits/symbols for any pair of inputs. We can assume that Σ is a power of two, thus

$$\mathsf{CC}(\pi) = 2T \cdot \log |\Sigma|.$$

Definition 2.4. *The communication complexity of a function f, $\mathsf{CC}(f)$, is the minimal communication complexity of a protocol π that computes f,*

$$\mathsf{CC}(f) = \min_{\pi \text{ computes } f} \mathsf{CC}(\pi)$$

An alternative definition for interactive protocols assumes that the parties exchange messages in an alternating manner, that is, Alice sends a message in odd rounds, and Bob in even rounds. For such protocols, $\mathsf{CC}(\pi) = T \log |\Sigma|$. It is clear that the two models are equivalent, maybe up to a factor two in the communication complexity f.

2.3 Channels and Noise Models

In a more realistic model, the channel between A and B may be noisy. We model the i-th instantiation of the the channel as the distribution

$$\mathsf{CH}(out_i \mid in_i, (in, out)_{i-1}, \ldots, (in, out)_1, x, y)$$

where $in_j \in \Sigma$ is the input to the channel at the j-th channel's instantiation, $out_j \in \Sigma$ is the corresponding output symbol, and x, y are the inputs of the parties. As mentioned above, we consider two types of noisy channels: random and adversarial.

2.3.1 Random Noise Channels

Generally, we say that the noise model is random if the channel is memoryless, that is, the noise is independent of previous transmissions.

Definition 2.5. *A discrete memory-less channel over alphabet Σ, is a channel that given input $x \in \Sigma$, outputs $y \in \Sigma$ according to a given transition probability $p(y \mid x)$. The channel is memory-less if the probability distribution $p()$ is independent of previous input symbols.*

A representative example of a channel with random noise is the *binary symmetric channel*:

Definition 2.6. *A binary symmetric channel (BSC) with error probability p_{BSC} is a binary channel $\mathsf{CH}_{BSC} : \{0,1\} \to \{0,1\}$ such that each input bit is independently flipped with probability p_{BSC}. (I.e., for $b \in \{0,1\}$, $p(b \mid 1-b) = p_{BSC}$ and $p(b \mid b) = 1 - p_{BSC}$.)*

The BSC 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.

Definition 2.7. *The* capacity *of a discrete memoryless channel is*

$$C = \max_X I(X;Y)$$

where X is the random variable describing the distribution of the input x, Y is the distribution of the output y, and the maximum is taken over all possible input distributions $p(x)$. $I(X;Y) = \sum_{x \in X, y \in Y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)}$ is the mutual information of X and Y.

The capacity of the BSC channel is given by $C_{BSC} = 1 - H(p_{BSC})$, obtained when the input is uniform over $\{0,1\}$. $H(x) = x \log \frac{1}{x} + (1-x) \log \frac{1}{1-x}$ is the binary entropy function.

A main ingredient for communication over noisy channels is Shannon's *Coding Theorem* [Sha48] that shows that an exponentially small decoding error in the length of the message $|m|$ can be achieved if the message is encoded into a codeword of length $c|m|$, for some constant $c > 1$ determined by the channel's capacity.

Lemma 2.1 (Shannon Coding Theorem [Sha48])**.** *For any discrete memoryless channel* CH *with capacity* C*, any message space* S *and any* $\xi > 1$*, 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 for any* $m \in S$*,*

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

That is, for random noise it is possible to encode a single message by increasing its length by a constant, and ensuring that the decoder will obtain the correct message with overwhelming probability.

2.3.2 Adversarial Noise Channels

While the above random noise model captures a very realistic (and simple) type of disturbed channels, the adversarial noise model captures, in some sense, the worst case scenario. In this model, there are no limitations about the channel, besides the total number of flipped bits (or the number of corrupted symbols).

Definition 2.8. *The noise rate* η *induced by a channel* $\mathsf{CH} : \Sigma \to \Sigma$*, is the fraction of channel instantiations for which the output symbol differs from the input symbol.*

We usually think of such a channel as an all-knowledgeable adversary Eve that corrupts exactly those transmissions that will cause the most damage. Eve is assumed to know the protocol and the inputs of the parties, and is bounded only by the amount of symbols she is allowed to corrupt.

2.3.3 Erasure Channels

The erasure channel $\mathsf{CH} : \Sigma \to \Sigma \cup \{\perp\}$ captures a model in which transmissions may be dropped by the network, in that case the receiver gets an erasure mark $\perp$. However, when the transmission is not erased it is guaranteed that $\sigma = \mathsf{CH}(\sigma)$. In other words, the receiver either gets the correct symbol, or an erasure mark $\perp$. Erasure channels can have random erasures (e.g., each transmission is erased with constant probability) or adversarial erasures for which only the total amount of erasures is bounded.

2.4 Emulation Schemes

The protocol π of Definition 2.1 is guaranteed to work if the channel is noiseless, however its output over a noisy channel is not necessarily correct. The goal of an *emulation* is to "run" π over a noisy channel. That is,we wish to obtain $\mathsf{trans}_\pi(x, y)$ and recreate the transcript of π on inputs x, y.

Definition 2.9. *Let π be a two party protocol that assumes a noiseless channel. An* emulation *of π over a noisy channel CH with success probability $1 - \delta$, is a two-party protocol Π such that if each message is sent over CH, then for any input x, y,*

$$\Pr\left[\Pi(x, y) \neq \mathsf{trans}_\pi(x, y)\right] < \delta.$$

We can define the *rate* of an emulation as the fraction of the noiseless communication from the noise-resilience emulation. Formally,

Definition 2.10. *The* rate *of an emulation Π of a noiseless protocol π is,*

$$\textit{rate} = \frac{\mathsf{CC}(\pi)}{\mathsf{CC}(\Pi)}.$$

Equivalently, the slowdown *or the* dilation *of an emulation is $1/\textit{rate}$.*

The desired property of the emulation is to succeed with high probability, while keeping their communication complexity linear in the communication of the noiseless protocol, that is, to have a *constant rate*.

Definition 2.11. A good emulation *of π over a channel CH, is a protocol Π with rate $= O(1)$, such that if each message of Π is sent over CH, the success probability is $1 - 2^{-\Omega(T)}$.*

Assume π runs T rounds and Π runs N rounds. In cases where each message of the protocol comes from a finite alphabet Σ (which is always the case for our protocols), then the communication complexity and the round complexity differ only by a constant. Thus, $T/N = O(1)$ implies a constant rate. A good emulation is thus a protocol Π with with a constant slowdown $N = O(T)$, and exponentially small failure probability, $\delta = 2^{-\Omega(N)}$.

2.5 The Shared Randomness Model

Some of our schemes assume the following *shared-randomness model*. The legitimate users (Alice and Bob) have access to a random string R of unbounded length, which is unknown to the adversary (Eve). Protocols in this model are thus *probabilistic*, and are required to succeed with high probability over the choice of R. We assume that all the randomness comes from R and that for a fixed R the protocols are deterministic.

CHAPTER 3

Potent Tree Codes

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.

In this chapter we introduce our notion of potent tree codes, and show a construction of potent tree codes of depth N such that the description of the entire tree takes $O(N)$ bits, and each label in the tree can be constructed in time $poly(N)$.

We begin in Section 3.1 with the definition of a (standard) tree code as originally defined by Schulman [Sch96]. In Section 3.2 we discuss the properties a tree code needs to have in order to be useful for interactive communication, and define our relaxed notion of potent tree codes. In Section 3.3 we consider trees that are uniformly sampled out of the entire space of possible trees. We show that such trees are potent with high probability. This construction, although efficient and very simple, yields trees whose description is very long, namely $O(\exp(N))$. This motivates our second construction, which is based on sampling from a small-biased space. We prove that such a tree is potent with high probability and show that its description is succinct (Section 3.4). In Section 3.5 we construct a stronger potent tree, which will be very useful for interactive communication in the multi-party setting.

The results of this chapter are based on [GMS11, GMS14].

3.1 (Schulman's) Tree Codes

A d-ary *tree code* [Sch96] over alphabet Σ is a rooted d-regular tree of arbitrary depth N whose edges are labeled with elements of Σ. For any string $x \in [d]^{\leq N}$, a d-ary tree code $\mathcal{T}$ implies *an encoding* of x, $\mathsf{TCenc}_{\mathcal{T}}(x) = w_1 w_2 .. w_{|x|}$ with $w_i \in \Sigma$, defined by concatenating the labels along the path defined by x, i.e., the path that begins at the root and whose i-th node is the x_i-th child of the $(i-1)$-th node. We sometimes identify the string x with the node in the tree defined by the above path, and use these two notions interchangeably. We usually omit the subscript $\mathcal{T}$ when the tree is clear from the context. Note that tree code encoding is *online*: for $\sigma \in [d]$, in order to communicate $\mathsf{TCenc}(x\sigma)$ given that $\mathsf{TCenc}(x)$ was already communicated, we only need to send one symbol of Σ. Hence, if $|\Sigma| = O(1)$ the encoding scheme has a constant rate.

For any two paths (strings) $x, y \in [d]^{\leq N}$ of the same length n, let ℓ be the longest common prefix of both x and y. Denote by $anc(x, y) = n - |\ell|$ the distance from the n-th level to the least common ancestor of paths x and y.

Definition 3.1. *A tree code is said to have* a distance α *if for any $k \in [N]$ and any distinct $x, y \in [d]^k$, the Hamming distance of their encodings satisfies*

$$\Delta(\mathsf{TCenc}(x), \mathsf{TCenc}(y)) \geq \alpha \cdot anc(x, y).$$

For a string $w \in \Sigma^n$, decoding w using the tree code $\mathcal{T}$ means returning the string $x \in [d]^n$ whose encoding minimizes the Hamming distance to the received word, namely,

$$\mathsf{TCdec}_{\mathcal{T}}(w) = \operatorname*{argmin}_{x \in [d]^n} \Delta(\mathsf{TCenc}_{\mathcal{T}}(x), w).$$

Applications that transmit N symbols via a tree code require a tree of depth N (and size $\exp(N)$). In order for such applications to be efficient, it is required that a encoding of strings of length N could be done in time $poly(N)$. Indeed, constructing the entire tree still takes exponential time, yet the application uses only polynomial fraction of the tree and can be efficient. We say that such constructions, in which the time for retrieving a single element is polynomial in its depth, are *element-wise efficient*.

Tree Code Constructions. Although tree codes (of unbounded depth N and constant-size alphabet) are known to exist [Sch96], finding a deterministic element-wise efficient construction remains an open question.

Peczarski [Pec06] provides a randomized way for constructing absolute tree codes. The construction succeeds with probability $1 - \epsilon$ using alphabet of size $O(\exp(\sqrt{\log \epsilon^{-1}}))$. Therefore, if we use Peczarski's method to construct a tree code with exponentially small failure probability ϵ, encoding a string yields super-logarithmic slowdown if ϵ is negligible (in the length of the string to be encoded).

Braverman [Bra12] defines a "product" operator, which takes two trees and yields a deeper tree with essentially the same distance parameter α. By repeatedly performing products of short trees, one can obtain a tree code of depth N in sub-exponential time in N.

Other methods for constructing a tree code are reported by Schulman [Sch03], yet they require polynomial-size alphabet (in the depth of the tree), resulting in a logarithmic slowdown using Schulman's emulation [Sch96]. Schulman [Sch03] 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).

Moore and Schulman [MS14] provide a first candidate for efficient deterministic construction of tree codes, yet their construction is based on a strong assumption regarding sums of exponents.

3.2 Potent Tree Codes

3.2.1 Relaxing the Tree Code Notion

We ask 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.) 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). We formally describe the emulation process in Chapter 4, but for the meantime assume that each party decides on the next message to send by some function of its input and the messages received so far. This message is encoded by the tree-code as explained above, and sent over the noise channel to the other side. On the receiving side, each party takes all the (tree code encoded) symbols received so far and decodes via the tree code all the messages sent by the other party so far. Then, the receiving party can send a new message according to this transcript and its input. Very informally, the emulation advances correctly as long as both parties decode the correct information. However, when errors occur and the parties decode wrong messages the emulation may temporarily advance incorrectly. Nevertheless, if the number of errors is small, the tree code will eventually decode the correct path and put the emulation back on track. That is, in order to succeed, the parties need to be "on the right track" for enough rounds to complete the emulation correctly.

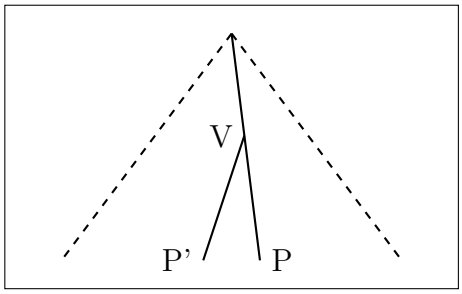

Figure 3.1: *A very bad tree code with two long paths that violate the Hamming distance condition.*

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 3.1 has the property that the relative Hamming distance between the path from node V to P and the path from node V 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.

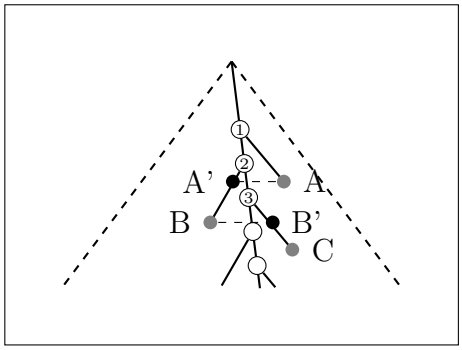

Figure 3.2: *A bad tree code with multiple short paths that violate the Hamming distance.*

Consider the tree code illustrated in Figure 3.2. 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 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 emulating the original protocol.

Schulman [Sch93] dealt with this problem by simply insisting that *all* divergent paths have large relative Hamming distance in his definition of a tree code (Definition 3.1). 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 satisfy the tree code property.

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 3.3), 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.

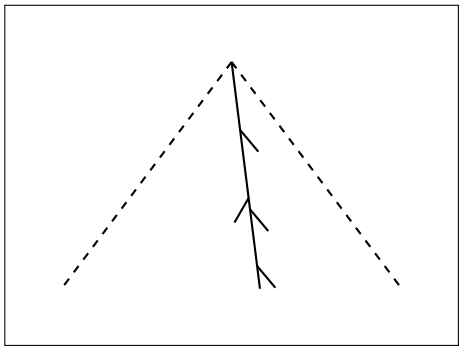

Figure 3.3: *A potent tree code with small number of bad paths.*

3.2.2 Potent Tree Codes: Definition

We now formally define the set of *potent tree codes* 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 $\varepsilon > 0$.

Roughly speaking, 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 of 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 one for which in 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. Formally,

Definition 3.2. *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 $\Delta(\mathsf{TCenc}(u), \mathsf{TCenc}(v)) < \alpha\ell$. 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 3.3. *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 an (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.

3.3 Random Tree Codes as Potent Trees

Suppose any label in a tree code is chosen, randomly and independently, out of an alphabet S and call such a tree a *Random Tree Code* (RTC). An RTC is a potent tree except with probability exponentially small in the depth of the tree.

Theorem 3.1. *Suppose $\varepsilon, \alpha \in (0, 1)$. Except with probability $2^{-\Omega(N)}$, a RTC with alphabet $|S| > (2d)^{(1+2/\varepsilon)/(1-\alpha)}$ is (ε, α)-potent.*

Proof. We begin by some technical lemmas.

Lemma 3.2. *Fix a d-ary RTC over S, and let v_1 and v_2 be any two nodes at some common depth h, with least common ancestor at depth $h - l$, then for every $0 \leq \alpha \leq 1$,*

$$\Pr\left[\Delta(\mathsf{TCenc}(v_1), \mathsf{TCenc}(v_2)) \leq \alpha l\right] \leq \left(\frac{2}{|S|^{1-\alpha}}\right)^{l}.$$

Proof. We sum the probability for any possible Hamming distance $i = 0, 1, \ldots, \alpha l$. A direct calculation gives

$$\Pr[\Delta(\mathsf{TCenc}(v_1), \mathsf{TCenc}(v_2)) \leq \alpha l] \leq \sum_{i=0}^{\alpha l} \binom{l}{l-i} \left(\frac{1}{|S|}\right)^{l-i} \left(\frac{|S|-1}{|S|}\right)^{i} \leq \frac{2^l}{|S|^{l(1-\alpha)}} \ .$$

□

Next we show that the set of all (ϵ, α)-bad RTC for constants $\epsilon, \alpha \in (0, 1)$, is exponentially small.

Proposition 3.3. *Suppose $\varepsilon, \alpha \in (0, 1)$. The probability for a RTC of depth N with alphabet $|S| > (2d)^{(1+2/\varepsilon)/(1-\alpha)}$ to be (ε, α)-bad, is at most $2^{-\Omega(N)}$*

Proof. We begin by fixing a leaf z, and later we use a union bound to bound the probability over the entire tree. Assume that there exist bad intervals of total length at least εN, then there must exist *disjoint* bad intervals of total length at least $\varepsilon N/2$, as stated by the following lemma [Sch96].

Lemma 3.4 ([Sch96])**.** *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 $\sum_{i \in I} |\ell_i| \geq X/2$.*

The proof is given in [Sch96].

There are at most $\sum_{j=\epsilon N/2}^{N} \binom{N}{j} \leq 2^N$ ways to distribute these disjoint intervals along the path from the RTC's root to z. Using Lemma 3.2 and a union bound we are assured that the probability of having a bad interval of length ℓ is less than $(2d/|S|^{1-\alpha})^{\ell}$. The probability for a specific pattern of disjoint bad intervals to jointly occur is the multiplication of the probability for each interval to occur (the intervals are independent since they are disjoint). According to the above, for large enough S, the probability for a RTC to be (ε, α)-bad is

bounded by

$$\Pr[\ \mathsf{RTC} \text{ is } (\varepsilon,\alpha)\text{-bad }] \le \sum_{z} \sum_{\substack{\ell_1,\ell_2,\ldots \text{ disjoint,} \\ \text{of length} \ge \varepsilon N/2}} \prod_{i} (2d/|S|^{1-\alpha})^{\ell_i}$$

$$\le d^N \cdot 2^N \cdot (2d/|S|^{1-\alpha})^{\sum_i \ell_i} \le (2d)^N (2d/|S|^{1-\alpha})^{\varepsilon N/2}$$

which is exponentially small in N for $|S| > (2d \cdot (2d)^{2/\varepsilon})^{1/(1-\alpha)}$. □

This completes the proof of Theorem 3.1. □

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*.[1] Using a method of Alon, Goldreich, Håstad and Peralta [AGHP92] 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.

3.4 Small-Biased Random Trees as Potent Trees

In order to agree on an 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 [AGHP92].

Definition 3.4 (ϵ-biased sample space [NN90, AGHP92])**.** *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*

$$|\Pr[y=0] - \Pr[y=1]| \le \epsilon.$$

[1] This is similar to the case of random codes, see e.g., [Gal68].

We use [AGHP92, 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 $x \pmod p$. 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 3.5 ([AGHP92], Proposition 2)**.** *The sample space $\mathbf{B}_n$ is $\frac{n-1}{\sqrt{p}} + \frac{n}{p}$-biased with respect to linear tests.*

A proof appears in [AGHP92].

The above is used 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. We describe the tree as the $d^N \log|S|$-bit string obtained 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 3.5. *A d-ary* Small-Biased Tree Code *(SBTC) of depth N, is a tree described by a sample from the ϵ-biased sample space $\mathbf{B}_n$ with $n = d^N \log|S|$, $\epsilon = 2^{-cN\log|S|}$ 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 3.6 (almost k-wise independence [AGHP92])**.** *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,*

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

Due to a lemma by Vazirani [Vaz86] (see also Corollary 1 in [AGHP92]), 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 3.6. *Let $\mathcal{T}$ be a d-ary SBTC of depth N, then any $1 \leq k \leq d^N$ 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 element-wise efficient. 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 3.7. *Suppose $\varepsilon, \alpha \in (0,1)$. A d-ary SBTC of depth N over alphabet $|S| > (2d)^{(2+2/\varepsilon)/(1-\alpha)}$ is (ε, α)-potent with probability $1 - 2^{-\Omega(N)}$.*

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. The technique is similar to the proof of Theorem 3.1 for the case of RTC.

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 that forms a bad interval of length l. Denote by $\mathsf{TCenc}_l(u)$ the last l labels of $\mathsf{TCenc}(u)$. Since the tree is $(2^{-cN \log |S|}, 2l \log |S|)$-independent, then $\mathsf{TCenc}_l(u)$ and $\mathsf{TCenc}_l(v)$ are almost independent.

Lemma 3.8. *For any two nodes v, u in a SBTC that are at the same depth and have a common ancestor l levels away,*

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

Proof. Note that $\mathsf{TCenc}(u)$ and $\mathsf{TCenc}(v)$ are identical except for the last l labels. As noted in Corollary 3.6, the labels are almost independent (Definition 3.6) and we can bound the

probability $\Pr[\Delta(\mathsf{TCenc}(u), \mathsf{TCenc}(v)) = j]$ by

$$\sum_{\xi_u, \xi_v} \Pr[\mathsf{TCenc}_l(u) = \xi_u, \mathsf{TCenc}_l(v) = \xi_v]$$
$$\times \Pr[\Delta(\mathsf{TCenc}_l(u), \mathsf{TCenc}_l(v)) = j \mid \mathsf{TCenc}_l(u) = \xi_u, \mathsf{TCenc}_l(v) = \xi_v]$$
$$\leq (2^{-2l \log |S|} + 2^{-cN \log |S|}) 2^{2l \log |S|} \binom{l}{l-j} \left(\frac{1}{|S|}\right)^{l-j} \left(\frac{|S|-1}{|S|}\right)^{j}.$$

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, which holds for any $c > 2$ since $l \leq N$. □

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

Corollary 3.9.

$$\Pr[\Delta(\mathsf{TCenc}(v), \mathsf{TCenc}(u)) \leq \alpha l] \leq 2\frac{2^l}{|S|^{(1-\alpha)l}} .$$

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

Assume that the tree is bad, that is, there exists a path from the root to some node z with bad intervals of total length εN. Due to Lemma 3.4, there must exist *disjoint* bad intervals of total length at least $\varepsilon N/2$. Note that there are at most $\sum_{j=\varepsilon N/2}^{N} \binom{N}{j} \leq 2^N$ ways to distribute these disjoint intervals along the path from root to z.

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 [Sch96], 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

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

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

3.5 Potent Tree Codes with Arbitrarily Small ϵ from Linear-SBTC

So far we have assumed that ϵ, α are constant. Observe that for the case that $\epsilon \to 0$, the above construction gives an alphabet of size $|S| = d^{\Omega(1/\epsilon)}$, which is not constant. 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 3.10. *For any $\alpha \in (0,1)$ and $d \in \mathbb{N}$, there exists a constant alphabet size $|S|$ such that for any ε there exists an efficient construction of a d-ary (ε,α)-potent tree-code of depth N over S. The construction succeeds with probability at least $1 - 2^{-\Omega(\varepsilon N)}$.*

Proof. 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 [Sch03]: we find a 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 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 tree codes.

In the second part we construct a linear-SBTC $\mathcal{T}_2$ in the following way. Consider the small-biased random[2] lower-triangular matrix $G_{N\times N}$ over some finite field $\mathbb{F}$, say of characteristic

[2]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 $n = O(N^2\log|\mathbb{F}|)$ and bias $2^{-\Omega(N\log|\mathbb{F}|)}$.

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.

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 has bad sub-paths of total length at least εN, and thus it is α-bad in both $\mathcal{T}_1$ and $\mathcal{T}_2$. We focus on $\mathcal{T}_2$ for the rest of the analysis.

Using Lemma 3.4, there exist disjoint 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 $l_i > \log N$, and without loss of generality we can assume $n \leq \varepsilon N / 2 \log N$. Trivially, there are at most $N^{2n} \leq 2^{\varepsilon N}$ ways to distribute the intervals $l_1, \ldots, l_n$ along $[1, N]$.

Next, we show an analog for Lemma 3.8 and bound the probability that two divergent paths have a Hamming distance j.

Lemma 3.11. *For any two nodes v, u in a linear-SBTC that are at the same depth and have a common ancestor l levels away,*

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

Proof. First, observe that due to the linear structure of the tree,

$$\Delta(\mathsf{TCenc}(u), \mathsf{TCenc}(v)) = \Delta(\mathsf{TCenc}(\bar{0}), \mathsf{TCenc}(u - v)).$$

Let $z = u - v$ and recall that z is non-zero only in the last l labels. It is easy to verify that, due to our construction and the fact that G is small-biased, the labels on any path that diverges from the $\bar{0}$ path, are almost k-independent (for any $k < N$). Then, the probability for $\Pr[\Delta(\mathsf{TCenc}(\bar{0}), \mathsf{TCenc}(z)) = j]$ is

$$\sum_{\xi} \Pr[\mathsf{TCenc}_l(z) = \xi] \Pr[\Delta(\mathsf{TCenc}_l(\bar{0}), \mathsf{TCenc}_l(z)) = j \mid \mathsf{TCenc}_l(z) = \xi],$$

which is bounded by

$$(2^{-l \log |S|} + 2^{-cN \log |S|}) 2^{l \log |S|} \binom{l}{l-j} \left(\frac{1}{|S|}\right)^{l-j} \left(\frac{|S|-1}{|S|}\right)^{j}.$$

In a similar way to the standard SBTC, we can bound this probability by $2\binom{l}{l-j} \left(\frac{1}{|S|}\right)^{l-j}$. $\square$

In a similar way to the analysis performed above, let v_i be the node on the $\bar{0}$ at the end of the i-th interval. It follows that the probability that v_i is α-bad of length l_i (with respect to some node u) equals the probability of the divergent path to have Hamming distance at most αl_i, which is bounded by $\sum_{j=0}^{\alpha l_i} 2\binom{l}{l_i-j} \left(\frac{1}{|S|}\right)^{l_i-j} \leq 2 \cdot 2^{l_i} / |S_2|^{(1-\alpha) l_i}$.

Using a union bound on all d^{l_i} possible nodes u whose least common ancestor with v_i is l_i levels away, we bound the probability of v_i to be α-bad of length l_i by $2(2d/|S_2|^{1-\alpha})^{l_i}$. Finally,

$$\begin{aligned}
\Pr[\mathcal{T} \text{ is } \alpha\text{-bad }] &\leq \sum_{\substack{l_1, l_2, \ldots \text{ disjoint,} \\ \text{of length} \geq \varepsilon N/2}} \prod_i 2(2d/|S_2|^{1-\alpha})^{l_i} \\
&\leq 2^{\varepsilon N} 2^n (2d/|S_2|^{1-\alpha})^{\sum_i l_i} \\
&\leq (16d/|S_2|^{1-\alpha})^{\varepsilon N/2}.
\end{aligned}$$

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

CHAPTER 4

Interactive Communication with Random Noise: Efficiency

In this chapter we discuss interactive communication protocols over channels with random noise. We begin by recalling a good emulation scheme by Schulman [Sch96] that was designed for channels with random errors (the scheme also works for channel with adversarial errors for noise rates $\eta < 1/240$). The scheme crucially uses (standard) tree codes, and as such is not efficient. We show that it is possible to replace the tree code with a potent code. By doing so we achieve an *efficient* scheme for the case of random noise (Section 4.1).

In Section 4.2 we extend the discussion to the multiparty case, where m parties perform an interactive computation of some function on their joint inputs. The underlying scheme is an emulation by Rajagopalan and Schulman [RS94]. Similarly to the above, we show that it is possible to replace the tree code used in [RS94] by a potent code, and obtain an efficient scheme. However, we note that this emulation is not "good" as its blowup is $\log m$ in the worst case, rather than constant.

The results of this chapter are based on [GMS11, GMS14].

4.1 The Two-Party Case

4.1.1 Schulman's Emulation Scheme [Sch96]

Let us re-state the notion of a (noiseless) protocol π for interactive communication, using the notations used in [Sch96]. Assume that the parties A and B hold inputs x_A and x_B, respectively. 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 $\pi(x, m_1 \ldots m_t)$.

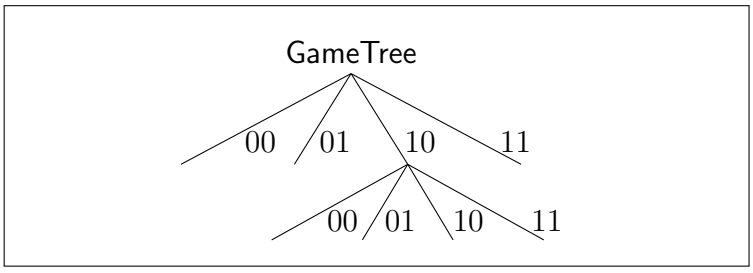

Figure 4.1: *An illustration of the GameTree.*

The computation over a noiseless channel (the transcript) can be described as a single route $\gamma_{\pi,x}$ along the GameTree, a 4-ary tree of depth T (see Figure 4.1). The path $\gamma_{\pi,x}$ begins at the root of the tree and the t-th edge is determined by the 2 bits exchanged at 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.

Our emulation process for any two-party interactive protocol π follows a method by Schulman [Sch96]. The idea behind the emulation is that each user keeps a record of (his belief of) the current progress of π, described as a pebble on one of GameTree's nodes. The users update their belief of the current progress of π according to their inputs and the messages received so far.

The information communicated at each round (for user $i \in \{A, B\}$) consists of two parts. (i) the message defined by π for that user; according to our assumption, this is a single bit $b = \pi_i(x_i, v)$, where v is the most updated position[1] of the user's pebble on the GameTree. (ii) a "control" information, which describes the change of the pebble according to π's progress. At each round the pebble can move to any of its adjacent nodes (including

[1]I.e., the position after moving the pebble according to the control part of the same round.

its father), or to stay at the same place. This gives 6 possible movements. Along with the bit b, each user sends one of 6×2 possible values.

The movement of the pebble is determined in the following way. First, the user tries to guess the position of the other users' pebble, according to all the messages he received so far (each message contains a single move for that pebble). Since the channel might cause some errors, that inferred position (the *guess*) is possibly incorrect. The user then moves his own pebble according to this guess: if both user's pebbles are at the same place, the user moves his pebble one step down along the path $\gamma_{\pi,x}$. Observe that the next edge of $\gamma_{\pi,x}$ is well defined via the bit b sent by each user in the previous round. On the other hand, when an inconsistency is suspected, the user "backtracks" his pebble to the least common ancestor of both pebbles. This is done by either keeping one's own pebble at the same position or moving it one step towards the root.

Informally speaking, the emulation works since the least common ancestor always lies along the path $\gamma_{\pi,x}$. If both users have the correct guess for the other party's pebble, they will correctly emulate one step of π and their pebbles move down 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.

The second ingredient needed for the emulation process is an (online) encoding procedure for the messages exchanged between the users. This is done using a tree code in the following manner. Each one of the 12 possible values represents a child in a 12-ary tree code called the StateTreeCode (Figure 4.2). The user begins at the root of the StateTreeCode. In order to send the value $v \in \{1, 2, ..., 12\}$, the user moves to the v-th child of his current position, and communicates the label associated with that edge over the noisy channel.[2] The current StateTreeCode's node of each user is called the user's *state*.

[2]Labels are not sent 'as-is', but are encoded via a standard (Shannon) error-correction code, with parameters that correspond the capacity of the noisy channel. This yields a constant dilation over the size of the labels (Lemma 2.1). A channel error is to be understood as the failure of the Shannon code. Throughout this chapter, any transmission of a label is to be understood in this manner.

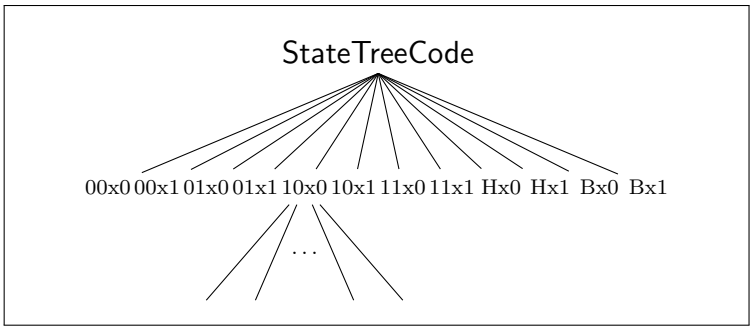

Figure 4.2: *An illustration of the StateTreeCode. The first value is the movement of the pebble according to the 4 possible children of the current pebble position, 'B' indicates a move to the father node and 'H' indicates no change in pebble's position. The second value is the output bit b defined by π for that user, assuming π's progress is according to the pebble's position.*

As hinted in Section 3, Encoding messages via a tree code has a 'self-healing' property: due to the large Hamming distance between any two divergent paths, decoding a divergent path whose least common ancestor is more than ℓ levels away, happens only if more than $\alpha\ell$ errors have happened during the last ℓ rounds. Hence, as the protocol continues, the decoded path does not diverge "too much" from the correct path except with a negligible probability. Since the diversion is small, also the guessed position of the other party's pebble is not too far away from the correct position, and eventually the protocol is able to backtrack to the correct progress.

The formal details of the emulation process are given in Figure 4.3 (described for user A; the protocol for B is identical). It is shown in [Sch96] that, assuming a tree code with distance $\alpha > 0$, the emulation process succeeds with probability $1 - 2^{-\Omega(\alpha N)}$.

4.1.2 Efficient Emulation for 2-party Interactive Communication with Random Errors

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

Begin with own pebble at the root of GameTree and own state S_A at the StateTreeCode root's child labeled $H \times \pi_A(x_A, \emptyset)$. Repeat the following $N = 5T$ times[3]:

1. Send the last label of $\mathsf{TCenc}(S_A)$ to party B.
2. Given the sequence of messages Z received so far from party B, guess the current state g of B as the node that minimizes $\Delta(\mathsf{TCenc}(g), Z)$. From g, infer party B's assumed pebble position, pebble(g), as well as party 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.

Figure 4.3: *The interactive protocol* Sim_π *for emulating* π *over noisy channels [Sch96], described for user A.*

Theorem 4.1. *Given a* $(\frac{1}{20}, \alpha)$*-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 4.3) efficiently emulates* π *over a BSC, takes* $N = O(T)$ *rounds and succeeds with probability* $1 - 2^{-\Omega(T)}$ *over the channel errors, assuming small enough label error probability.*

In Section 4.1.3 we show a decoding procedure that is efficient on average, given that the tree is SBTC. This immediately leads to the following Theorem..

Theorem 4.2. *There exists an efficient emulation that computes any distributed 2-party protocol* π *of length* T*, using a BSC for communication and a pre-shared SBTC. The emulation has a constant dilation and it 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 4.1 and later complete the formal proof. We begin by defining a good move: a move that advances the emulation of π in one step, and a bad move: an erroneous step in the emulation that requires backing up and re-emulating 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.

Recall the following properties of the emulation process Sim_π [Sch96].

Lemma 4.3. *The least common ancestor of the two pebbles lies on* $\gamma_{\pi,x}$.

Lemma 4.4. *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 [Sch96].

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 emulation 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 emulated.

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 StateTreeCode, are distanced l levels away (we call this event an error of *magnitude* l). Define the *error interval* (of length l) corresponding to the erroneous guess as $[t - l, t]$.

[3]For the emulation 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.

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

Proof. **(Theorem 4.1)**. Suppose the parties share a $(\frac{1}{20}, \alpha)$-potent tree code[4], for some $0 < \alpha < 1$. Assume that a specific run of the emulation failed. It follows that more than $N/5$ bad-moves have occurred. Without loss of generality, assume that at least $N/10$ of the bad-moves are due to decoding errors performed by the first user.

For that specific user, any possible path between the root of the $\mathsf{StateTreeCode}$ to one of its leaves contains at most $N/20$ α-bad nodes, due to the tree's potency. We can account each such bad node towards a bad move, which implies that at least $N/20$ additional bad moves have occurred when decoding non α-bad nodes of the $\mathsf{StateTreeCode}$. We show that the probability to have at least $N/20$ additional bad moves in the remaining (non-bad) 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, $\mathsf{TCenc}(s)$ and $\mathsf{TCenc}(r)$ 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.

Note that each such bad move (i.e., a tree-decoding error of a non α-bad node) induces an error interval of length $l_i \geq 1$, and the union of these intervals is of length at least $N/20$. Each such error interval happens with probability at most $\sum_{j=\alpha l_i/2}^{l_i} \binom{l_i}{j} p^j \leq 2^{l_i} p^{\alpha l_i/2}$, where p is the probability of having a channel error (recall that a channel error means a failure of the Shannon code used to encode labels.) Due to Lemma 3.4 we can find a set of disjoint intervals of length at least $N/40$. Since the intervals are disjoint, the channel errors that

[4]Theorem 3.7 guarantees that as long as $|S| > (2d)^{42/(1-\alpha)}$, only exponentially-small fraction of the SBTCs are $(\frac{1}{20}, \alpha)$-bad. Therefore, for obtaining a potent tree with overwhelming probability, we require $\log |S| \geq 193$.

cause each interval 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* one of the possible disjoint channel-errors patterns of total length at least $N/40$ along the bad moves associated with nodes which are not α-bad, by using a union bound over all the $\sum_{j=N/40}^{N} \binom{N}{j} \leq 2^N$ possible channel-error patterns. The probability is bounded by

$$\sum_{\text{user } U} \sum_{\substack{\text{pattern of} \\ l \geq N/40 \text{ errors}}} (2p^{\alpha/2})^l \leq 2^N (2p^{\alpha/2})^{N/40},$$

which is $2^{-\Omega(N)} = 2^{-\Omega(T)}$ for $p < 2^{-82/\alpha}$. □

The constraint on p implies a large block length of the underlying Shannon code, which might impact the overall efficiency of the scheme. We note that we do not try to optimize the rate of the scheme but only keep it positive.

4.1.3 Efficient Decoding

A decoding process outputs a node u (at depth t) that minimizes the Hamming distance $\Delta(\mathsf{TCenc}(u), \mathbf{r})$, where $\mathbf{r} = r_1 r_2 \cdots r_t$ is the received string of labels. Although the above Theorem 4.1 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 [Sch96] (which are based on ideas from [Woz57, Rei60, Fan63]), and show an efficient decoding that succeeds if the node is not α-bad. While the decoding process of [Sch96] is based on the fact that the underlying tree is a tree code, in our case the tree code is a SBTC. The same proof also applies to linear-SBTC discussed in section 3.5, and to the special case of RTC discussed in section 3.3.

The decoding procedure is described in Figure 4.4.

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 $g_1, g_2, \ldots, g_{t-1}$.

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}.

2. Recall that $\mathsf{TCenc}_m(u)$ denotes the m-suffix of $\mathsf{TCenc}(u)$, 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 $\Delta(r_i r_{i+1} \cdots r_t, \mathsf{TCenc}_{t-i+1}(g_t)) \geq \alpha(t-i)/2$. For that specific i, exhaustively search the subtree of g_i and output the node u (at depth t) that minimizes the Hamming distance $\Delta(r_1 r_2 \cdots r_t, \mathsf{TCenc}(u))$.

Figure 4.4: *An efficient decoding scheme for tree-codes. The scheme is correct for any non α-bad node.*

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 $\Delta(\mathsf{TCenc}_{l'}(g_t), \mathsf{TCenc}_{l'}(g'_t)) \geq \alpha l'$. Therefore, if all the suffixes of length $l' > l$ satisfy $\Delta(r_{t-l'+1} \cdots r_t, \mathsf{TCenc}_{l'}(g_t)) < \alpha l'/2$, 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}, 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 4.5. *Assume a SBTC (standard or linear) is used to communicate the string* $\mathsf{TCenc}(v)$ *over a memoryless noisy channel. 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 $\Delta(\mathbf{r}, \mathsf{TCenc}(u)) \leq \Delta(\mathbf{r}, \mathsf{TCenc}(v))$. This happens when the Hamming distance $\Delta(\mathsf{TCenc}(u), \mathsf{TCenc}(v))$ 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 k-wise independence of the tree's labels. Recall that the probability for a given Hamming distance between $\mathsf{TCenc}(u)$ and $\mathsf{TCenc}(v)$ is bounded by Lemma 3.8 (for SBTC) and by Lemma 3.11 for a linear-SBTC. 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 (label) errors have occurred. Using a union bound for every possible node u, the probability of part (i) is bounded by

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

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, $\Delta(r_{t-l+1} \cdots r_t, \mathsf{TCenc}_l(u)) < \alpha l/2$. However, since u does not minimize the Hamming distance, there must exist a node z of distance at most l, such that $\Delta(\mathsf{TCenc}_l(z), r_{t-l+1} \cdots r_t) \leq \Delta(r_{t-l+1} \cdots r_t, \mathsf{TCenc}_l(u))$. By the triangle inequality, $\Delta(\mathsf{TCenc}_l(z), \mathsf{TCenc}_l(u)) \leq \alpha l$. 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 4.1 is still valid, since it only requires the decoding procedure to succeed when the node is not α-bad (and assumes that the emulation has a bad move in each node which is a bad node).

We now show that this procedure is efficient in expectation, and complete the proof for Theorem 1.1 and Theorem 4.2.

Proposition 4.6. *Assume a string of length N is encoded via a SBTC with alphabet S and sent over a channel with error probability $p < |S|^{-2}$ per label. Then, decoding the string, bit-by-bit, using the decoding procedure in Figure 4.4 takes $O(N)$ time, in expectation.*

Proof. Let $L(t)$ be the depth of the subtree explored at time t. Then, 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 time t, if $L(t) = l$ then

$$\Delta(r_{t-l+1} \cdots r_t, \mathsf{TCenc}_l(g_t)) \geq \alpha l/2,$$

yet for $l' > l$,

$$\Delta(r_{t-l'+1} \cdots r_t, \mathsf{TCenc}_{l'}(g_t)) < \alpha l'/2,$$

thus $\Delta(r_{t-l+1} \cdots r_t, \mathsf{TCenc}_l(g_t)) = \lceil \alpha l/2 \rceil$. Let the transmitted sequence of labels be $\mathsf{TCenc}(v)$ for some node v of depth t. A Hamming distance of exactly $\lceil \alpha l/2 \rceil$ happens with probability

at most

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

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

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

The same proof holds for a linear-SBTC with alphabet S, and therefore also for the tree $\mathcal{T}$ of Section 3.5. Finally, we mention that [Sch96] presents a data structure which allows us to perform the above decoding procedure with overhead $O(L(t))$. □

4.2 The Multi-Party Case

In this section we extend our result to support an emulation 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 emulating an m-party protocol over a disturbed channel developed by Rajagopalan and Schulman [RS94]. The paper [RS94] shows that a scheme for emulating 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 (those are called his *neighbours*).

4.2.1 The Rajagopalan-Schulman Emulation Scheme [RS94]

Rajagopalan and Schulman describe how to adapt the 2-party emulation of [Sch96] to an arbitrary number of users. The key idea is to replace the 12-ary StateTreeCode 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 emulated round, and bkp indicates that the last emulated round is suspected to be invalid and should be deleted and re-emulated ("back up" one level). For instance, if a party receives "$0, 0, 1, bkp, 1, bkp, 0$", he will interpret the emulation as just finishing the 3rd round, where the transcript so far is "$0, 0, 0$".

The emulation (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 emulated round of π, and sends his output bit to user j (by communicating the label assigned with the arc to child 0 or 1 respectively, in the ternary StateTreeCode 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. Inconsistency is defined as one of the two following cases: (1) the current decoded transcript of the StateTreeCode disagrees with the bits sent so far, or (2) the user received bkp from one of his neighbors.
- The user shares a ternary tree code with each of his r neighbours, 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.

We refer the reader to [RS94] for the complete details.

4.2.2 Efficient Emulation for Multi-Party Interactive Communication with Random Errors

One can easily check that the bulk of the analysis performed in [RS94] hold also in the case of replacing the 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 emulates step $t - l$ of π then at least $l/2$ errors have occurred in decoding the correct tree-node during 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 tree code with a SBTC, in order to complete the proof, we must adapt the second part to using a SBTC. This is done by Lemma 4.9 below.

Let us formally describe these two parts. We begin by defining all the transmissions that affect the user state at time t. Let (p, t) denote a user p at time t.

Definition 4.1 ([RS94])**.** *(p, t) and (p', t') are **time-like** if messages sent by user p at time t has an effect 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 4.2 ([RS94])**.** *A t **time-like path** is a sequence $\{(p_i, i) \mid 1 \leq i \leq t\}$ 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 correctness of the multiparty protocol of [RS94] follows from the next two lemmas

Lemma 4.7 (Lemma (5.1.1) of [RS94])**.** *If a user p at time t has successfully emulated 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 4.8 (Lemma (5.1.2) of [RS94])**.** *For 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 emulation 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 4.8 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 4.7 holds regardless of the tree code in use, we prove a variant of the above Lemma 4.8 for the case of using a potent tree code. Moreover, although Lemma 4.8 holds for any time $1 \leq t \leq N$, only $t = N$ is required for completing the proof for the multiparty case, which we prove in the following lemma.

Lemma 4.9. *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. 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 3.4, 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[5], and since the intervals are disjoint, they are independent. As above (see proof of Theorem 4.1), 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

[5]This claim also applies to the efficient decoding procedure, as it always returns the node that minimizes the Hamming distance, if it is not α-bad. See the proof of Theorem 4.1 and discussion in Section 4.1.3.

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 $j \geq N/4$ 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 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 $\frac{1}{(2(r+1))^N}$.

The above lemma replaces Lemma 5.1.2 of [RS94], and leads to the following theorem.

Theorem 4.10. *For any m-party protocol π of length T defined over a noiseless channel, there exists an efficient emulation over a memoryless noisy channel, that emulates π with probability $1 - 2^{-\Omega(T)}$ and has a dilation of $O(m)$.*

Proof. Suppose each two users share a SBTC[6] with $|S| \geq ((2d)^{32m+2})^{1/(1-\alpha)}$ for some $\alpha \in (0,1)$ (as before, it is possible to begin the emluation by communicating the tree code, instead of assuming it is pre-shared). By Theorem 3.7 the SBTC is $(\frac{1}{16m}, \alpha)$-potent with overwhelming probability. Note that each label takes $\log |S| = O(m)$ bits. By Lemma 2.1, it is possible to use an error correcting code such that each encoded transmission is $O(m)$ and the label error probability is $p \leq (5(r+1))^{-32/\alpha}$. Note that in order to obtain efficient decoding it is required that $p < |S|^{-2}$, which can also be achieved while keeping the length of the encoded label $O(m)$.

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 $1/(2(r+1))^N$ over the channel errors. Therefore, the emulation fails with only a negligible probability since, if we set $t = N = 2T$ in Lemma 4.7, then at least $N/4$ tree-decoding errors must have occurred, which completes the proof. □

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

Finally, we obtain Theorem 1.2 by using the potent-tree construction described in Section 3.5 and setting $\varepsilon = O(1/m)$. The construction yields a $(1/16m, \alpha)$-potent tree code with a constant size alphabet S, and succeeds with probability $1 - 2^{-\Omega(N/m)}$. We can furthermore encode each label via Theorem 2.1 and achieve a label error-probability $p < (5(r+1))^{-32/\alpha}$ with a dilation of $O(\log(r+1))$.

CHAPTER 5

Interactive Communication with Adversarial Noise: Resilience

We now turn to the adversarial noise model, where the only limit on the noise is the total noise rate η. In the plain model, it is known that $\eta = 1/4$ is a tight bound on the rate of noise: for any function f and any $\varepsilon > 0$ there exists a protocol that correctly computes f as long as $\eta < 1/4 - \varepsilon$ [BR11]; on the other hand, for noise $\eta \geq 1/4$ some functions cannot be computed [BR11] (to see this impossibility, consider the party that speaks less in the entire protocol, and assume the channel corrupts half of that party's transmissions.)

In this chapter we wish to get better noise resilience and "break" the $\eta = 1/4$ bound by relaxing the setup assumptions of the model, and assuming the *shared randomness model* (Section 2.5). Specifically, we allow the users to share some private randomness, unknown to the adversarial channel Eve.

We show that assuming shared randomness changes the noise bound to $\eta = 1/2$. That is, for adversarial noise rate of $1/2$ or higher, no constant-rate protocol can compute functions that require interaction between the parties, while for any function f there is a protocol that correctly computes f if the adversarial noise rate is below $1/2$. Formally, we show the following separation theorems,

Theorem 5.1. *For any function f which depends on both x and y, the following holds. If the adversarial corruption rate is $\frac{1}{2}$ or higher then no constant-rate interactive protocol correctly computes f with probability higher than the probability of guessing $f(x, y)$ given only the input x (or only the input y).*

Theorem 5.2. *For any constant $\varepsilon > 0$ and for any function f, there exists an interactive protocol with constant overhead such that if the adversarial corruption rate is at most $\eta = \frac{1}{2} - \varepsilon$, the protocol outputs f with overwhelming probability over the shared random string R.*

The proof of the impossibility part is rather simple:

Proof. (**Theorem 5.1**) Assume that the protocol takes T rounds. Furthermore, recall that in our model it is assumed that at each round both parties send exactly one message.[1] Hence and without loss of generality, Alice is the sender of at most $T/2$ of the transmissions. Eve corrupts all the transmissions originated by Alice. Effectively, the unidirectional channel from Alice to Bob has zero capacity, and it cannot be that Bob correctly computes $f(x, y)$ with probability higher than guessing $f(x, y)$ given only y. □

In the rest of this chapter we will construct a protocol that correctly computes any $f(x, y)$ with overwhelming probability as long as the adversarial corruption rate is $\frac{1}{2} - \varepsilon$ for $\varepsilon > 0$. The protocol is based on an emulation process by Braverman and Rao [BR11] that resists noise rates up to $1/4 - \varepsilon$ in the plain model. We begin by describing the *Blueberry code* (Section 5.1), a random error-detection code that effectively doubles the noise-resilience by detecting most of the adversarial corruptions. Later, we concatenate the Braverman-Rao protocol (Section 5.2) with the Blueberry code, and prove Theorem 5.2 (Section 5.3).

The results of this chapter are based on [FGOS13].

[1] The proof also holds for protocols for which there exists a function Next(i) which defines for each round i, which of the parties sends a message, and is independent of the messages sent so far (these kind of protocols are called *oblivious* in [BR11]). In that case there exists one party that communicates at most $T/2$ messages at rounds known to Eve in advance.

On the other hand, the proof does not hold for the most general model, in which the protocol *adaptively* determines who is to speak next, possibly according to the noise observed so far. Indeed, adaptive protocols may resist a higher level of noise, see [GHS13, AGS13].

5.1 The Blueberry Code

The main ingredient of our construction is an error-detection code we name the *Blueberry code*[2]. The Blueberry code uses the shared randomness in order to detect corruptions made by the channel, and marks them as *erasures*. One can think about this code as a weak message authentication code (MAC) that authenticates each symbol separately with a constant probability (see [Gol04] for a formal definition of MAC). To this end, each symbol of the input alphabet Σ is randomly and independently mapped to a larger alphabet Γ (the channel alphabet). This means that only a small subset of the channel alphabet is meaningful and the other symbols serve as "booby-traps". Since each symbol is encoded independently, any corruption is caught with constant probability $\frac{|\Sigma|-1}{|\Gamma|-1}$ and marked with a special sign $\perp$ to denote it was corrupted by the channel (this will be called an *erasure*). Most of the corruptions made by an adversary become erasures and only a small fraction (arbitrarily small, controlled by the size of $|\Gamma|$) turns into errors.

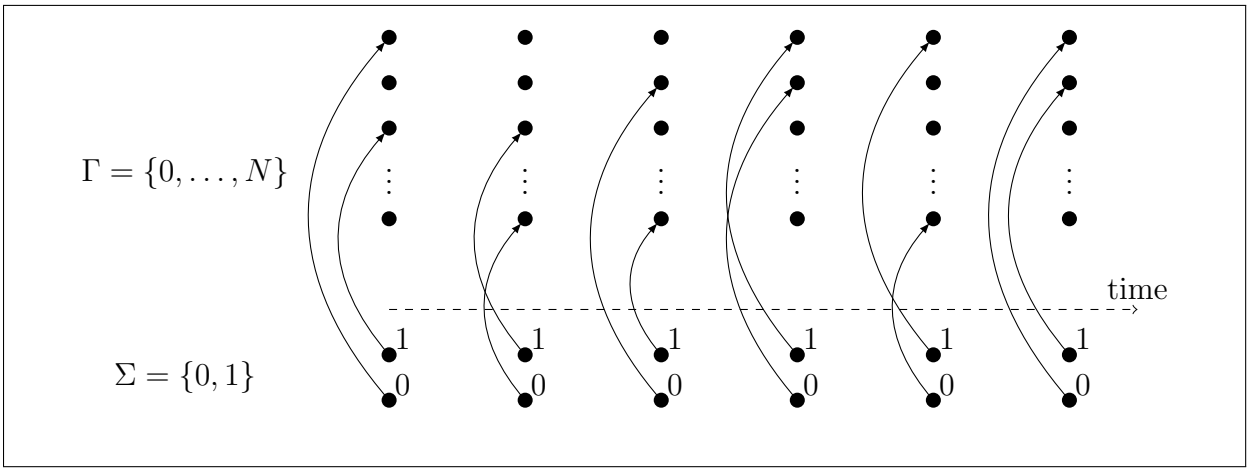

Figure 5.1: *A demonstration of the Blueberry code: at any given time each symbol in Σ is randomly mapped to a symbol of Γ. Symbols of Γ with no incoming arrow are "booby-traps", which detect corruptions.*

The main insight is that erasures are more easy to correct than errors: Indeed, assume

[2]The name of the Blueberry code is inspired by the children's book "The case of the hungry stranger" [Bon63] in which a blueberry pie is gone missing, and the thief (who turns out to be the dog) is identified by his big blue grin.

that the Hamming distance of two strings, x and y, is m. Then if x was communicated but y is decoded it means that at least $m/2$ errors have occurred, or alternatively, at least m erasures. More generally, assuming we decode by minimizing the Hamming distance, then our decoding fails if the number of errors e and the number of erasures d satisfy $2e + d \geq m$.

Definition 5.1. *For $i \geq 1$ let $B_i : [L+1] \to [L+1]$ be a random and independently chosen permutation. The* Blueberry code *maps a string x of arbitrary length n to*

$$B(x) = B_1(x_1)B_2(x_2)\cdots B_n(x_n).$$

We denote such a code as $B : [L+1]^* \to [L+1]^*$.

We use the Blueberry code in the shared-randomness model where the legitimate parties share the random permutations B_i, unknown to the adversary. Although B_i is a permutation on $[L+1]$, we actually use it to encode strings over a smaller alphabet $[S+1]$ with $S < L$; that is, we focus on the induced mapping $B : [S+1]^* \to [L+1]^*$. The adversary does not know the specific permutations B_i, and has probability of at most S/L to change a transmission into a symbol whose pre-image is in $[S+1]$.

Definition 5.2. *Assume that at some time i, $y_i = B_i(x_i)$ is transmitted and $\tilde{y}_i \neq y_i$ is received. If $B_i^{-1}(\tilde{y}) \notin [S+1]$, we mark the transmission as* an erasure *(specifically, the decoding algorithm outputs $\perp$); otherwise, this event is called* an error.

Corollary 5.3. *Let $x \in [S+1]^n$ and assume $B(x)$ is communicated over a noisy channel. Every symbol altered by the channel will cause either an* error *with probability S/L, or an* erasure *with probability $1 - S/L$.*

Assuming $S \ll L$, most of the corruptions done by the channel are marked as erasures, and only a small fraction of the corruptions percolate through the Blueberry code and cause an error.

Lemma 5.4. *Let $S, L \in \mathbb{N}$ be fixed and assume a Blueberry code $B : [S+1]^* \to [L+1]^*$ is used to transmit a string $x \in [S+1]^n$ over a noisy channel. For any constant $0 \leq c \leq 1$, if the channel's corruption rate c, then with probability $1 - 2^{-\Omega(n)}$ at least a $(1 - 2\frac{S}{L})$-fraction of the corruptions are marked as erasures.*

Proof. Denote by z_i the random variable which is 1 if the i-th corrupted-transmission is marked as an erasure and 0 otherwise. These are independent Bernoullis with probability $1 - \frac{S}{L}$. Let $Z = \sum_i z_i$ and note that $\mathbb{E}[Z] = cn(1 - \frac{S}{L})$. By Chernoff-Hoeffding inequality,

$$\Pr_R \left[\frac{1}{n} \sum_i z_i < c\left(1 - 2\frac{S}{L}\right) \right] < e^{-2n(cS/L)^2}.$$

□

Corollary 5.5. *Let $S, L \in \mathbb{N}$ be fixed. If out of n received transmissions, cn were marked as erasures by a Blueberry code $B : [S+1]^* \to [L+1]^*$, then except with probability $2^{-\Omega(n)}$ over the shared randomness, the adversarial corruption rate is at most $c/(1 - 2\frac{S}{L})$.*

We will use the Blueberry code concatenated with another (outer) code that is less sensitive to erasures than to errors. From the outer code's point of view, this effectively increases the channel's "error rate resilience" from $1-2c$ to $1-c(1+S/L)$. The construction of the code B from independent B_i's allows us to encode and decode each x_i independently, which is crucial for on-line applications in which the message x to be sent is not fully known in advance.

5.2 The Braverman-Rao Emulation Scheme

We now give an outline of the Braverman-Rao emulation process, and refer the reader to [BR11] for the full details (in order to be self contained, we also recall the protocol in Appendix A).

Recall that Alice and Bob wish to compute some function $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$, where Alice holds $x \in \mathcal{X}$ and Bob holds $y \in \mathcal{Y}$. The computation is performed interactively: at each round, both parties communicate a message which depends on their input and previous transmissions. Without loss of generality, we assume each message is a single bit, and that the protocol takes T rounds. At the end of the computation Alice outputs $z_A \in \mathcal{Z}$ and Bob outputs $z_B \in \mathcal{Z}$, and we say that f was correctly computed if $z_A = z_B = f(x, y)$. Without loss of generality we assume the output is a single bit, $|\mathcal{Z}| = 2$.

The users consider π as a binary tree $\mathcal{T}$: each root-leaf path in the tree describes a possible transcript of π, where odd levels describe Alice's outputs and even levels describe Bob's outputs. See Figure 5.2 for an illustration of a possible $\mathcal{T}$.

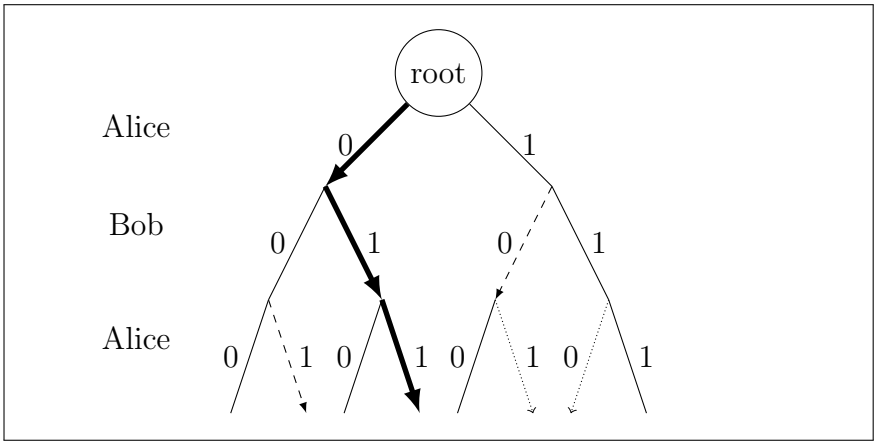

Figure 5.2: *A tree $\mathcal{T}$ illustrating a path P (bold edges) taken by Alice and Bob for computing $f(x, y)$ for some input. Dashed edges represent the hypothetical reply of Alice and Bob given that a different path P' was taken (when such replies are defined).*

The emulation process takes $R = O(T)$ rounds, in each of which each party sends one symbol and receives one symbol. The parties use a tree-code (or a potent tree code) to communicate the vertices of $\mathcal{T}$ according to their inputs. At each round of the emulation, the party decodes via the tree code all the received messages so far and obtains a set of edges in $\mathcal{T}$, which belong to the other party; then, it communicates one additional edge of $\mathcal{T}$ owned by him, that extends the longest consistent path on $\mathcal{T}$.

The encoding of the next edge is performed as follows. To simplify the explanation, assume that each transmission is over the alphabet $\Sigma = \{0, \ldots, N\} \times \{0, 1\}^{\leq 2}$. Intuitively, the transmission $(e, s) \in \Sigma$ means "extend the path P by taking at most two steps defined by s starting at the child of the edge I have transmitted at transmission number e". Each symbol (e, s) is communicated to the other side via a $|\Sigma|$-ary tree code with distance α and alphabet Γ. Although this alphabet is not of constant size, it is easy to obtain a constant size alphabet by encoding each (e, s) into a delimited binary string (see Section 6 in [BR11]). See Appendix A for the detailed protocol.

5.3 Resisting Noise Rate $\eta < 1/2$ in the Shared Randomness Model

The emulation protocol. The protocol for resisting higher rates of noise is simply obtained by performing the Braverman-Rao protocol, yet encoding each transmission using a Blueberry code. Specifically, each symbol (e, s) is first encoded via a $|\Sigma|$-ary tree code with distance α and alphabet Γ, and then encoded via a Blueberry code $B : [S+1]^* \to [L+1]^*$ with $|S+1| = |\Gamma|$ and S/L to be determined later. Thus, at time n Alice computes $a_n \in \Gamma$, the last symbol of

$$\mathsf{TCenc}((e,s)_1, \ldots, (e,s)_n) = a_1 a_2 \cdots a_n,$$

and communicates $B_n(a_n)$; Bob, after decoding the Blueberry code receives $\tilde{a}_n \in \Gamma \cup \{\perp\}$, possibly with added noise or an erasure mark (similarly, Bob's labels are b_n, and Alice receives a noisy or deleted version $\tilde{b}_n$).

We now analyze the above emulation protocol and prove Theorem 5.2.

Analysis. Let $\mathsf{TCdec}(\tilde{a}_1, \ldots, \tilde{a}_n)$ denote the string Bob decodes at time n (similarly, Alice decodes $\mathsf{TCdec}(\tilde{b}_1, \ldots, \tilde{b}_n)$). For every $i > 0$, we denote with $m(i)$ the largest number such that the first $m(i)$ symbols of $\mathsf{TCdec}(\tilde{a}_1, \ldots, \tilde{a}_i)$ equal to $a_1, \ldots, a_{m(i)}$ and the first $m(i)$ symbols of $\mathsf{TCdec}(\tilde{b}_1, \ldots, \tilde{b}_i)$ equal to $b_1, \ldots, b_{m(i)}$.

Define $\mathcal{N}(i, j)$ to be the "effective" number of adversarial corruptions in interval $[i, j]$: the number of erasures plus twice the number of errors in the $[i, j]$ interval of the simulation (for both users).

Definition 5.3. *Let*

$$\mathcal{N}_a(i,j) = |\{k \mid i \le k \le j, \tilde{a}_k = \perp\}| + 2|\{k \mid i \le k \le j, \tilde{a}_k \notin \{a_k, \perp\}\}|$$

be the noise on the channel from A to B, and similarly define $\mathcal{N}_b(i, j)$ as the noise on the other direction. The effective number of corruptions *in interval $[i, j]$ is*

$$\mathcal{N}(i,j) = \mathcal{N}_a(i,j) + \mathcal{N}_b(i,j).$$

We begin by showing that if $m(i) < i$ then many corruptions must have happened in the interval $[m(i)+1, i]$.

Lemma 5.6. $\mathcal{N}(m(i)+1, i) \geq \alpha(i - m(i))$.

Proof. Assume that at time i Bob decodes the string $\mathsf{TCdec}(\tilde{a}_1, \ldots, \tilde{a}_i) = \tilde{\sigma}_1, \ldots, \tilde{\sigma}_i$. By the definition of $m(i)$, $\tilde{\sigma}_1, \ldots, \tilde{\sigma}_{m(i)} = \sigma_1, \ldots, \sigma_{m(i)}$, and assume without loss of generality that $\tilde{\sigma}_{m(i)+1} \neq \sigma_{m(i)+1}$. Note that the Hamming distance between $\mathsf{TCenc}(\sigma_1, \ldots, \sigma_i)$ and $\mathsf{TCenc}(\tilde{\sigma}_1, \ldots, \tilde{\sigma}_i)$ must be at least $\alpha(i - m(i))$. It is immediate that for Bob to make such a decoding error, $\mathcal{N}_a \geq \alpha(i - m(i))$. □

Lemma 5.7 ([BR11])**.** *Let $t(i)$ be the earliest time such that both users announced the first i edges of P within their transmissions. For $i \geq 0$, $k \geq 1$, if $t(k) > i+1$, then $t(k-1) > m(i)$.*

Proof. The proof is taken from [BR11]: Without loss of generality, assume that the k-th edge of P describes Alice's move. Suppose $t(k-1) \leq m(i)$ and $t(k) > i+1$. Then it must be the case that the first $k-1$ edges of P have already been announced within the first $m(i)$ transmissions of both parties, yet the k-th edge has not. By the protocol definition, Alice will announce this edge at round $i+1$, in contradiction to our assumption that $t(k) > i+1$. □

Next we show that if at some time i the length of the proposed P is not long enough (less than k), then many transmissions must have been corrupted.

Lemma 5.8. *For $i \geq -1$, $k \geq 0$, if $t(k) > i+1$, then $\mathcal{N}(1, i) \geq \alpha(i - k + 1)$.*

Proof. We prove by induction on i and k. The claim vacuously holds for $k = 0$ and trivially holds for $i \leq 0$ since $\mathcal{N}(1, i)$ is non-negative. Otherwise, we have

$$\mathcal{N}(1, i) = \mathcal{N}(1, m(i)) + \mathcal{N}(m(i)+1, i).$$

The second term, by Lemma 5.6 gives $\mathcal{N}(m(i)+1, i) \geq \alpha(i - m(i))$. For the first term, Lemma 5.7 suggests that $t(k-1) > m(i)$ and we can use the inductive hypothesis with $i' = m(i) - 1$ and $k' = k - 1$ to get

$$\mathcal{N}(1, m(i)) \geq \mathcal{N}(1, i') \geq \alpha(i' - k' + 1) = \alpha(m(i) - k + 1).$$

□

The above Lemmas allow us to complete the proof of Theorem 5.2 by showing that if the simulation of P failed, there must have been "too many" corruptions.

Proof. (**Theorem 5.2**) Assume an unsuccessful run of the simulation protocol. That is, the simulation of the path P has failed, $m(N) < t(T)$. The number of adversarial corruptions throughout the protocol is given by $\mathcal{N}(1, N) = \mathcal{N}(1, m(N)) + \mathcal{N}(m(N) + 1, N)$ which by Lemma 5.8 and Lemma 5.6 is lower bounded by

$$\mathcal{N}(1, N) \geq \alpha(m(N) - T + 1) \ + \ \alpha(N - m(N)) \geq \alpha(N - T) \geq \alpha(N - (1 - \alpha)N) = \alpha^2 N.$$

Yet, assume the adversary is restricted to corrupt at most $c = 1/2 - \varepsilon$ fraction of the $2N = 2\lceil \frac{T}{1-\alpha} \rceil$ transmissions, then Lemma 5.4 guarantees that with overwhelming probability there will be at least $2cN(1-2S/L)$ erasures. This implies that with overwhelming probability $\mathcal{N}(1, N) \leq 2cN(1 + 2S/L)$. For any $0 < \varepsilon \leq 1/2$ we can choose constants $\alpha < 1$ and $L > S$ such that $\alpha^2 > (1-2\varepsilon)(1+2S/L)$ and conclude that the protocol succeeds with overwhelming probability over the shared randomness. □

5.4 Resisting Noise $\eta < 1/2$ over Erasure Channels in the Plain Model

We make the simple observation, that the shared randomness (i.e., the Blueberry code) is used for the main purpose of detecting Eve's noise, transforming corrupted symbols into erasures. With the above analysis, it immediately follows that the Braverman-Rao scheme [BR11] resists a maximal noise rate of $1/2 - \varepsilon$ over an *erasure channel*. Such channels are modeled as a function $\mathsf{ch} : \Sigma \to \Sigma \cup \{\perp\}$, where each transmitted symbol either goes through intact, or turns into a deletion mark $\{\perp\}$. The parties need not pre-share any randomness nor use the Blueberry code, since the adversary is restricted to only making erasures to begin with.

Corollary 5.9. *For any constant $\varepsilon > 0$ and for any function f, there exists an interactive protocol over an erasure channel, that has a constant rate and resists adversarial erasure rate of up to $\frac{1}{2} - \varepsilon$.*

CHAPTER 6

Efficient Coding Schemes for Data Streams

In this chapter we discuss the model of data stream. A *data stream* S is a (potentially infinite) sequence of elements $(x_0, x_1, x_2, \ldots)$ where $x_t \in \{1, \ldots, u\}$ arrives at time t. We assume the data stream is generated at Alice side, and should be communicated to Bob over a noisy channel.

As in previous sections, our communication model consists of a channel $\mathsf{ch} : \Sigma \to \Sigma$ subject to corruptions. The noise model is such that any symbol σ sent through the channel can turn into another symbol $\tilde{\sigma} \in \Sigma$. It is not allowed to insert or delete symbols. For all of our applications we assume that one symbol $\sigma_i \in \Sigma$ is sent at any time slot i.[1] We say that the adversarial *corruption rate* is c if for n transmissions, at most cn symbols were corrupted.

In Section 6.1 we consider the plain model, we show that via a simple usage of tree codes one can get a scheme that allows Bob to decode, at any time n, a prefix $(x_0, x_1, x_2, \ldots, x_{1-2cn})$ of the stream as long as the fraction of noise is $c < 1/2$. We also show that this result is tight for the plain model.

Next, in Section 6.2, we consider the shared randomness model, and show a scheme that allows Bob to decode a prefix $(x_0, x_1, x_2, \ldots, x_{1-cn})$ of the stream, as long as the noise is $c < 1$. This is obtained by concatenating tree codes with Blueberry codes: the Blueberry code prevents the adversary from corrupting too many transmissions without being noticed, and given that the noise level is low enough, the tree code correctly decodes a prefix of the stream whose length is determined by the average noise level up to that time.

[1] The channel time slots need not correspond with the times in which stream symbols are received. I.e, it is possible that between the arrival of stream elements x_i and x_{i+1}, several channel-symbols are transmitted.

The only caveat of the above construction is that tree code decoding is not necessarily efficient and may be, in the worst case, exponential in the length of the received transmission. In Section 6.3 we obtain an *efficient* authentication scheme by splitting the stream into small segments and repeatedly sending random segments of the history. That way, even if some part of the transmission was changed by the channel, the same information will keep being retransmitted at random future times, and eventually (with high probability) will be received at the other side intact.

Roughly speaking, we use $n/\log n$ tree codes to encode chunks of the stream (each of length roughly $\log n$). Note that as n grows, so does the number of the trees in use, and the expected depth of each tree. At each time step, we randomly select one of the $n/\log n$ trees and transmit the next label of the path defined by the corresponding chunk of the stream. For most of the trees, the expected number of labels transmitted is $\Theta(\log n)$, and the decoding of the specific chunk succeeds except with polynomially small probability. Since each tree code is used to encode a word of length $O(\log n)$, the decoding can be performed efficiently by an exhaustive search.

The results of this chapter are based on [FGOS13].

6.1 Error Correction of Data Streams

We begin with a simple, non-efficient, constant-rate error-correction scheme for data streams that withstands noise $c < 1/2$ and decodes a prefix of length $1 - 2c$ of the stream sent so far. The scheme is obtained by simply encoding the stream via a tree code $\mathcal{T}$ with large enough distance $\alpha \in (0, 1)$ and a constant-size alphabet.

Theorem 6.1. *For any constants $c < 1/2$ and $\varepsilon > 0$ there exists a constant-rate error-correction scheme for data stream $x_1, x_2, \ldots$ such that at any given time n the receiver outputs a string $x'_1, x'_2, \ldots, x'_n$, and if the noise rate until time n is at most c, then*

$$x'_1, x'_2, \ldots, x'_{(1-2c)n-\varepsilon n} = x_1, x_2, \ldots, x_{(1-2c)n-\varepsilon n}$$

that is, a prefix of the stream of length at least $(1 - 2c)n - \varepsilon n$ is correctly decoded.

Proof. Assume that Alice encodes each stream symbol using $\mathsf{TCenc}_{\mathcal{T}}()$ using some tree code $\mathcal{T}$ whose parameters we fix shortly.

For a specific time n, consider a string $\tilde{x} \in \{0,1\}^n$, such that $anc(x, \tilde{x}) \geq (2c+\varepsilon)n$. Due to the tree distance property, the Hamming distance between $\mathsf{TCenc}(\tilde{x})$ and $\mathsf{TCenc}(x)$ is at least $\alpha(2c+\varepsilon)n$. Assume that Eve causes e errors. A maximal-likelihood decoding will prefer x over $\tilde{x}$ as long as $\lfloor \alpha(2c+\varepsilon)n \rfloor > 2e$. Since Eve's corruption rate is limited to c, we know that $e \leq cn$. By setting $\alpha > \frac{2c}{2c+\varepsilon}$ we guarantee that $\alpha(2c+\varepsilon)n > 2e$, and Bob decodes a string x' such that $anc(x, x') < (2c+\varepsilon)n$ with certainty. □

6.2 Perpetual Authentication

Sending a data stream over a noisy channel is not a simple task, especially when the noise model is adversarial. Our goal is to design an encoding and decoding scheme such that the encoding has a constant rate and the decoding recovers the encoded transmitted stream, or else aborts. Furthermore, we wish an "authentication" guarantee, that is, if the decoding scheme did not abort, it decodes the *correct* data with high probability (note that the probability that the scheme aborts potentially differs from the probability that the decoding scheme outputs incorrect data). The amount of recoverable data depends on the noise and the goal is to output (and authenticate) the longest possible prefix of the stream, given a constant corruption rate.

Definition 6.1. *A* $(c(n), \gamma(n), \kappa(n))$-Streaming Authentication Scheme *with constant rate* r *is an encoding* $e : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^r$ *that encodes a stream* $x_1, x_2, \ldots$ *into a stream* $y_1 = e(x_1, R)$, $y_2 = e(x_1 x_2, R)$, $\ldots$, $y_i = e(x_1 \cdots x_i, R)$, *and a decoding* $d : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^* \cup \{\perp\}$ *such that the following holds. For any* n, *and for any adversary* $Adv(x_1 \cdots x_n, y_1 \cdots y_n) = y'_1 \cdots y'_n$, *either* $d(y'_1 \cdots y'_n, R) = x'_1 x'_2 \cdots x'_n$ *or* $d(y'_1 \cdots y'_n, R) = \perp$, *and if at most* $c(n)$ *transmissions were corrupted, then*

1. *the scheme aborts with probability at most* $\kappa(n)$,

$$\Pr_R[d(y'_1 \cdots y'_n, R) = \perp] < \kappa(n).$$

2. *if not aborted, the probability to decode an incorrect $\gamma(n)$-prefix of the stream is at most $\kappa(n)$,*

$$\Pr_R[d(y'_1 \cdots y'_n, R) \neq \perp \ \wedge \ x'_1 \cdots x'_{\gamma(n)} \neq x_1 \cdots x_{\gamma(n)}] < \kappa(n).$$

Eve is given both the raw stream and the channel transmissions, however she does not know the shared random string R used as the secret authentication key. It is desired that as long as Eve corrupts only a small fraction of the transmissions, Bob will be able to correctly decode a prefix of the stream, or otherwise be aware of the adversarial intervention and abort.

We show the following dichotomy: If the adversarial corruption rate is some constant c, then there exists a streaming authentication stream that decodes a prefix of at most $(1-c)$-fraction of the stream received so far. In addition, there does not exist a streaming authentication scheme that is capable of decoding a longer prefix with non-negligible probability.

Theorem 6.2. *In the shared-randomness model, for every constants c, ε such that $0 \leq c < 1$ and $0 < \varepsilon \leq (1-c)/2$ there exists a constant-rate $(cn, (1-c)n - \varepsilon n, 2^{-\Omega(n)})$-Streaming Authentication Scheme. Moreover, there exists an* efficient *constant-rate $(cn, (1-c)n - \varepsilon n, 2^{-\Omega(\log n)})$-Streaming Authentication Scheme.*

For any constant $c_{th} > c$, if the adversarial corruption rate exceeds c_{th}, the schemes abort with overwhelming probability over the shared randomness.

Theorem 6.3. *Assume that a bitstream $x_1, x_2, \ldots$ is communicated using some encoding protocol with a constant rate, and assume that at time n the receiver decodes the bitstring $x'_1, \ldots, x'_n$. If the rate of adversarial corruptions is $0 \leq c \leq 1$, then for any constant $\varepsilon > 0$,*

$$\Pr[x'_1 \cdots x'_{(1-c)n+\varepsilon n} = x_1 \cdots x_{(1-c)n+\varepsilon n}] \leq 2^{-\Omega(\varepsilon n)}$$

where the probability is over the coin tosses of the decoding algorithm, assuming $\{x_i\}$ are uniformly, independently distributed.

We now prove Theorem 6.3 and then construct the protocols guaranteed by Theorem 6.2.

Proof. Consider an adversary that, starting at time $(1-c)n$, corrupts all the transmissions. It is easy to verify that the corruption rate is c. Clearly, from time $(1-c)n$ and on, the effective capacity of the channel is 0. This means that the decoder has no use of transmissions of times $\geq (1-c)n$ and he decodes only using transmissions received up to time $(1-c)n$. However, due to the streaming nature of the model, transmissions at times $< (1-c)n$ depend only on $x_1, \ldots, x_{(1-c)n}$ (the suffix of the stream is yet unknown to the sender). The receiver has no information about any bit x_i with $i > (1-c)n$ and his best strategy is to guess them. The probability to correctly guess the last εn bits is at most $2^{-\lfloor \varepsilon n \rfloor}$. □

In order to construct a streaming authentication scheme, we use two concatenated layers of online codes. The inner code is a Blueberry $B : [S+1]^* \to [L+1]^*$ code with constant S, L, and the outer code A is an online code that allows a prefix decoding in the presence of errors and erasures. The entire process can be described by

$$(x_1, \ldots) \xrightarrow{A} (y_1, \ldots) \xrightarrow{B} (z_1, \ldots) \xrightarrow{channel} (\tilde{z}_1, \ldots) \xrightarrow{B^{-1}} (\tilde{y}_1, \ldots) \xrightarrow{A^{-1}} (\tilde{x}_1, \ldots)$$

We begin with a simple and elegant construction which, although not efficient, demonstrates the power of the Blueberry code.

Proposition 6.4. *Let c, ε be constants $0 \leq c < 1$, $0 < \varepsilon \leq (1-c)$ and let $A = \mathsf{TCenc}()$ be an encoding using a binary tree code and B a Blueberry code with constant parameters determined by c, ε. The concatenation of A and B is a $(cn, (1-c)n - \varepsilon n, 2^{-\Omega(n)})$-streaming authentication scheme.*

Proof. Assume that in order to encode the bitstream $x_1, x_2, \ldots$, we use a binary tree code over alphabet $[S+1]$ with distance α to be determined later, concatenated with a Blueberry-code $B : [S+1]^* \to [L+1]^*$. We show that if at time n we decode a string $\tilde{x}_1 \cdots \tilde{x}_n$ whose prefix $\tilde{x}_1 \cdots \tilde{x}_{(1-c-\varepsilon)n}$ differs from $x_1 \cdots x_{(1-c-\varepsilon)n}$, then the corruption rate was larger than c.

For a specific time n, consider a string $\tilde{x} \in \{0,1\}^n$, such that $anc(x, \tilde{x}) \geq (c+\varepsilon)n$. Due to the tree distance property, the Hamming distance between $\mathsf{TCenc}(\tilde{x})$ and $\mathsf{TCenc}(x)$ is at

least $\alpha(c+\varepsilon)n$. Assume Eve causes d erasures and e errors, a maximal-likelihood decoding will prefer x over $\tilde{x}$ as long as $\lfloor \alpha(c+\varepsilon)n \rfloor > 2e + d$.

If Eve's corruption rate is limited to c, Lemma 5.4 implies that with overwhelming probability at most $2cnS/L$ of these corruptions become errors and the rest are marked as erasures. Setting $\alpha > \frac{c}{c+\varepsilon}(1 + \frac{2S}{L})$ we guarantee that with overwhelming probability $\alpha(c+\varepsilon)n > 2 \cdot 2cnS/L + cn(1 - 2S/L)$,[2] thus Bob decodes a string $\tilde{x}$ such that $anc(x, \tilde{x}) < (c+\varepsilon)n$ with overwhelming probability , as claimed.

Note that the actual fraction of adversarial corruptions can be estimated out of the number of erasures marked by the Blueberry code. We abort the decoding if at a specific time n the number of erasures exceeds cn. Lemma 5.4 guarantees that if the adversary corrupts more than a $c/(1 - \frac{2S}{L})$-fraction of the transmissions, she will cause at least cn erasures, except with negligible probability. Choosing L such that $(1 - \frac{2S}{L}) \geq \frac{c}{c_{th}}$ completes the proof for the non-efficient case of Theorem 6.2. □

We note that although in the above proof we require ε to be constant, for the case of $c = 0$ (i.e., when the channel is not inherently noisy) we can let ε be smaller. For instance, if we let $\varepsilon = \kappa/n$ for a security parameter κ, the scheme is comparable to a (non-streaming) authentication scheme with the same security parameter: in order to change even a single bit in a prefix of length n, after $n + \kappa$ symbols were transmitted, the adversary must change at least $\alpha\kappa/2$ transmissions, and will be caught except with probability $2^{-\Omega(\kappa)}$. Since the above holds for any time n, we get a perpetual authentication of the stream.

The case where $c > 0$ has a meaning of communicating over a noisy channel (regardless of the adversary). The users do not abort the authentication scheme although they know the message was changed by the channel. Instead, the scheme features both error-correction and authentication abilities and the parties succeed to recover (a prefix of) the original message with high probability.

[2] It is required to have $\alpha < 1$, thus the choice of (the constant) L should depend on ε and c, specifically, $L > 2S\frac{c}{\varepsilon}$. Also note that S depends on α, however L is independent of both. For a fixed value of α (and $S = d^{O(1/(1-\alpha))}$) there is always a way to choose a constant L that satisfies the conditions.

6.3 Efficient Streaming Authentication

We now complete the proof of Theorem 6.2 by defining an efficient randomized code A_{eff} for streaming prefix-decoding in the presence of errors and erasures (Figure 6.1). The protocol partitions the stream into words of logarithmic size and encodes each using a tree code. At any time n, one of the $O(n/\log n)$ words is chosen at random and its next encoded symbol is transmitted. The value n increases as the protocol progresses which means that the length of each encoded word increases as well. This however causes no problem: each word is encoded by a tree code (rather than, say, a block code), which is performed in an online manner without assuming knowledge of the word's length. Decoding can be performed efficiently by an exhaustive search since each word is of logarithmic length in the current time n. We note that the parties hold the entire stream in their memory throughout the protocol, which is different from the common practice of streaming algorithms in which there is only a single party (rather than two) which aim to compute some statistics of the stream using poly-logarithmic memory.

Proposition 6.5. *For any constants $0 \leq c < 1$, $0 < \varepsilon \leq (1-c)/2$ and a constant $c_1 > 0$, the concatenation of A_{eff} (Figure 6.1) with a Blueberry code $[S+1]^* \to [L+1]^*$ (with suitable constants S, L) satisfies the following condition for any set of infinite strings $\{\mathbf{x}^1, \mathbf{x}^2, \ldots\}$ and for any sufficiently large time n except with polynomially small probability in n.*

If the corruption rate at time n is at most c then the scheme correctly decodes a prefix of length $c_1 \log n$ of each one of the strings $\mathbf{x}^k$ with $k \in \{\lceil \frac{\varepsilon n/4}{\log \varepsilon n/4} \rceil, \ldots, \lceil \frac{(1-c-\varepsilon)n}{\log(1-c-\varepsilon)n} \rceil\}$. Moreover, up to time n the encoding scheme assumes knowledge of only strings $\mathbf{x}^k$ with $k \leq n/\log n$.

Proof. We begin by showing that for any time n, if we look at k's which are not too close to the start or to the end, that is, k's in $K_n \triangleq \{\lceil \frac{\varepsilon n/4}{\log \varepsilon n/4} \rceil, \ldots, \lceil \frac{(1-c-\varepsilon)n}{\log(1-c-\varepsilon)n} \rceil\}$, then every string $\mathbf{x}^k$ is selected by the encoding scheme $\Theta(\log n)$ times in expectation. In addition, a constant fraction of a specific tree's transmissions is received intact, while the expected number of errors is a smaller fraction, controlled by L. Therefore, a logarithmic prefix of the string $\mathbf{x}^k$ can be decoded with high probability.

Let $0 \le c < 1$ and $0 < \varepsilon < (1-c)/2$ be fixed parameters of the protocol. Let c_0, c_1 be some constants which depend on c and ε. Let $\mathcal{T}$ be a tree code over alphabet $[S+1]$ with distance α to be set later.

A_{eff} **Encoding:** For every $k > 0$ set $count_k = 0$.

At any time $n > 1$, repeat the following process for $j = 1, 2, \ldots, c_0$:

(a) randomly choose $k \in \{1, \ldots, \lfloor n/\log n \rfloor\}$.

(b) set $count_k = count_k + 1$.

(c) transmit $y_{n,j} \in [S+1]$, the next symbol of the encoding of $\mathbf{x}^k$ using $\mathcal{T}$, that is, the last symbol of

$$\mathsf{TCenc}(\mathbf{x}_1^k \cdots \mathbf{x}_{count_k}^k) = \mathsf{TCenc}(\mathbf{x}_1^k \cdots \mathbf{x}_{count_k-1}^k) \circ y_{n,j}.$$

A_{eff} **Decoding:** For every $(i,j) \in \mathbb{N} \times [c_0]$ we denote by $\mathsf{ID}(i,j)$ the identifier k of the string $\mathbf{x}^k$ used at iteration (i,j). For each time n, mark all the transmissions $y_{i,j}$ with $i < \varepsilon n/4$ as erasures, and decode $\mathbf{x}^k$ for $\lceil \frac{\varepsilon n/4}{\log \varepsilon n/4} \rceil \le k \le \lceil \frac{(1-c-\varepsilon)n}{\log(1-c-\varepsilon)n} \rceil$:
let $Y_k = \{(i,j) \mid \mathsf{ID}(i,j) = k\}$. Decode the received string indexed by Y_k. That is, set

$$\hat{\mathbf{x}}^k = \mathsf{TCdec}(y_{|_{Y_k}}),$$

where $y_{|_{Y_k}}$ is the string given by concatenating all $y_{i,j}$ with $(i,j) \in Y_k$, where $y_{i,j}$ comes before $y_{i',j'}$ if $i < i'$ or $(i = i') \wedge (j < j')$. Consider a prefix of length $c_1 \log n$ of $\hat{\mathbf{x}}^k$ and ignore the rest.

Figure 6.1: *The efficient coding scheme A_{eff} for communicating a logarithmic prefix of $\{\mathbf{x}^1, \mathbf{x}^2, \ldots, \}$.*

Lemma 6.6. *Let c, ε be given. For a given time n and for every $k \in K_n$,*

1. *the expected number of transmissions (i,j) with $\mathsf{ID}(i,j) = k$ is $\Theta(c_0 \log n)$.*
2. *if the corruption rate is at most c, then for transmissions with $\mathsf{ID}(i,j) = k$, the expected number of transmissions not corrupted by the adversary is $\Theta(c_0 \log n)$ and the expected number of errors is $\Theta(c_0 \log n / L)$.*

Proof. Fix a $k \in K_n$, and recall that $Y_k = \{(i,j) \mid \mathsf{ID}(i,j) = k\}$. It is easy to verify that

$$\Pr[(i,j) \in Y_k] = \begin{cases} 0 & i/\log i < k \\ \frac{1}{\left\lfloor \frac{i}{\log i} \right\rfloor} & i/\log i \geq k \end{cases},$$

where the probability is over the shared randomness R. Assume that the channel's (Eve's) noise pattern is $P = p_1, \ldots, p_{cn}$, with $p_i \in [n] \times [c_0]$. First let us bound the number of erroneous transmissions of symbols from Y_k. For a specific instance of the scheme, let

$$\mathsf{ERR}_k = \left|\left\{(i,j) \in Y_k \mid \tilde{y}^{i,j} \text{ is an error}\right\}\right|.$$

We are interested in the (adversarial) corruption pattern that maximizes the expected number of these errors,

$$\max_P \mathbb{E}\left[\mathsf{ERR}_k\right].$$

Since the decoding process ignores the first $\varepsilon n/4$ transmissions, and since the expected number of errors is a $\frac{S}{L}$-fraction of the corrupted transmissions in Y_k, this equals to

$$\max_P \mathbb{E}\left[\tfrac{S}{L}\left|\left\{(i,j) \;\middle|\; \mathsf{ID}(i,j) = k,\ i > \varepsilon n/4 \ \text{ and } \ \tilde{y}^{i,j} \neq y^{i,j}\right\}\right|\right],$$

where the expectation is over the shared randomness R. Note that $\Pr[(i,j) \in Y_k]$ is monotonically decreasing for $i \geq t(k)$, and zero otherwise. The pattern P that maximizes Eve's probability to hit transmissions in Y_k is $P_\circ \triangleq \{t(k), t(k)+1, \ldots, t(k)+cn\}$. However, the decoding algorithm ignores the first $\varepsilon n/4$ indices and Eve has no use in attacking them. Therefore, if $P_\circ \cap [\varepsilon n/4] \neq \emptyset$, Eve's best strategy is to shift her attack to the window $P = \{\lfloor \varepsilon n/4 \rfloor, \lfloor \varepsilon n/4 \rfloor + 1, \ldots, \lceil (\varepsilon/4 + c)n \rceil\}$.

$$\begin{aligned}
\max_P \mathbb{E}\left[\mathsf{ERR}_k\right] &= \max_P \frac{S}{L} \sum_{(i,j)\in P} \Pr[(i,j) \in Y_k] \\
&\leq \frac{c_0 S}{L} \sum_{i=t(k)}^{cn+t(k)} \frac{1}{\frac{i}{\log i} - 1} \\
&\leq \frac{c_0 S \log n}{L} \sum_{i=\varepsilon n/4}^{n} \frac{1}{i - \log i} \\
&\leq \frac{c_0 S \log n}{L} \sum_{i=\varepsilon n/4}^{n} \frac{1}{c'' i} = \frac{c_0 S \log n}{c'' L}(H_n - H_{\varepsilon n/4})
\end{aligned}$$

where the second inequality applies to both the cases of empty and non-empty $P_\circ \cap [\varepsilon n/4]$, and $c'' < 1$ is some constant such that $c''i \leq i - \log i$ for $i \geq \varepsilon n/4$, for a sufficiently large n. H_n is the n-th Harmonic number, and it holds that $0 < H_n - \ln(n) < 1$. We get $\max_P \mathbb{E}\left[\mathsf{ERR}_k\right] = O(c_0 \log n/L)$.

On the other hand, we can lower bound the amount of uncorrupt transmissions in Y_k. In a similar way to the above we define $\mathsf{INTACT}_k = \left|\{(i,j) \in Y_k| \;\; \tilde{y}^{i,j} = y^{i,j}\}\right|$, and wish to lower bound the quantity $\min_P \mathbb{E}\left[\mathsf{INTACT}_k\right]$. It is easy to verify that Eve's strategy from above is optimal for this case as well, thus

$$\begin{aligned}\min_P \mathbb{E}\left[\mathsf{INTACT}_k\right] \geq \min_P \sum_{\substack{(i,j)\notin P \\ i>\varepsilon n/4}} \Pr[(i,j) \in Y_k] \geq \sum_{i=(c+\varepsilon/4)n+t(k)}^{n} \frac{c_0 \log i}{i} \\ \geq c_0\big((1-c-\varepsilon/4)n - t(k)\big)\frac{\log((c+\varepsilon/4)n + t(k))}{n},\end{aligned}$$

which is $\Omega(c_0 \log n)$ for $t(k) \leq (1-c-\varepsilon)n$, hence the claim holds for

$$k \leq \frac{(1-c-\varepsilon)n}{\log(1-c-\varepsilon)n}.$$

Finally, define $\mathsf{TOTAL}_k = |Y_k|$ to be the total amount of transmissions with $\mathsf{ID}(i,j) = k$ (erasures, errors, and intact). The expected amount of this quantity is at least

$$\mathbb{E}[\mathsf{TOTAL}_k] \leq c_0 \sum_{i=\varepsilon n/4}^{n} \frac{\log i}{i - \log i} \leq c_0 \sum_{\varepsilon n/4}^{n} \frac{\log i}{c''i} \leq \frac{c_0 \log n}{c''}(H_n - H_{\varepsilon n/4}) = O(c_0 \log n)$$

with some small constant $c'' < 1$ for a sufficiently large n. The sum begins from $\varepsilon n/4$ since $\mathbf{x}^k$ is declared only at time $t(k) \geq \varepsilon n/4$, if $k \geq \varepsilon n/4 \log(\varepsilon n/4)$. Since the number of intact transmissions is $\Omega(c_0 \log n)$, the total amount of transmissions is lower-bounded by the same quantity, thus $\mathbb{E}[\mathsf{TOTAL}_k] = \Theta(c_0 \log n)$. □

At time n, assume $\max_{k\in K_n} \mathbb{E}[\mathsf{TOTAL}_k] < C_T \log n$ and $\min_{k\in K_n} \mathbb{E}[\mathsf{INTACT}_k] > C_I \log n$, and define $\beta = C_I/C_T$. Note that β is independent of n and c_1. Fix $k \in K_n$. Denote by $\mathcal{BAD}_1$ the event that there were too many erasures and errors for the k-th codeword, i.e. $\mathsf{INTACT}_k/\mathsf{TOTAL}_k < \beta/2$, and by $\mathcal{BAD}_2$ the event that there were not enough transmissions for the k-th codeword, $\mathsf{TOTAL}_k < \frac{2c_1}{\beta}\log n$.

By an appropriate choice of $c_0 = O(c, \varepsilon, c_1, 1/\beta)$, we can bound the probability of any bad event to be polynomially small. For large enough c_0 we can assure that $\mathbb{E}[\mathsf{TOTAL}_k] > \frac{4c_1}{\beta} \log n$, and thus by Chernoff, $\Pr[\mathcal{BAD}_2] \leq 2^{-\Omega(\log n)}$. Furthermore, a union bound gives

$$\Pr[\mathcal{BAD}_1] < \Pr\left[\mathsf{TOTAL}_k > \sqrt{2} C_T \log n\right] + \Pr\left[\mathsf{INTACT}_k < \frac{1}{\sqrt{2}} C_I \log n\right],$$

and by Chernoff inequality,

$$\Pr[\mathcal{BAD}_1] \leq 2^{-\Omega(\mathbb{E}[\mathsf{TOTAL}_k])} + 2^{-\Omega(\mathbb{E}[\mathsf{INTACT}_k])} = 2^{-\Omega(\log n)}.$$

Conditioned on the fact that $\mathcal{BAD}_1$ and $\mathcal{BAD}_2$ did not occur, we know that $n^* > \frac{2c_1}{\beta} \log n$ symbols of $\mathsf{TCenc}(\mathbf{x}^k)$ were transmitted, and the adversary has corrupted at most $c^* < (1 - \beta/2)$ fraction of these transmissions. Proposition 6.4 suggests that for an appropriate choice of constant L, we are able to decode a prefix of length at least $\approx (1 - c^*)n^* = c_1 \log n$ except with probability $\exp(-\Omega(n^*)) = \exp(-\Omega(\log n))$. A union bound over all the possible $k \in K_n$ completes the proof. □

We note that since each codeword is of length $O(\log n)$, decoding via exhaustive search can be performed with polynomial computational effort. Hence, it is easy to verify that both the encoding and decoding can be done efficiently. We also emphasize that each encoded symbol is of constant size, thus the scheme has constant rate.

In order to complete the proof of Theorem 6.2, we are left to explain how to split the stream $x_1, x_2, \ldots$ into words $\{\mathbf{x}^1, \mathbf{x}^2, \ldots\}$ such that for any time n, the entire prefix $x_1, \ldots, x_{(1-c-\varepsilon)n}$ appears in the $c_1 \log n$-prefixes of strings $\{\mathbf{x}^k\}$ with $k \in K_n$.

For every k, define $\mathbf{x}^k$ to be the string that contains the stream prefix $x_{t(k)}$ downto x_1 concatenated with as many zeros as needed, $\mathbf{x}^k = x_{t(k)} x_{t(k)-1} \cdots x_2 x_1 000 \cdots$, where $t(k)$ is defined to be the minimal time such that $t(k)/\log t(k) > k$. We say that $\mathbf{x}^k$ is *declared* at time $t(k)$, meaning that only from this time and on the algorithm may choose to send symbols of the encoding of $\mathbf{x}^k$. It is easy to verify that the string $\mathbf{x}^k$ is well defined at the time it is declared (the corresponding x_i's are known).

If some string $\mathbf{x}^k$ is declared at time $t(k)$ then the string $\mathbf{x}^{k+1}$ will be declared at time $t(k+1) \approx t(k) + \log t(k) + O(\log\log t(k))$. By setting $c_1 = 2$ we are guaranteed that, for every $\varepsilon n/4 \le \ell \le (1-c-\varepsilon)n$, x_ℓ appears in a correctly decoded $c_1 \log n$-prefix of some $\mathbf{x}^k$ with $k \in K_n$.

Lemma 6.7. *If $\mathbf{x}^k$ is the latest string declared at time $i > 8$, then $\mathbf{x}^{k+1}$ is declared at time sooner than $i + 2\log i$.*

Proof. Let $f(i) = \frac{i+2\log i}{\log(i+2\log i)} - \frac{i}{\log i}$. f is monotonically increasing, and $f(8) > 1$. □

Corollary 6.8. *For any time $n > 8$, and any ℓ, the bit x_ℓ is within the first $2\log n$ symbols of $\mathbf{x}^{\lceil \ell/\log \ell \rceil}$. Hence, every x_ℓ with $\varepsilon n/4 \le \ell \le (1-c-\varepsilon)n$, appears in a $2\log n$-prefix of (at least) one of the strings $\{\mathbf{x}^k\}_{k\in K_n}$.*

Unfortunately, with the above choice of $\mathbf{x}^k$s, only part of the stream, $x_{\varepsilon n/4}, \ldots, x_{(1-c-\varepsilon)n}$, is decoded by the protocol. In order to communicate the prefix $x_1, \ldots, x_{\varepsilon n/4}$ we run another instance of the scheme guaranteed by Proposition 6.5 for the following set of infinite strings $\{\mathbf{v}^1, \mathbf{v}^2, \ldots\}$. (We explain how to combine these two instances below). Define $\mathbf{v}^k$ in the following way

$$\mathbf{v}_i^k = \begin{cases} x_1 & k = 1,\ \forall i \\ x_{1+(\ell \bmod \lceil t(k)/2 \rceil + 1)} & k > 1,\ i = 1 \text{ and } \mathbf{v}^{k-1}_{2\log t(k-1)} = x_\ell \\ x_{1+(\ell \bmod \lceil t(k)/2 \rceil + 1)} & k > 1,\ i > 1 \text{ and } \mathbf{v}^k_{i-1} = x_\ell \end{cases}$$

It is easy to verify that at time n, the string $\mathbf{v}^{\lfloor n/\log n \rfloor}$ is well defined and known to the encoder.

Lemma 6.9. *For every time $n > 256/(1-c-\varepsilon)$, any bit x_ℓ with $1 \le \ell \le \varepsilon n/4$ appears in a $2\log n$-prefix of (at least) one of the strings $\{\mathbf{v}^k\}_{k\in K_n}$.*

Proof. Note that the concatenation of $O(\log n)$-prefix of the $\mathbf{v}^k$s gives a string of the form $V \triangleq x_1 x_2 \ldots x_{\lceil t(k_1)/2 \rceil} x_1 x_2 \ldots x_{\lceil t(k_2)/2 \rceil} x_1 x_2 \ldots$, and V is decoded by A_{eff} with high probability.[3]

[3] To be more accurate, V is a substring of the string decoded by the scheme.

By taking $c_1 = 2$ and recalling that $\varepsilon < (1-c)/2$, (and thus, $(1-c-\varepsilon)n/4 > \varepsilon n/4$) the length of V is lower bounded by the amount of indices in prefixes of size $2\log\frac{1}{4}(1-c-\varepsilon)n$ of $\{\mathbf{v}^{(1-c-\varepsilon)n/4}, \ldots, \mathbf{v}^{(1-c-\varepsilon)n}\}$,

$$\begin{aligned} 2\log\frac{1}{4}(1-c-\varepsilon)n &\left(\frac{(1-c-\varepsilon)n}{\log(1-c-\varepsilon)n} - \frac{\frac{1}{4}(1-c-\varepsilon)n}{\log\frac{1}{4}(1-c-\varepsilon)n}\right) \\ &\geq \frac{3}{2}(1-c-\varepsilon)n - 4\frac{(1-c-\varepsilon)n}{\log(1-c-\varepsilon)n} \\ &\geq (1-c-\varepsilon)n \end{aligned}$$

where the last inequality holds for $n > \frac{256}{1-c-\varepsilon}$. Consider the latest place in V where x_1 appears. If that place is at least $(1-c-\varepsilon)/4$ indices from the end of V, it is clear that $x_1 \ldots x_{(1-c-\varepsilon)/4}$ appears in the $(1-c-\varepsilon)/4$-suffix of the decoded V. For the other case, let the bit that precedes this x_1 be x_ℓ. By the way we defined $\{\mathbf{v}^k\}$ it follows that $\frac{3}{8}(1-c-\varepsilon) \leq \ell \leq \frac{1}{2}(1-c-\varepsilon)$ which means that $x_1 \ldots x_{(1-c-\varepsilon)/4}$ must appear in a prefix of size $3/4 \cdot (1-c-\varepsilon)n$ of V. Since $(1-c-\varepsilon)n/4 > \varepsilon n/4$, the claim holds. □

One cannot run A_{eff} twice, once for $\{\mathbf{x}\}$ and separately for $\{\mathbf{v}\}$. Indeed, Eve can block all the transmissions of one of the instances, thus prevent the correct decoding of the stream with probability one, while her corruption rate does not exceed $c = 1/2$. One possible solution is to set $c_1 = 4$ and interleave the transmitted data, that is, define the set $\{\mathbf{z}^1, \mathbf{z}^2, \ldots\}$ where $\mathbf{z}^k = \mathbf{x}_1^k \mathbf{v}_1^k \mathbf{x}_2^k \mathbf{v}_2^k \ldots$, etc.

Corollary 6.10. *Let c, ε be constants $0 \leq c < 1$, $0 < \varepsilon \leq (1-c)/2$, and let B be a Blueberry code with constant parameters determined by c, ε. For the strings $\{\mathbf{z}^1, \mathbf{z}^2, \ldots\}$ defined above, the concatenation of A_{eff} with B is an efficient $(cn, (1-c)n - \varepsilon n, 2^{-\Omega(\log n)})$-streaming authentication scheme.*

6.4 Streaming Authentication: Extensions

6.4.1 Decoding a Prefix Longer than $(1-c)n$

Although our scheme decodes a prefix of length $(1-c)n$ in the worst case, the successfully decoded prefix can in fact be longer. The worst case, as demonstrated by Theorem 6.3,

happens when the adversary blocks the suffix of the transmitted stream. On the other hand, if the adversary blocks the prefix of the transmissions, then the scheme of Proposition 6.4 correctly decodes the entire stream! We show the following,

Theorem 6.11. *At any given time n in which the noise rate does not exceed c, there exists a time $(1-c-\varepsilon)n < t < n$, such that for all $1 \leq m \leq t$,*

$$\Pr[x'_m \neq x_m] \leq 2^{-\Omega(|t-m|)}.$$

Furthermore, Bob can output estimators t_{min} and t_{max} such that $\Pr[t_{min} > t] < 2^{-\Omega(t_{min}-t)}$ and $\Pr[t_{max} < t] < 2^{-\Omega(t-t_{max})}$.

Proof. We begin by showing that any time t that satisfies the following *suffix condition*, is a times in which Bob will be able to decode the prefix of the stream up to almost t. Later (in Proposition 6.14) we show that Bob can get a very close estimation of the latest time that satisfies the suffix condition.

Definition 6.2. *For any constant $0 \leq \gamma < 1$, we say that time n satisfies the γ-suffix condition if any suffix $x_t \dots x_n$ has at most $\gamma(n-t)$ corrupted transmissions.*

Definition 6.3. *Let $c < 1$ and $\gamma \in (c, 1)$ be given. For any time n let $N_\gamma(n)$ be the latest index that satisfies the γ-suffix condition. When n is clear from the context, we denote $N_\gamma(n)$ simply as N_γ.*

The following Lemma guarantees that for any $\gamma \in (c, 1)$ it holds that $(1 - c/\gamma)n \leq N_\gamma(n) \leq n$.

Lemma 6.12. *For every corruption rate c and constant $1 < \xi < 1/c$ there exist a time $t > (1 - \frac{1}{\xi})n$ that satisfies the $c\xi$-suffix condition.*

Proof. Look at a suffix $y_{t+1}, \dots, y_n$ for which the number of corruptions is strictly larger than $c\xi(n-t)$. If no such suffix exists then the lemma is true for $y_1, \dots, y_n$. Otherwise, discard $y_{t+1}, \dots, y_n$, and repeat the process with $y_1, \dots, y_t$. At each iteration we remove more than $c\xi(n-t)$ corrupted transmissions and shorten the string by $n-t$ symbols. Assume that

the process stops with some prefix of length $L < (1 - \frac{1}{\xi})n$, then we have removed at least $c\xi(n - L) > c\xi(n - n + n/\xi) > cn$ corruptions which is a contradiction. Therefore, the entire process must stop with some prefix of length at least $(1 - \frac{1}{\xi})n$. □

Let the corruption rate c and $\varepsilon > 0$ be fixed, and set $\gamma = c/(c + \varepsilon)$. Then, for any time n that satisfies the γ-suffix condition, Bob will correctly decode almost all the way up to n. That is, the probability that a bit that is located t indices away from n was not correctly decoded decreases exponentially in $n - t$. Formally,

Lemma 6.13. *Let c and ϵ be given and let $\gamma = c/(c + \epsilon)$. Assume that time n satisfies the γ-suffix condition, and let $x'_1, \ldots, x'_n$ be the string Bob outputs at time n in the protocol of Proposition 6.4. Then, for any $t < n$,*

$$\Pr[x'_t \neq x_t] < 2^{-\Omega(n-t)}.$$

Proof. Assume towards contradiction that Bob recovers the stream correctly only until time $t < n$. It follows that the errors and erasures in the $[t, n]$ suffix (e and d respectively,) satisfy $2e + d \geq \alpha(n - t)$. However, we know that n satisfies the γ-suffix condition so $e + d < \gamma(n - t)$. Except with probability $2^{-\Omega(n-t)}$ it holds that $2e + d < (1 + 2\frac{S}{L}) \cdot \gamma(n - t)$, thus except with the same probability, $\alpha < (1 + 2\frac{S}{L})\frac{c}{c+\epsilon}$ which is a contradiction to the way we choose α in the protocol described in Proposition 6.4. □

Next, we show how Bob can estimate the value of N_γ by checking the number of erasures marked by the Blueberry code.

Proposition 6.14. *Bob can efficiently compute a (lower-bound) estimation N'_γ for N_γ, such that $N'_\gamma > (1 - c - \epsilon)n$ and*

$$\Pr[N'_\gamma > N_\gamma] < 2^{-\Omega(N'_\gamma - N_\gamma)}.$$

Proof. Consider the following procedure for estimating N_γ.
Let $c \in [0, 1)$ be given. For an input $\gamma \in (c, 1)$, at time n, Bob tries to find the longest suffix that satisfies the γ-suffix condition. To this end Bob performs the following.

1. Set $i = n$.

2. Check all the suffixes $x_t, \ldots, x_i$, with $t < i$.

   (a) If, in all such suffixes, the number of erasures is less than $\gamma(i-t)(1-2\frac{S}{L})$, output $N'_\gamma = i$.

   (b) Otherwise set $i \leftarrow i-1$ and repeat. If $i < (1-c/\gamma)n$ break and output $N'_\gamma = \lfloor (1-c/\gamma)n \rfloor$.

Assume that Bob outputs $N'_\gamma > N_\gamma$. Since N_γ is the latest index that satisfies the γ-suffix condition, there must exist some time t such that $[t, N'_\gamma]$ has more than $\gamma(N'-t)$ corruptions, yet the number of erasures in that interval is less than $\gamma(N'_\gamma - t)(1-2\frac{S}{L})$. This happens with probability exponentially small in $N'_\gamma - t$. If $t < N_\gamma$ then $N'_\gamma - t > N'_\gamma - N_\gamma$ which proves the claim for this case.

For the case where $t > N_\gamma$, we note that time t does not satisfy the γ-suffix condition, therefore there must exist some time $t_1 < t$, for which the number of corruptions in the interval $[t_1, t]$ is more than $\gamma(t-t_1)$. If $t_1 > N_\gamma$, then there must exist time t_2 and interval $[t_2, t_1]$ that doesn't satisfy the γ-suffix condition. We repeat this reasoning until we find the first interval $[t_j, t_{j-1}]$ such that $t_j < N$. By considering the union of all these intervals, it follows that the number of corruptions in $[t_j, N'_\gamma]$ is more than $\gamma(N'_\gamma - t_j) > \gamma(N'_\gamma - N_\gamma)$. However, Bob's estimation process found at most $\gamma(N'_\gamma - N_\gamma)(1-2\frac{S}{L})$ erasures when it examined the suffix interval $[t_j, N'_\gamma]$ (otherwise, it would have failed the check in Step 2a). According to Lemma 5.4 and Corollary 5.5, this happens with probability exponentially small in $(N'_\gamma - N_\gamma)$. □

We note that Bob can repeat the same procedure and compute a value N''_γ which usually upper-bounds N_γ, by finding the latest time i for which every suffix $[t, i]$ has less than $\gamma(i-t)(1-\frac{1}{2}\frac{S}{L})$. As above, the probability of the bad event that $N''_\gamma < N_\gamma$ is exponentially small in $(N_\gamma - N''_\gamma)$. This completes the proof of Theorem 6.11. □

6.4.2 Efficient Authentication Scheme with Exponentially Small Error

It is possible to improve the efficient scheme of Theorem 6.2 so that it aborts with polynomially small probability, however, given that it did not abort, the probability that the decoded prefix is incorrect is *exponentially small*. More accurately, the 'trust' Bob has in the decoded string *increases* with the amount of received transmissions. Thus, except for the last fraction of the stream, the decoded stream is identical to the one sent by Alice with overwhelming probability.

Theorem 6.15. *For any $0 \leq c < 1$, $0 < \varepsilon \leq \frac{1}{2}(1-c)$ there exists an efficient $(cn, (1-c)n - \varepsilon n, 2^{-\Omega(\log n)})$-streaming authentication protocol that, for any time n in which the decoding procedure did not abort, for any $1 \leq \ell \leq (1-c-\varepsilon)n$ it holds that*

$$\Pr\left[x'_\ell \neq x_\ell\right] < 2^{-\Omega(n)}.$$

To this end, we add a parallel transmission of random hash values of the entire stream (up to time n), where the hash length is logarithmic in n.[4] More formally, define an additional set of infinite strings $\{\mathbf{h}^1, \mathbf{h}^2, \dots\}$. We identify a string $a = a_1 a_2 \cdots a_n$ with the n-dimensional vector $(a_1, a_2, \dots, a_n)$, and define $\mathbf{h}^k$ in the following way. Randomly pick a matrix $R_k \in \{0,1\}^{\log t(k) \times t(k)}$ and a vector $V_k \in \{0,1\}^{\log t(k)}$ and set $\mathbf{h}^k = R_k \cdot (x_1, x_2, \dots, x_{t(k)})^T + V_k$, concatenated with as many zeroes as needed. The strings $\{\mathbf{h}^k\}$ are interleaved with the strings $\{\mathbf{x}^k\}$ and $\{\mathbf{v}^k\}$, and c_1 increases as explained in Section 6.3. We call the resulting scheme A_{eff} *with hash testing.*

Proposition 6.5 guarantees that except with polynomially small probability in n all the hash values $\{\mathbf{h}^k\}_{k \in K_n}$ are correctly decoded. The decoding at time n aborts if any of the hash values $\{\mathbf{h}^k\}_{k \in K_n}$ mismatches the corresponding prefix $x_1 \cdots x_{t(k)}$.

Proposition 6.16. *Given that A_{eff} with hash testing did not abort, let x' be the decoded stream, then for every $\ell \leq (1-c-\varepsilon)n$,*

$$\Pr_R\left[x'_\ell \neq x_\ell\right] < 2^{-\Omega(\max\{(1-c-\varepsilon)n-\ell\ ,\ \log n\})}.$$

[4] This method is similar to the classic authentication method of splitting a stream into chunks of size $\log n$ and adding a MAC of logarithmic size after each chunk.

Proof. Eve is oblivious of R_i and V_i, thus for *any* two vectors $\tilde{\mathbf{x}} \in \{0,1\}^{t(i)}$, $\tilde{\mathbf{h}} \in \{0,1\}^{\log t(i)}$ chosen by Eve, $\Pr_{R_i}[\tilde{\mathbf{h}} = R_i \cdot \tilde{\mathbf{x}}^T + V_i] < 2^{-\log t(i)}$.

Clearly, the smaller ℓ is, the more hash values that are checked to be consistent with the decoded x'_ℓ. For $\ell > \varepsilon n/4$ there are at least $((1-c-\varepsilon)n-\ell)/2\log n$ independent hash values of stream prefixes longer than ℓ, where the smallest hash length is $\approx \log(\varepsilon n/4)$. Hence, the probability that $x'_\ell \neq x_\ell$ yet the decoding procedure did not abort is at most

$$2^{-\Omega\left(\log(\varepsilon n/4)\frac{(1-c-\varepsilon)n-\ell}{2\log n}\right)} = 2^{-\Omega((1-c-\varepsilon)n-\ell))}.$$

Clearly, for $\ell < \varepsilon n/4$ there are as many hash tests as for $\ell = \varepsilon n/4$, thus the probability to incorrectly decode x_ℓ with $\ell < \varepsilon n/4$ is exponentially small in n as well. Finally, for the case where $((1-c-\varepsilon)n-\ell) < \log n$ we note that at least one hash value must be consistent, $\mathbf{h}^{\lceil (1-c-\varepsilon)n/\log(1-c-\varepsilon)n\rceil}$. The probability to incorrectly decode x_ℓ and pass the hash check is at most $2^{-\Omega(\log n)}$, which completes the proof. □

The proof of Theorem 6.15 is immediate from the above Proposition.

Proof. (**Theorem 6.15**) Let c, ε be fixed. Perform A_{eff} with hash testing with parameters $c, \varepsilon' = \varepsilon/2$. By Proposition 6.16, every decoded x'_ℓ with $\ell \leq (1-c-\varepsilon)n$ satisfies

$$\Pr[x'_\ell \neq x_\ell] < 2^{-\Omega((1-c-\varepsilon')n-\ell)} \leq 2^{-\Omega(\varepsilon n/2)}.$$

□

APPENDIX A

The Braverman-Rao Emulation Scheme [BR11]

For self-containment, we give here the detailed emulation process of Braverman and Rao. This section is taken almost as is from [BR11].

Any communication protocol of T rounds where the parties alternately exchange bits can be be reduced to the following *jumping pointer problem*. Let $\mathcal{T}$ be a binary tree of depth T, Let $\mathcal{X}$ denote the set of edges leaving vertices at even depth, and $\mathcal{Y}$ denote the set of edges leaving vertices at odd depth. Given any set A of edges in the tree, we say that A is *consistent* if A contains at most one edge leaving every vertex of the tree. We write $v(A)$ to denote the unique vertex of maximal depth that is reachable from the root using the edges of A. Let $X \subset \mathcal{X}$, $Y \subset \mathcal{Y}$ be consistent subsets, and observe that $X \cup Y$ is also consistent. Now, any specific protocol π can be seen as giving the parties such sets X_π, Y_π, and the goal is compute $v(X_\pi \cup Y_\pi)$.

The emulation protocol is defined over the alphabet

$$\Sigma = \{0, 1, \ldots, R\} \times \{0,1\}^{\leq 2},$$

encoded using a tree code. Let $\mathsf{TCenc} : \Sigma^{\leq R} \to \Gamma^{\leq R}$ and $\mathsf{TCdec} : \Gamma^{\leq R} \to \Sigma^{\leq R}$ be the encoding and decoding functions of a $|\Sigma|$-ary tree code of distance $\alpha = 1 - \epsilon$.

Given any $\sigma \in \Sigma^k$, σ represents a set of at most k edges in the binary tree $\mathcal{T}$, computed as follows. Let $\sigma_i = (r_i, s_i)$, where $r_i \in \{0, 1, \ldots, R\}$ and $s_i \in \{0,1\}^{\leq 2}$. We think of r_i as a pointer to the r_i'th edge determined by $\sigma_1, \ldots, \sigma_{r_i}$, and of s_i as specifying an edge that is a descendant of the r_i'th edge.

Formally, for $i = 1, \ldots, k$, we use the following procedure to determine an edge e_i from σ_i:

1. If $r_i = 0$, set e'_i to be the edge of depth at most 2 that is specified by the bits s_i.

2. If $r_i < i$ and e_{r_i} has been defined, let e'_i be the edge specified by starting at the child vertex of the edge e_{r_i} and then taking (at most) two steps in the tree using the bits s_i.

3. If e'_i has been assigned an edge, and there there is no $j < i$ for which e_j, e'_i share a common parent, set $e_i = e'_i$. Else leave e_i undefined.

Given such a string σ, we write $E(\sigma)$ to denote the set of edges that have been assigned to e_i for some i in the process above. Observe that for every edge in $E(\sigma)$, there is a unique index i to which the edge has been assigned to e_i. Further, $E(\sigma)$ is consistent and $E(\sigma_1, \ldots, \sigma_i) \subseteq E(\sigma_1, \ldots, \sigma_{i+1})$.

Our protocol will proceed in rounds. In the i'th round, the first party (resp. second party) computes a symbol σ_i that extends longest consistent path in $\mathcal{T}$. The party then transmits a_i, the last symbol of $\mathsf{TCenc}(\sigma)$; the second party receives (a possibly corrupt) $\tilde{a}_i$ (resp., the second party computes ρ_i and transmits b_i, the last symbol of $\mathsf{TCenc}(\rho)$, which becomes $\tilde{b}_i$ after the noisy channel). The transmissions will ensure that $E(\sigma_1, \ldots, \sigma_{i-1}) \subseteq X$ and $E(\rho_1, \ldots, \rho_{i-1}) \subseteq Y$. Define

$$A_i = \begin{cases} \emptyset & \text{if } i = 1, \\ E(\sigma_1, \ldots, \sigma_{i-1}) \cup (E(\mathsf{TCdec}(\tilde{b}_1, \ldots, \tilde{b}_{i-1})) \cap \mathcal{Y}) & \text{else.} \end{cases}$$

and

$$B_i = \begin{cases} \emptyset & \text{if } i = 1, \\ E(\rho_1, \ldots, \rho_{i-1}) \cup (E(\mathsf{TCdec}(\tilde{a}_1, \ldots, \tilde{a}_{i-1})) \cap \mathcal{X}) & \text{else.} \end{cases}$$

The Emulation Protocol.

1. For $i = 1, \ldots, R$,

   First Party: If $v(A_i) \neq v(A_i \cup X)$, set σ_i so that $E(\sigma_1, \ldots, \sigma_i)$ contains the edge out of $v(A_i)$ from the set X. Else, set $\sigma_i = (R, 00)$, so that σ_i cannot represent any edge. Transmit a_i, the last symbol of $\mathsf{TCenc}(\sigma_1, \ldots, \sigma_i)$.

Second Party: If $v(B_i) \neq v(B_i \cup Y)$ set ρ_i so that $E(\rho_1, \ldots, \rho_i)$ contains the edge out of $v(B_i)$ from the set Y. Else, set $\rho_i = (R, 00)$, so that ρ_i cannot represent any edge. Transmit b_i, the last symbol of $\mathsf{TCenc}(\rho_1, \ldots, \rho_i)$.

2. The outcome of the protocol from the first (resp. second) party's point of view is the vertex $v(A_{R+1})$ (resp. $v(B_{R+1})$).

Note that the protocol is well defined since if $v(A_i) \neq v(A_i \cup X)$, then $v(A_i)$ must be at even depth, and $E(\sigma_1, \ldots, \sigma_{i-1})$ must contain an edge touching a grandparent of $v(A_i)$ that was transmitted by the first party. Thus there is always a symbol σ_i that can extend $\sigma_1, \ldots, \sigma_{i-i}$ to encode the next edge on the path to $v(A_i)$.

References

[AGS13] S. Agrawal, R. Gelles, and A. Sahai. Adaptive protocols for interactive communication. Manuscript, arXiv:1312.4182 (cs.DS), 2013.

[AGHP92] N. Alon, O. Goldreich, J. Håstad, and R. Peralta. Simple constructions of almost k-wise independent random variables. *Random Structures & Algorithms*, 3(3):289–304, 1992.

[BKTA13] A. Badr, A. Khisti, W.-T. Tan, and J. Apostolopoulos. Streaming codes for channels with burst and isolated erasures. *INFOCOM, 2013 Proceedings IEEE*, pp. 2850–2858. 2013.

[Bon63] C. Bonsall. *The case of the hungry stranger*. HarperCollins, 1963.

[BK12] Z. Brakerski and Y. T. Kalai. Efficient interactive coding against adversarial noise. *Proceedings of the 53rd IEEE Annual Symposium on Foundations of Computer Science*, FOCS '12, pp. 160–166. IEEE Computer Society, 2012.

[BN13] Z. Brakerski and M. Naor. Fast algorithms for interactive coding. *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '13, pp. 443–456. 2013.

[BTT13] G. Brassard, A. Tapp, and D. Touchette. Noisy interactive quantum communication. Manuscript, arXiv:1309.2643 (quant-ph), 2013.

[Bra12] M. Braverman. Towards deterministic tree code constructions. *ITCS '12: Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pp. 161–167. ACM, 2012.

[BE14] M. Braverman and K. Efremenko. List and unique coding for interactive communication in the presence of adversarial noise. *Electronic Colloquium on Computational Complexity (ECCC)*, 2014. http://eccc.hpi-web.de/report/2014/007.

[BR11] M. Braverman and A. Rao. Towards coding for maximum errors in interactive communication. *STOC '11: Proceedings of the 43rd annual ACM symposium on Theory of Computing*, pp. 159–166. ACM, New York, NY, USA, 2011.

[CPT13] K.-M. Chung, R. Pass, and S. Telang. Knowledge-preserving interactive coding. *Proceedings of the 54th annual IEEE Symposium on Foundations of Computer Science*, FOCS '13, pp. 449–458. 2013.

[EGM90] S. Even, O. Goldreich, and S. Micali. On-line/off-line digital signatures. G. Brassard, ed., *Advances in Cryptology – CRYPTO '89 Proceedings*, *Lecture Notes in Computer Science*, vol. 435, pp. 263–275. Springer Berlin / Heidelberg, 1990.

[Fan63] R. M. Fano. A heuristic discussion of probabilistic decoding. *IEEE Transactions on Information Theory*, 9(2):64–74, 1963.

[FGOS13] M. Franklin, R. Gelles, R. Ostrovsky, and L. J. Schulman. Optimal coding for streaming authentication and interactive communication. R. Canetti and J. A. Garay, eds., *Advances in Cryptology – CRYPTO 2013*, *Lecture Notes in Computer Science*, vol. 8043, pp. 258–276. Springer Berlin, 2013.

[Gal68] R. G. Gallager. *Information theory and reliable communication*. John Wiley & Sons, 1968. ISBN W-471-29048-3.

[GMS11] R. Gelles, A. Moitra, and A. Sahai. Efficient and explicit coding for interactive communication. *Proceedings of the 52nd IEEE Annual Symposium on Foundations of Computer Science*, FOCS '11, pp. 768–777. 2011.

[GMS14] R. Gelles, A. Moitra, and A. Sahai. Efficient coding for interactive communication. *IEEE Transactions on Information Theory*, 60(3):1899–1913, 2014.

[GOR14] R. Gelles, R. Ostrovsky, and A. Roytman. Efficient error-correcting codes for sliding windows. V. Geffert, B. Preneel, B. Rovan, J. Štuller, and A. Tjoa, eds., *SOFSEM 2014: Theory and Practice of Computer Science*, *Lecture Notes in Computer Science*, vol. 8327, pp. 258–268. Springer International Publishing, 2014.

[GSW14] R. Gelles, A. Sahai, and A. Wadia. Private interactive communication across an adversarial channel. *Proceedings of the 5th Conference on Innovations in Theoretical Computer Science*, ITCS '14, pp. 135–144. ACM, 2014.

[GR97] R. Gennaro and P. Rohatgi. How to sign digital streams. B. Kaliski, ed., *Advances in Cryptology – CRYPTO '97*, *Lecture Notes in Computer Science*, vol. 1294, pp. 180–197. Springer Berlin / Heidelberg, 1997.

[GH13] M. Ghaffari and B. Haeupler. Optimal error rates for interactive coding II: Efficiency and list decoding. Manuscript, arXiv:1312.1763 (cs.DS), 2013.

[GHS13] M. Ghaffari, B. Haeupler, and M. Sudan. Optimal error rates for interactive coding I: Adaptivity and other settings. Manuscript, arXiv:1312.1764 (cs.DS), 2013.

[Gol04] O. Goldreich. *Foundations of cryptography. Vol II: Basic applications*. Cambridge University Press, New York, 2004. ISBN 0521830842.

[GM01] P. Golle and N. Modadugu. Authenticating streamed data in the presence of random packet loss. *ISOC Network and Distributed System Security Symposium, NDSS'01*. 2001.

[Hae14] B. Haeupler, 2014. Personal communication.

[KR13] G. Kol and R. Raz. Interactive channel capacity. *STOC '13: Proceedings of the 45th annual ACM symposium on Symposium on theory of computing*, pp. 715–724. ACM, 2013.

[Lan04] M. Langberg. Private codes or succinct random codes that are (almost) perfect. *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '04, pp. 325–334. IEEE Computer Society, 2004.

[LH12] D. Leong and T. Ho. Erasure coding for real-time streaming. *Information Theory Proceedings (ISIT), 2012 IEEE International Symposium on*, pp. 289–293. 2012.

[MS02] E. Martinian and C.-E. W. Sundberg. Low delay burst erasure correction codes. *Communications, 2002. ICC 2002. IEEE International Conference on*, vol. 3, pp. 1736–1740 vol.3. 2002.

[MS01] S. Miner and J. Staddon. Graph-based authentication of digital streams. *Security and Privacy, 2001, IEEE Symposium on*, pp. 232 –246. 2001.

[MS14] C. Moore and L. J. Schulman. Tree codes and a conjecture on exponential sums. *Proceedings of the 5th Conference on Innovations in Theoretical Computer Science*, ITCS '14, pp. 145–154. ACM, 2014.

[NN90] J. Naor and M. Naor. Small-bias probability spaces: efficient constructions and applications. *STOC '90: Proceedings of the twenty-second annual ACM symposium on Theory of Computing*, pp. 213–223. ACM, New York, NY, USA, 1990.

[Pec06] M. Peczarski. An improvement of the tree code construction. *Information Processing Letters*, 99(3):92–95, 2006.

[PCTS00] A. Perrig, R. Canetti, J. Tygar, and D. Song. Efficient authentication and signing of multicast streams over lossy channels. *Security and Privacy, 2000, IEEE Symposium on*, pp. 56–73. 2000.

[RS94] S. Rajagopalan and L. Schulman. A coding theorem for distributed computation. *STOC '94: Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pp. 790–799. ACM, 1994.

[Rei60] B. Reiffen. Sequential encoding and decoding for the discrete memoryless channel. Tech. Rep. 374, Research Laboratory of Electronics. Massachusetts Institute of Technology, 1960.

[Sch92] L. J. Schulman. Communication on noisy channels: a coding theorem for computation. *Proceedings of the 33rd IEEE Annual Symposium on Foundations of Computer Science*, FOCS '92, pp. 724–733. IEEE Computer Society, 1992.

[Sch93] L. J. Schulman. Deterministic coding for interactive communication. *STOC '93: Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pp. 747–756. ACM, 1993.

[Sch96] L. J. Schulman. Coding for interactive communication. *IEEE Transactions on Information Theory*, 42(6):1745–1756, 1996.

[Sch03] L. J. Schulman. postscript to "coding for interactive communication" [Online], 2003. Available: `http://www.cs.caltech.edu/~schulman/Papers/intercodingpostscript.txt`.

[Sha58] C. E. Shannon. A note on a partial ordering for communication channels. *Information and Control*, 1(4):390 – 397, 1958.

[Sha48] C. E. Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001. Originally appeared in *Bell System Tech. J.* 27:379–423, 623–656, 1948.

[Smi07] A. Smith. Scrambling adversarial errors using few random bits, optimal information reconciliation, and better private codes. *Proceedings of the 18th annual ACM-SIAM symposium on Discrete algorithms*, SODA '07, pp. 395–404. 2007.

[THYJ12] O. Tekin, T. Ho, H. Yao, and S. Jaggi. On erasure correction coding for streaming. *Information Theory and Applications Workshop (ITA), 2012*, pp. 221–226. 2012.

[Vaz86] U. V. Vazirani. *Randomness, Adversaries and Computation.* Ph.D. thesis, EECS, UC Berkeley, 1986.

[Woz57] J. M. Wozencraft. Sequential decoding for reliable communication. Tech. Rep. 325, Research Laboratory of Electronics, Massachusetts Institute of Technology, 1957.