Synchronization Strings: Channel Simulations and Interactive Coding for Insertions and Deletions

Bernhard Haeupler[1]
Carnegie Mellon University, Pittsburgh, PA, USA
haeupler@cs.cmu.edu

Amirbehshad Shahrasbi[2]
Carnegie Mellon University, Pittsburgh, PA, USA
shahrasbi@cs.cmu.edu

Ellen Vitercik
Carnegie Mellon University, Pittsburgh, PA, USA
vitercik@cs.cmu.edu

Abstract

We present many new results related to reliable (interactive) communication over insertion-deletion channels. *Synchronization errors*, such as insertions and deletions, strictly generalize the usual *symbol corruption errors* and are much harder to protect against.

We show how to hide the complications of synchronization errors in many applications by introducing very general *channel simulations* which efficiently transform an insertion-deletion channel into a regular symbol corruption channel with an error rate larger by a constant factor and a slightly smaller alphabet. We utilize and generalize synchronization string based methods which were recently introduced as a tool to design essentially optimal error correcting codes for insertion-deletion channels. Our channel simulations depend on the fact that, at the cost of increasing the error rate by a constant factor, synchronization strings can be decoded in a streaming manner that preserves linearity of time. Interestingly, we provide a lower bound showing that this constant factor cannot be improved to $1+\varepsilon$, in contrast to what is achievable for error correcting codes. Our channel simulations drastically and cleanly generalize the applicability of synchronization strings.

We provide new interactive coding schemes which simulate any interactive two-party protocol over an insertion-deletion channel. Our results improve over the interactive coding schemes of Braverman et al. [TransInf '17] and Sherstov and Wu [FOCS '17] which achieve a small constant rate and require exponential time computations with respect to computational and communication complexities. We provide the first computationally efficient interactive coding schemes for synchronization errors, the first coding scheme with a rate approaching one for small noise rates, and also the first coding scheme that works over arbitrarily small alphabet sizes. We also show tight connections between synchronization strings and edit-distance tree codes which allow us to transfer results from tree codes directly to edit-distance tree codes.

Finally, using on our channel simulations, we provide an explicit low-rate binary insertion-deletion code that improves over the state-of-the-art codes by Guruswami and Wang [TransInf '17] in terms of rate-distance trade-off.

2012 ACM Subject Classification Mathematics of computing → Coding theory, Theory of computation → Interactive computation

Keywords and phrases Synchronization Strings, Channel Simulation, Coding for Interactive Communication

[1] Supported in part by NSF grants CCF-1527110, CCF-1618280 and NSF CAREER award CCF-1750808.
[2] Supported in part by NSF grants CCF-1527110, CCF-1618280 and NSF CAREER award CCF-1750808.

45th International Colloquium on Automata, Languages, and Programming (ICALP 2018).
Editors: Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella;
Article No. 75; pp. 75:1–75:14

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Digital Object Identifier 10.4230/LIPIcs.ICALP.2018.75

Related Version An extended version is available at `https://arxiv.org/pdf/1707.04233.pdf`.

Acknowledgements The authors thank Allison Bishop for valuable discussions in the early stages of this work.

1 Introduction

Communication in the presence of *synchronization errors*, which include both insertions and deletions, is a fundamental problem of practical importance which eluded a strong theoretical foundation for decades. This remained true even while communication in the presence of *half-errors*, which consist of symbol corruptions and erasures, has been the subject of an extensive body of research with many groundbreaking results. Synchronization errors are strictly more general than half-errors, and thus synchronization errors pose additional challenges for robust communication.

In this work, we show that one-way and interactive communication in the presence of synchronization errors can be reduced to the problem of communication in the presence of half-errors. We present a series of efficient channel simulations which allow two parties to communicate over a channel afflicted by synchronization errors as though they were communicating over a half-error channel with only a slightly larger error rate. This allows us to leverage existing coding schemes for robust communication over half-error channels in order to derive strong coding schemes resilient to synchronization errors.

One of the primary tools we use are *synchronization strings*, which were recently introduced by Haeupler and Shahrasbi in order to design essentially optimal error correcting codes (ECCs) robust to synchronization errors [19]. For every $\varepsilon > 0$, synchronization strings allow a sender to index a sequence of messages with an alphabet of size $\varepsilon^{-O(1)}$ in such a way that k synchronization errors are efficiently transformed into $(1+\varepsilon)k$ half-errors for the purpose of designing ECCs. Haeupler and Shahrasbi provide a black-box construction which transforms any ECC into an equally efficient ECC robust to synchronization errors. However, channel simulations and interactive coding in the presence of synchronization errors present a host of additional challenges that cannot be solved by the application of an ECC. Before we describe our results and techniques in detail, we begin with an overview of the well-known interactive communication model.

Interactive communication. Throughout this work, we study a scenario where there are two communicating parties, whom we call Alice and Bob. The two begin with some input symbols and wish to compute a function of their input by having a conversation. Their goal is to succeed with high probability while communicating as few symbols as possible. In particular, if their conversation would consist of n symbols in the noise-free setting, then they would like to converse for at most αn symbols, for some small α, when in the presence of noise. One might hope that Alice and Bob could correspond using error-correcting codes. However, this approach would lead to poor performance because if a party incorrectly decodes a single message, then the remaining communication is rendered useless. Therefore, only a very small amount of noise could be tolerated, namely less than the amount to corrupt a single message.

There are three major aspects of coding schemes for interactive communication that have been extensively studied in the literature. The first is the coding scheme's **maximum**

tolerable error-fraction or, in other words, the largest fraction of errors for which the coding scheme can successfully simulate any given error-free protocol. Another important quality of coding schemes for interactive communication, as with one-way communication, is **communication rate**, i.e., the amount of communication overhead in terms of the error fraction. Finally, the **efficiency** of interactive coding schemes have been of concern in the previous work.

Schulman initiated the study of error-resilient interactive communication, showing how to convert an arbitrary two-party interactive protocol to one that is robust to a $\delta = 1/240$ fraction of adversarial errors with a constant communication overhead [22, 23]. Braverman and Rao increased the bound on the tolerable adversarial error rate to $\delta < 1/4$, also with a constant communication overhead [9]. Brakerski et al. [2] designed the first efficient coding scheme resilient to a constant fraction of adversarial errors with constant communication overhead. The above-mentioned schemes achieve a constant overhead no matter the level of noise. Kol and Raz were the first to study the trade-off between error fraction and communication rate [21]. Haeupler then provided a coding scheme with a communication rate of $1 - O(\sqrt{\delta \log\log(1/\delta)})$ over an adversarial channel [17]. Further prior work has studied coding for interactive communication focusing on communication efficiency and noise resilience [18, 7, 14] as well as computational efficiency [4, 3, 2, 12, 13, 14]. Other works have studied variations of the interactive communication problem [15, 11, 10, 1, 5].

The main challenge that *synchronization errors* pose is that they may cause the parties to become "out of sync." For example, suppose the adversary deletes a message from Alice and inserts a message back to her. Neither party will know that Bob is a message behind, and if this corruption remains undetected, the rest of the communication will be useless. In most state-of-the-art interactive coding schemes for symbol corruptions, the parties communicate normally for a fixed number of rounds and then send back and forth a series of checks to detect any symbol corruptions that may have occurred. One might hope that a synchronization error could be detected during these checks, but the parties may be out of sync while performing the checks, thus rendering them useless as well. Therefore, synchronization errors require us to develop new techniques.

Very little is known regarding coding for interactive communication in the presence of synchronization errors. A 2016 coding scheme by Braverman et al. [8], which can be seen as the equivalent of Schulman's seminal result, achieves a small constant communication rate while being robust against a $1/18 - \varepsilon$ fraction of errors. The coding scheme relies on edit-distance tree codes, which are a careful adaptation of Schulman's original tree codes [23] for edit distance, so the decoding operations are not efficient and require exponential time computations. A recent work by Sherstov and Wu [25] closed the gap for maximum tolerable error fraction by introducing a coding scheme that is robust against $1/6 - \varepsilon$ fraction of errors which is the highest possible fraction of insertions and deletions under which any coding scheme for interactive communication can work. Both Braverman et al. [8] and Sherstov and Wu [25] schemes are of constant communication rate, over large enough constant alphabets, and inefficient. In this work we address the next natural questions which, as arose with ordinary corruption interactive communication, are on finding interactive coding schemes that are computationally efficient or achieve super-constant communication efficiency.

Our results. We present very general channel simulations which allow two parties communicating over an insertion-deletion channel to follow any protocol designed for a regular symbol corruption channel. The fraction of errors on the simulated symbol corruption channel is only slightly larger than that on the insertion-deletion channel. Our channel simulations

are made possible by synchronization strings. Crucially, at the cost of increasing the error rate by a constant factor, synchronization strings can be decoded in a streaming manner which preserves linearity of time. Note that the similar technique is used in Haeupler and Shahrasbi [19] to transform synchronization errors into ordinary symbol corruptions as a stepping-stone to obtain insertion-deletion codes from ordinary error correcting codes in a black-box fashion. However, in the context of error correcting codes, there is no requirement for this transformation to happen in real time. In other words, in the study of insertion-deletion codes by Haeupler and Shahrasbi [19], the entire message transmission is done and then the receiving party uses the entire message to transform the synchronization errors into symbol corruptions. In the channel simulation problem, this transformation is required to happen on the fly. Interestingly, we have found out that in the harder problem of channel simulation, the factor by which the number of synchronization errors increase by being transformed into corruption errors cannot be improved to $1 + o(1)$, in contrast to what is achievable for error correcting codes. This work exhibits the widespread applicability of synchronization strings and opens up several new use cases, such as coding for interactive communication over insertion-deletion channels. Namely, using synchronization strings, we provide techniques to obtain the following simulations of corruption channels over given insertion-deletion channels with binary and large constant alphabet sizes.

▶ **Theorem 1** (Informal Statement)**.**

(a) *Suppose that n rounds of a one-way/interactive insertion-deletion channel over an alphabet Σ with a δ fraction of insertions and deletions are given. Using an ε-synchronization string over an alphabet Σ_{syn}, it is possible to simulate $n\,(1 - O_\varepsilon(\delta))$ rounds of a one-way/interactive corruption channel over Σ_{sim} with at most $O_\varepsilon\,(n\delta)$ symbols corrupted so long as $|\Sigma_{sim}| \times |\Sigma_{syn}| \leq |\Sigma|$.*

(b) *Suppose that n rounds of a binary one-way/interactive insertion-deletion channel with a δ fraction of insertions and deletions are given. It is possible to simulate $n(1 - \Theta(\sqrt{\delta \log(1/\delta)}))$ rounds of a binary one-way/interactive corruption channel with $\Theta(\sqrt{\delta \log(1/\delta)})$ fraction of corruption errors between two parties over the given channel.*

Based on the channel simulations presented above, we present novel interactive coding schemes which simulate any interactive two-party protocol over an insertion-deletion channel.

We use our large alphabet interactive channel simulation along with constant-rate efficient coding scheme of Ghaffari and Haeupler [14] for interactive communication over corruption channels to obtain a coding scheme for insertion-deletion channels that is efficient, has a constant communication rate, and tolerates up to $1/44 - \varepsilon$ fraction of errors. Note that despite the fact that this coding scheme fails to protect against the optimal $1/6 - \varepsilon$ fraction of synchronization errors as the recent work by Sherstov and Wu [25] does, it is an improvement over all previous work in terms of computational efficiency as it is the first efficient coding scheme for interactive communication over insertion-deletion channels.

▶ **Theorem 2.** *For any constant $\varepsilon > 0$ and n-round alternating protocol Π, there is an efficient randomized coding scheme simulating Π in presence of $\delta = 1/44 - \varepsilon$ fraction of edit-corruptions with constant rate (i.e., in $O(n)$ rounds) and in $O(n^5)$ time that works with probability $1 - 2^{\Theta(n)}$. This scheme requires the alphabet size to be a large enough constant $\Omega_\varepsilon(1)$.*

Next, we use our small alphabet channel simulation and the corruption channel interactive coding scheme of Haeupler [17] to introduce an interactive coding scheme for insertion-deletion channels. This scheme is not only computationally efficient, but also the first with super

constant communication rate. In other words, this is the first coding scheme for interactive communication over insertion-deletion channels whose rate approaches one as the error fraction drops to zero. Our computationally efficient interactive coding scheme achieves a near-optimal communication rate of $1 - O(\sqrt{\delta \log(1/\delta)})$ and tolerates a δ fraction of errors. Besides computational efficiency and near-optimal communication rate, this coding scheme improves over all previous work in terms of alphabet size. As opposed to coding schemes provided by the previous work[8, 25], our scheme does not require a large enough constant alphabet and works even for binary alphabets.

▶ **Theorem 3.** *For sufficiently small δ, there is an efficient interactive coding scheme for fully adversarial binary insertion-deletion channels which is robust against δ fraction of edit-corruptions, achieves a communication rate of $1 - \Theta(\sqrt{\delta \log(1/\delta)})$, and works with probability $1 - 2^{-\Theta(n\delta)}$.*

We also utilize the channel simulations in one-way settings to provide efficient binary insertion-deletion codes correcting δ-fraction of synchronization errors–for δ smaller than some constant–with a rate of $1 - \Theta(\sqrt{\delta \log(1/\delta)})$. This is an improvement in terms of rate-distance trade-off over the state-of-the-art low-rate binary insertion-deletion codes by Guruswami and Wang [16]. The codes by Guruswami and Wang [16] achieve a rate of $1 - O(\sqrt{\delta} \log(1/\delta))$.

Finally, we introduce a slightly improved definition of edit-distance tree codes, a generalization of Schulman's original tree codes defined by Braverman et al. [8]. We show that under our revised definition, edit-distance tree codes are closely related to synchronization strings. For example, edit-distance tree codes can be constructed by merging a regular tree code and a synchronization string. This transfers, for example, Braverman's sub-exponential time tree code construction [6] and the candidate construction of Schulman [24] from tree codes to edit-distance tree codes. Lastly, as a side note, we will show that with the improved definition, the coding scheme of Braverman et al. [8] can tolerate $1/10 - \varepsilon$ fraction of synchronization errors rather than $1/18 - \varepsilon$ fraction that the scheme based on their original definition did. This improved definition is independently observed by Sherstov and Wu [25].

1.1 Definitions and preliminaries

In this section, we define the channel models and communication settings considered in this work. We also provide notation and define synchronization strings.

Error model and communication channels. In this work, we study two types of channels, which we call *corruption channels* and *insertion-deletion channels*. In the corruption channel model, two parties communicate with an alphabet Σ, and if one party sends a message $c \in \Sigma$ to the other party, then the other party will receive a message $\tilde{c} \in \Sigma$, but it may not be the case that $c = \tilde{c}$.

In the one-way communication setting over an insertion-deletion channel, messages to the listening party may be inserted, and messages sent by the sending party may be deleted. The two-way channel requires a more careful setup. We emphasize that we cannot hope to protect against arbitrary insertions and deletions in the two-way setting because in the message-driven model, a single deletion could cause the protocol execution to "hang." Therefore, following the standard model of Braverman et al.'s work [8] that is employed in all other previous works on this problem [25], we restrict our attention to *edit corruptions*, which consist of a single deletion followed by a single insertion, which may be aimed at either party. Braverman et al. [8] provide a detailed discussion on their model and show that it is strong

enough to generalize other natural models one might consider, including models that utilize clock time-outs to overcome the stalling issue.

In both the one- and two-way communication settings, we study *adversarial channels* with *error rate* δ. Our coding schemes are robust in both the *fully adversarial* and the *oblivious adversary* models.

Interactive and one-way communication protocols. In an *interactive protocol* Π over a channel with an alphabet Σ, Alice and Bob begin with two inputs from Σ^* and then engage in n *rounds* of communication. In a single round, each party either listens for a message or sends a message, where this choice and the message, if one is generated, depends on the party's state, its input, and the history of the communication thus far. After the n rounds, the parties produce an output. We study *alternating protocols*, where each party sends a message every other round and listens for a message every other round. In this message-driven paradigm, a party "sleeps" until a new message comes, at which point the party performs a computation and sends a message to the other party. In the presence of noise, we say that a protocol Π' *robustly simulates* a deterministic protocol Π over a channel C if given any inputs for Π, the parties can decode the transcript of the execution of Π on those inputs over a noise-free channel from the transcript of the execution of Π' over C.

Finally, we also study *one-way communication*, where one party sends all messages and the other party listens. Coding schemes in this setting are known as *error-correcting codes*.

Synchronization Strings. In short, synchronization strings [19] allow communicating parties to protect against synchronization errors by indexing their messages without blowing up the communication rate. We describe this technique by introducing two intermediaries, C_A and C_B, that conduct the communication over the given insertion-deletion channel. C_A receives all symbols that Alice wishes to send to Bob, C_A sends the symbols to C_B, and C_B communicates the symbols to Bob. C_A and C_B handle the synchronization strings and all the extra work that is involved in keeping Alice and Bob in sync by guessing the actual index of symbols received by C_B. In this way, Alice and Bob communicate via C_A and C_B as though they were communicating over a half-error channel.

Unfortunately, trivially attaching the indices $1, 2, \ldots, n$ to each message will not allow us to maintain a near optimal communication rate. If C_A attaches an index to each of Alice's messages, it would increase the size of Σ by a factor of n and the rate would increase by a factor of $1/\log n$, which is far from optimal. Synchronization strings allow the communicating parties to index their messages using an alphabet size that is independent of the total communication length n.

Suppose that with each of Alice's n messages, C_A sends an encoding of her index using a symbol from Σ. Let S be a "synchronization string" consisting of all n encoded indices sent by C_A. Further, suppose that the adversary injects a total of $n\delta$ insertions and deletions, thus transforming the string S to the string S_τ. Let some element of S like $S[i]$ pass through the channel without being deleted by the adversary and arrive as $S_\tau[j]$. We call $S_\tau[j]$ a *successfully transmitted symbol*.

We assume that C_A and C_B know the string S *a priori*. The intermediary C_B will receive a set of transmitted indices $S_\tau[1], \ldots, S_\tau[n]$. Upon receipt of the jth transmitted index, C_B guesses the actual index of the received symbol when sent by C_A. We call the algorithm that C_B runs to determine this an *(n, δ)-indexing algorithm.* The algorithm can also return a symbol $\top$ which represents an "I don't know" response. Any successfully transmitted symbols that is decoded incorrectly is called a *misdecoding*. The number of misdecodings that

an (n, δ)-indexing algorithm might produced is used as a measure to valuate its quality. An indexing algorithm is *streaming* if its guess for a received symbol only depends on previously arrived symbols.

Haeupler and Shahrasbi [19] defined a family of synchronization strings that admit an (n, δ)-indexing algorithm with strong performance.

▶ **Definition 4** (ε-Synchronization String)**.** A string $S \in \Sigma^n$ is an ε-synchronization string if for every $1 \leq i < j < k \leq n+1$ we have that $ED\left(S[i,j), S[j,k)\right) > (1-\varepsilon)(k-i)$.

Haeupler and Shahrasbi [19, 20] prove the existence and provide several fast constructions for ε-synchronization strings and provide a streaming (n, δ)-indexing algorithm that returns a solution with $\frac{c_i}{1-\varepsilon} + \frac{c_d \varepsilon}{1-\varepsilon}$ misdecodings. The algorithm runs in time $O(n^5)$, spending $O(n^4)$ on each received symbol.

2 Channel Simulations

In this section, we show how ε-synchronization strings can be used as a powerful tool to simulate corruption channels over insertion-deletion channels. In Section 3, we use these simulations to introduce coding schemes resilient to insertion-deletion errors.

2.1 One-way channel simulation over a large alphabet

Assume that Alice and Bob have access to n rounds of communication over a one-way insertion-deletion channel where the adversary is allowed to insert or delete up to $n\delta$ symbols. In this situation, we formally define a corruption channel simulation over the given insertion-deletion channel as follows:

▶ **Definition 5** (Corruption Channel Simulation)**.** Let Alice and Bob have access to n rounds of communication over a one-way insertion-deletion channel with the alphabet Σ. The adversary may insert or delete up to $n\delta$ symbols. Intermediaries C_A and C_B *simulate* n' rounds of a corruption channel with alphabet Σ_{sim} over the given channel as follows. First, the adversary can insert a number of symbols into the insertion-deletion channel between C_A and C_B. Then for n' rounds $i = 1, \ldots, n'$, the following procedure repeats:

1. Alice gives $X_i \in \Sigma_{sim}$ to C_A.
2. Upon receiving X_i from Alice, C_A wakes up and sends a number of symbols (possibly zero) from the alphabet Σ to C_B through the given insertion-deletion channel. The adversary can delete any of these symbols or insert symbols *before*, *among*, or *after* them.
3. Upon receiving symbols from the channel, C_B wakes up and reveals a number of symbols (possibly zero) from the alphabet Σ_{sim} to Bob. We say all such symbols are *triggered* by X_i.

Throughout this procedure, the adversary can insert or delete up to $n\delta$ symbols. However, C_B is required to reveal exactly n' symbols to Bob regardless of the adversary's actions. Let $\tilde{X}_1, \cdots, \tilde{X}_{n'} \in \Sigma_{sim}$ be the symbols revealed to Bob by C_B. This procedure successfully simulates n' rounds of a corruption channel with a δ' fraction of errors if for all but $n'\delta'$ elements i of the set $\{1, \ldots, n'\}$, the following conditions hold: 1) $\tilde{X}_i = X_i$; and 2) $\tilde{X}_i$ is triggered by X_i.

When $\tilde{X}_i = X_i$ and $\tilde{X}_i$ is triggered by X_i, we call $\tilde{X}_i$ an *uncorrupted symbol*. The second condition, that $\tilde{X}_i$ is triggered by X_i, is crucial to preserving linearity of time, which is the fundamental quality that distinguishes channel simulations from channel codings. It forces C_A to communicate each symbol to Alice as soon as it arrives. Studying channel simulations

satisfying this condition is especially important in situations where Bob's messages depends on Alice's, and vice versa.

Conditions (1) and (2) also require that C_B conveys at most one uncorrupted symbol each time he wakes up. As the adversary may delete $n\delta$ symbols from the insertion-deletion channel, C_B will wake up at most $n(1-\delta)$ times. Therefore, we cannot hope for a corruption channel simulation where Bob receives more than $n(1-\delta)$ uncorrupted symbols. In the following theorem, we prove something slightly stronger: no deterministic one-way channel simulation can guarantee that Bob receives more than $n(1-4\delta/3)$ uncorrupted symbols and if the simulation is randomized, the expected number of uncorrupted transmitted symbols is at most $n(1-7\delta/6)$. This puts channel simulation in contrast to channel coding as one can recover $1-\delta-\varepsilon$ fraction of symbols there (as shown in [19]).

▶ **Theorem 6.** *Assume that n uses of a one-way insertion-deletion channel over an arbitrarily large alphabet Σ with a δ fraction of insertions and deletions are given. There is no deterministic simulation of a corruption channel over any alphabet Σ_{sim} where the simulated channel guarantees more than $n\,(1-4\delta/3)$ uncorrupted transmitted symbols. If the simulation is randomized, the expected number of uncorrupted transmitted symbols is at most $n(1-7\delta/6)$.*

We now provide a channel simulation using ε-synchronization strings. Every time C_A receives a symbol from Alice (from an alphabet Σ_{sim}), C_A appends a new symbol from a predetermined ε-synchronization string over an alphabet Σ_{syn} to Alice's symbol and sends it as one message through the channel. On the other side of channel, suppose that C_B has already revealed some number of symbols to Bob. Let $\mathtt{I}_B$ be the index of the next symbol C_B expects to receive. Upon receiving a new symbol from C_A, C_B uses the part of the message coming from the synchronization string to guess the index of the message Alice sent. We will refer to this decoded index as $\tilde{\mathtt{I}}_A$ and its actual index as $\mathtt{I}_A$. If $\widetilde{\mathtt{I}}_A < \mathtt{I}_B$, then C_B reveals nothing to Bob and ignores the message he just received. Meanwhile, if $\widetilde{\mathtt{I}}_A = \mathtt{I}_B$, then C_B reveals Alice's message to Bob. Finally, if $\widetilde{\mathtt{I}}_A > \mathtt{I}_B$, then C_B sends a dummy symbol to Bob and then sends Alice's message.Theorem 7 details the simulation guarantees.

▶ **Theorem 7.** *Assume that n uses of a one-way insertion-deletion channel over an alphabet Σ with a δ fraction of insertions and deletions are given. Using an ε-synchronization string over an alphabet Σ_{syn}, it is possible to simulate $n(1-\delta)$ rounds of a one-way corruption channel over Σ_{sim} with at most $2n\delta(2+(1-\varepsilon)^{-1})$ symbols corrupted so long as $|\Sigma_{sim}| \times |\Sigma_{syn}| \leq |\Sigma|$ and $\delta < 1/7$.*

2.2 Interactive channel simulation over a large alphabet

We now turn to channel simulations for interactive channels. As in Section 2.1, we formally define a corruption interactive channel simulation over a given insertion-deletion interactive channel. We then use synchronization strings to present one such simulation.

▶ **Definition 8** (Corruption Interactive Channel Simulation)**.** Let Alice and Bob have access to n rounds of communication over an interactive insertion-deletion channel with alphabet Σ. The adversary may insert or delete up to $n\delta$ symbols. The intermediaries C_A and C_B *simulate* n' rounds of a corruption interactive channel with alphabet Σ_{sim} over the given channel as follows. The communication starts when Alice gives a symbol from Σ_{sim} to C_A. Then Alice, Bob, C_A, and C_B continue the communication as follows:

1. Whenever C_A receives a symbol from Alice or C_B, he either reveals a symbol from Σ_{sim} to Alice or sends a symbol from Σ through the insertion-deletion channel to C_B.

2. Whenever C_B receives a symbol from Bob or C_A, he either reveals a symbol from Σ_{sim} to Bob or send a symbols from Σ through the insertion-deletion channel to C_A.
3. Whenever C_B reveals a symbol to Bob, Bob responds with a new symbol from Σ_{sim}.
4. Whenever C_A reveals a symbol to Alice, Alice responds with a symbol in Σ_{sim} except for the $\frac{n'}{2}$th time.

Throughout this procedure, the adversary can inject up to $n\delta$ edit corruptions. However, regardless of the adversary's actions, C_A and C_B have to reveal exactly $n'/2$ symbols to Alice and Bob respectively.

Let $X_1, \ldots, X_{n'}$ be the symbols Alice gives to C_A and $\tilde{X}_1, \ldots, \tilde{X}_{n'} \in \Sigma_{sim}$ be the symbols C_B reveals to Bob. Similarly, Let $Y_1, \ldots, Y_{n'}$ be the symbols Bob gives to C_B and $\tilde{Y}_1, \ldots, \tilde{Y}_{n'} \in \Sigma_{sim}$ be the symbols C_A reveals to Alice. We call each pair of tuples $(X_i, \tilde{X}_i)$ and $(Y_i, \tilde{Y}_i)$ a *round* of the simulated communication. We call a round *corrupted* if its elements are not equal. This procedure successfully simulates n' rounds of a corruption interactive channel with a δ' fraction of errors if for all but $n'\delta'$ of the rounds are corrupted.

The protocol and analysis in this large alphabet setting are similar to the harder case where the alphabet is binary. We cover interactive communication for the binary setting in the next section.

▶ **Theorem 9.** *Assume that n uses of an interactive insertion-deletion channel over an alphabet Σ with a δ fraction of insertions and deletions are given. Using an ε-synchronization string over an alphabet Σ_{syn}, it is possible to simulate $n - 2n\delta(1 + (1-\varepsilon)^{-1})$ uses of an interactive corruption channel over Σ_{sim} with at most a $\frac{2\delta(5-3\varepsilon)}{1-\varepsilon+2\varepsilon\delta-4\delta}$ fraction of symbols corrupted so long as $|\Sigma_{sim}| \times |\Sigma_{syn}| \leq |\Sigma|$ and $\delta < 1/14$.*

2.3 Binary interactive channel simulation

We now show that with the help of synchronization strings, a binary interactive insertion-deletion channel can be used to simulate a binary interactive corruption channel, inducing a $\widetilde{O}(\sqrt{\delta})$ fraction of bit-flips. In this way, the two communicating parties may interact as though they are communicating over a corruption channel. They therefore can employ corruption channel coding schemes while using the simulator as a black box means of converting the insertion-deletion channel to a corruption channel.

The key difference between this simulation and the one-way, large alphabet simulation is that Alice and Bob communicate through C_A and C_B for *blocks* of r rounds, between which C_A and C_B check if they are in sync. Due to errors, there may be times when Alice and Bob are in disagreement about which block, and what part of the block, they are in. C_A and C_B ensure that Alice and Bob are in sync most of the time.

When Alice sends C_A a message from a new block of communication, C_A holds that message and alerts C_B that a new block is beginning. C_A does this by sending C_B a header that is a string consisting of a single one followed by $s-1$ zeros (10^{s-1}). Then, C_A indicates which block Alice is about to start by sending a synchronization symbol to C_B. Meanwhile, when C_B receives a 10^{s-1} string, he listens for the synchronization symbol, makes his best guess about which block Alice is in, and then communicates with Bob and C_A accordingly. This might entail sending dummy blocks to Bob or C_A if he believes that they are in different blocks. To describe the guarantee that our simulation provides, we first define *block corruption channels*.

▶ **Definition 10** (Block Corruption Channel)**.** An n-round adversarial corruption channel is called a (δ, r)-*block corruption channel* if the adversary is restricted to corrupt $n\delta$ symbols which are covered by $n\delta/r$ blocks of r consecutively transmitted symbols.

▶ **Theorem 11.** *Suppose that n rounds of a binary interactive insertion-deletion channel with a δ fraction of insertions and deletions are given. For sufficiently small δ, it is possible to deterministically simulate $n(1-\Theta(\sqrt{\delta\log(1/\delta)}))$ rounds of a binary interactive $(\Theta(\sqrt{\delta\log(1/\delta)}), \sqrt{(1/\delta)\log(1/\delta)})$-block corruption channel between two parties, Alice and Bob, assuming that all substrings of form 10^{s-1} where $s = c\log(1/\delta)$ that Alice sends can be covered by $n\delta$ intervals of $\sqrt{(1/\delta)\log(1/\delta)}$ consecutive rounds. The simulation is performed efficiently if the synchronization string is efficient.*

Proof Sketch. Suppose Alice and Bob communicate via intermediaries C_A and C_B who act according to the algorithm described above. In total, Alice and Bob will attempt to communicate n_s bits to one another over the simulated channel, while C_A and C_B communicate a total of n bits to one another. The adversary is allowed to insert or delete up to $n\delta$ symbols and C_A sends $n/2$ bits, so C_B may receive between $n/2 - n\delta$ and $n/2 + n\delta$ symbols. To prevent C_B from stalling indefinitely, C_B only listens to the first $n(1-2\delta)/2$ bits he receives.

For $r = \sqrt{(1/\delta)\log(1/\delta)}$, we define a *chunk* to be $r_c := (s + |\Sigma_{syn}| + r/2)$ consecutive bits that are sent by C_A to C_B. In particular, a chunk corresponds to a section header and synchronization symbol followed by $r/2$ rounds of messages sent from Alice. As C_B cares about the first $n(1-2\delta)/2$ bits it receives, there are $\frac{n(1-2\delta)}{2r_c}$ chunks in total. Hence, $n_s = \frac{n(1-2\delta)}{2r_c} \cdot r$ since C_B and C_A's communication is alternating.

Note that if Alice sends a substring of form 10^{s-1} in the information part of a chunk, then Bob mistakenly detects a new block. With this in mind, we say a chunk is *good* if:

1. No errors are injected in the chunk or affecting C_B's detection of the chunk's header,
2. C_B correctly decodes the index that C_A sends during the chunk, and
3. C_A does not send a 10^{s-1} substring in the information portion of the chunk.

If a chunk is not good, we call it *bad*. If the chunk is bad because C_B does not decode C_A's index correctly even though they were in sync and no errors were injected, then we call it *decoding-bad*. If it is bad because Alice sends a 10^{s-1} substring, we call it *zero-bad* and otherwise, we call it *error-bad*. Throughout the protocol, C_B uses the variable $\mathtt{I}_B$ to denote the next index of the synchronization string C_B expects to receive and we use $\mathtt{I}_A$ to denote the index of the synchronization string C_A most recently sent. Notice that if a chunk is good and $\mathtt{I}_A = \mathtt{I}_B$, then all messages are correctly conveyed.

We now bound the maximum number of bad chunks that occur over the course of the simulation. Suppose the adversary injects errors into the i^{th} chunk, making that chunk bad. The $(i+1)^{th}$ chunk may also be bad, since Bob may not be listening for 10^{s-1} from C_A when C_A sends them, and therefore may miss the block header. However, if the adversary does not inject any errors into the $(i+1)^{th}$ and the $(i+2)^{th}$ chunk, then the $(i+2)^{th}$ chunk will be good. In effect, a single error may render at most two chunks useless. Since the adversary may inject $n\delta$ errors into the insertion-deletion channel, this means that the number of chunks that are error-bad is at most $2n\delta$. Additionally, by assumption, the number of zero-bad chunks is also at most $n\delta$.

We also must consider the fraction of rounds that are decoding-bad. In order to do this, we appeal to Theorem 6.24 from [19], which guarantees that if an ε-synchronization string of length N is sent over an insertion-deletion channel with a δ' fraction of insertions and deletions, then the receiver will decode the index of the received symbol correctly for all but $2N\delta'/(1-\varepsilon)$ symbols. In this context, N is the number of chunks, i.e. $N = n(1-2\delta)/(2r_c)$, and the fraction of chunks corrupted by errors is $\delta' = 4n\delta/N$. Therefore, the total number of bad chunks is at most $4\delta n + 2N\delta'/(1-\varepsilon) = 4\delta n(3-\varepsilon)/(1-\varepsilon)$.

In the rest of the proof, which is available in the extended version of this paper, we show that all but $12\frac{3-\varepsilon}{1-\varepsilon} \cdot \delta n$ chunks are good chunks and have $\mathtt{I}_A = \mathtt{I}_B$ upon their arrival on Bob's side and we conclude that the simulated channel is a $\left(\frac{3-\varepsilon}{1-\varepsilon}\frac{24\delta}{1-2\delta}r_c, r\right)$-block corruption channel. For the asymptotically optimal choice of $r = \sqrt{(1/\delta)\log(1/\delta)}$, we derive the simulation described in the theorem statement. ◀

The simulation stated in Theorem 11 burdens an additional condition on Alice's stream of bits by requiring it to have a limited number of substrings of form 10^{s-1}. We now introduce a high probability technique to modify a general interactive communication protocol in a way that makes all substrings of form 10^{s-1} in Alice's stream of bits fit into $n\delta$ intervals of length $r = \sqrt{(1/\delta)\log(1/\delta)}$.

▶ **Lemma 12.** *Assume that n rounds of a binary interactive insertion-deletion channel with an oblivious adversary who is allowed to inject $n\delta$ errors are given. There is a pre-coding scheme that can be utilized on top of the simulation introduced in Theorem 11. It modifies the stream of bits sent by Alice so that with probability $1 - e^{-\frac{c-3}{2}n\delta\log\frac{1}{\delta}(1+o(1))}$, all substrings of form 10^{s-1} where $s = c\log(1/\delta)$ in the stream of bits Alice sends over the simulated channel can be covered by $n\delta$ intervals of length $r = \sqrt{(1/\delta)\log(1/\delta)}$. This pre-coding scheme comes at the cost of a $\Theta(\sqrt{\delta\log(1/\delta)})$ fraction of the bits Alice sends through the simulated channel.*

Proof sketch. In the simulation process, each $r/2$ consecutive bits Alice sends forms one of the chunks C_A sends to C_B alongside some headers. The idea of this pre-coding scheme is simple. Alice uses the first $s/2$ data bits (and not the header) of each chunk to share $s/2$ randomly generated bits with Bob (instead of running the interactive protocol) and then both of them extract a string S' of $r/2$ $(s/2)$-wise independent random variables. Then, Alice XORs the rest of data bits she passes to C_A with S' and Bob XORs those bits with S' again to retrieve the original data. In the extended version, we show that this pre-coding scheme guarantees the requirements mentioned in the theorem statement. ◀

Applying this pre-coding for $c \geq 3$ on top of the simulation from Theorem 11 implies the following.

▶ **Theorem 13.** *Suppose that n rounds of a binary interactive insertion-deletion channel with a δ fraction of insertions and deletions performed by an oblivious adversary are given. For sufficiently small δ, it is possible to simulate $n(1 - \Theta(\sqrt{\delta\log(1/\delta)}))$ rounds of a binary interactive $(\Theta(\sqrt{\delta\log(1/\delta)}), \sqrt{(1/\delta)\log 1/\delta})$-block corruption channel between two parties over the given channel. The simulation works with probability $1 - \exp(-\Theta(n\delta\log(1/\delta)))$ and is efficient if the synchronization string is efficient.*

▶ **Lemma 14.** *Suppose that n rounds of a binary, interactive, fully adversarial insertion-deletion channel with a δ fraction of insertions and deletions are given. The pre-coding scheme proposed in Lemma 12 ensures that the stream of bits sent by Alice contains fewer than $n\delta$ substrings of form 10^{s-1} for $s = c\log(1/\delta)$ and $c > 5$ with probability $1 - e^{-\Theta(n\delta\log(1/\delta))}$.*

Theorem 11 and Lemma 14 allow us to conclude that one can perform the simulation stated in Theorem 11 over any interactive protocol with high probability. Note that one can trivially extend the results of Theorems 11 and 13 to one-way binary communication by ignoring the bits Bob sends.

3 Applications: New Interactive Coding Schemes

Efficient Coding Scheme Tolerating 1/44 Fraction of Errors. In this section, we will provide an efficient coding scheme for interactive communication over insertion-deletion channels by first making use of large alphabet interactive channel simulation provided in Theorem 9 to effectively transform the given channel into a simple corruption interactive channel and then use the efficient constant-rate coding scheme of Ghaffari and Haeupler [14] on top of the simulated channel. This will give an efficient constant-rate interactive communication over large enough constant alphabets as described in Theorem 2. We review the following theorem of Ghaffari and Haeupler [14] before proving Theorem 2.

▶ **Theorem 15** (Theorem 1.1 from [14]). *For any constant $\varepsilon > 0$ and n-round protocol Π there is a randomized non-adaptive coding scheme that robustly simulates Π against an adversarial error rate of $\rho \leq 1/4 - \varepsilon$ using $N = O(n)$ rounds, a near-linear $n \log^{O(1)} n$ computational complexity, and failure probability $2^{-\Theta(n)}$.*

Proof of Theorem 2. For a given insertion-deletion interactive channel over alphabet Σ suffering from δ fraction of edit-corruption errors, Theorem 9 enables us to simulate $n - 2n\delta(1 + (1-\varepsilon')^{-1})$ rounds of ordinary interactive channel with $\frac{2\delta(5-3\varepsilon')}{1-\varepsilon'+2\varepsilon'\delta-4\delta}$ fraction of symbol by designating $\log|\Sigma_{syn}|$ bits of each symbol to index simulated channel's symbols with an ε'-synchronization string over Σ_{syn}.

One can employ the scheme of Ghaffari and Haeupler [14] over the simulated channel as long as error fraction is smaller than $1/4$. Note that $\frac{2\delta(5-3\varepsilon')}{1-\varepsilon'+2\delta\varepsilon'-4\delta}\Big|_{\varepsilon'=0} = \frac{10\delta}{1-4\delta} < \frac{1}{4} \Leftrightarrow \delta < \frac{1}{44}$. Hence, as long as $\delta = 1/44 - \varepsilon$ for $\varepsilon > 0$, for small enough $\varepsilon' = O_\varepsilon(1)$, the simulated channel has an error fraction that is smaller than $1/4$. Therefore, by running the efficient coding scheme of Theorem 15 over this simulated channel one gets a constant rate coding scheme for interactive communication that is robust against $1/44 - \varepsilon$ fraction of edit-corruptions. Note that this simulation requires the alphabet size to be large enough to contain synchronization symbols (which can come from a polynomially large alphabet in terms of ε') and also meet the alphabet size requirements of Theorem 15. This requires the alphabet size to be $\Omega_\varepsilon(1)$, i.e., a large enough constant merely depending on ε. The success probability and time complexity are direct consequences of Theorem 15 and Theorem 6.24 from [19]. ◀

Efficient Coding Scheme with Near-Optimal Rate over Small Alphabets. In this section we present another insertion-deletion interactive coding scheme that achieves near-optimal communication efficiency as well as computation efficiency by employing a similar idea as in Section 3.

In order to derive a rate-efficient interactive communication coding scheme over small alphabet insertion-deletion channels, simulations described above can be used to simulate a corruption channel and then the rate-efficient interactive coding scheme for corruption channels introduced by Haeupler [17] can be used on top of the simulated channel.

▶ **Theorem 16** (Interactive Coding against Block Corruption). *By choosing an appropriate block length in the Haeupler [17] coding scheme for oblivious adversaries, one obtains a robust efficient interactive coding scheme for (δ_b, r_b)-block corruption channel with communication rate $1 - \Theta(\sqrt{\delta_b \max\{\delta_b, 1/r_b\}})$ that works with probability $1 - 2^{-\Theta(n\delta_b/r_b)}$.*

Applying the coding scheme of Theorem 16 over the simulation from Theorem 13 implies the following.

▶ **Theorem 17.** *For sufficiently small δ, there is an efficient interactive coding scheme over binary insertion-deletion channels which, is robust against δ fraction of edit-corruptions by an oblivious adversary, achieves a communication rate of $1 - \Theta(\sqrt{\delta \log(1/\delta)})$, and works with probability $1 - 2^{-\Theta(n\delta)}$.*

Moreover, in the extended version, we show that this result is extendable for the fully adversarial setup, as summarized in Theorem 3.

This insertion-deletion interactive coding scheme is, to the best of our knowledge, the first to be computationally efficient, to have communication rate approaching one, and to work over arbitrarily small alphabets.

References

1. Shweta Agrawal, Ran Gelles, and Amit Sahai. Adaptive protocols for interactive communication. In *Information Theory (ISIT), 2016 IEEE International Symposium on*, pages 595–599, 2016.
2. Zvika Brakerski, Yael Tauman Kalai, and Moni Naor. Fast interactive coding against adversarial noise. *Journal of the ACM (JACM)*, 61(6):35, 2014.
3. Zvika Brakerski and Moni Naor. Fast algorithms for interactive coding. In *Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 443–456, 2013.
4. Zvika Brakerski and Yael Tauman Kalai. Efficient interactive coding against adversarial noise. In *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)*, pages 160–166, 2012.
5. Gilles Brassard, Ashwin Nayak, Alain Tapp, Dave Touchette, and Falk Unger. Noisy interactive quantum communication. In *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)*, pages 296–305, 2014.
6. Mark Braverman. Towards deterministic tree code constructions. In *Proceedings of the ACM Conference on Innovations in Theoretical Computer Science (ITCS)*, pages 161–167, 2012.
7. Mark Braverman and Klim Efremenko. List and unique coding for interactive communication in the presence of adversarial noise. *SIAM Journal on Computing*, 46(1):388–428, 2017.
8. Mark Braverman, Ran Gelles, Jieming Mao, and Rafail Ostrovsky. Coding for interactive communication correcting insertions and deletions. *IEEE Transactions on Information Theory*, 63(10):6256–6270, 2017.
9. Mark Braverman and Anup Rao. Toward coding for maximum errors in interactive communication. *IEEE Transactions on Information Theory*, 60(11):7248–7255, 2014.
10. Klim Efremenko, Gelles Ran, and Haeupler Bernhard. Maximal noise in interactive communication over erasure channels and channels with feedback. *IEEE Transactions on Information Theory*, 62(8):4575–4588, 2016.
11. Matthew Franklin, Ran Gelles, Rafail Ostrovsky, and Leonard J. Schulman. Optimal coding for streaming authentication and interactive communication. *IEEE Transactions on Information Theory*, 61(1):133–145, 2015.
12. Ran Gelles, Ankur Moitra, and Amit Sahai. Efficient and explicit coding for interactive communication. In *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)*, pages 768–777, 2011.
13. Ran Gelles, Ankur Moitra, and Amit Sahai. Efficient coding for interactive communication. *IEEE Transactions on Information Theory*, 60(3):1899–1913, 2014.
14. Mohsen Ghaffari and Bernhard Haeupler. Optimal error rates for interactive coding ii: Efficiency and list decoding. In *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)*, pages 394–403, 2014.

15 Mohsen Ghaffari, Bernhard Haeupler, and Madhu Sudan. Optimal error rates for interactive coding i: Adaptivity and other settings. In *Proceedings of the Annual Symposium on Theory of Computing (STOC)*, pages 794–803, 2014.

16 Venkatesan Guruswami and Carol Wang. Deletion codes in the high-noise and high-rate regimes. *IEEE Transactions on Information Theory*, 63(4):1961–1970, 2017.

17 Bernhard Haeupler. Interactive channel capacity revisited. In *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)*, pages 226–235, 2014.

18 Bernhard Haeupler and Ran Gelles. Capacity of interactive communication over erasure channels and channels with feedback. In *Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1296–1311, 2015.

19 Bernhard Haeupler and Amirbehshad Shahrasbi. Synchronization strings: Codes for insertions and deletions approaching the singleton bound. In *Proceedings of the Annual Symposium on Theory of Computing (STOC)*, 2017.

20 Bernhard Haeupler and Amirbehshad Shahrasbi. Synchronization strings: Explicit constructions, local decoding, and applications. In *Proceedings of the Annual Symposium on Theory of Computing (STOC)*, 2018.

21 Gillat Kol and Ran Raz. Interactive channel capacity. In *Proceedings of the Annual Symposium on Theory of Computing (STOC)*, pages 715–724, 2013.

22 Leonard J. Schulman. Communication on noisy channels: A coding theorem for computation. In *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)*, pages 724–733, 1992.

23 Leonard J. Schulman. Deterministic coding for interactive communication. In *Proceedings of the Annual Symposium on Theory of Computing (STOC)*, pages 747–756, 1993.

24 Leonard J. Schulman. Postscript to "Coding for interactive communication". [Online; accessed 17-March-2017], 2003. URL: `http://users.cms.caltech.edu/~schulman/Papers/intercodingpostscript.txt`.

25 Alexander A. Sherstov and Pei Wu. Optimal interactive coding for insertions, deletions, and substitutions. In *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)*, pages 240–251, 2017.