Interactive Coding with Unbounded Noise

Eden Fargion
Faculty of Engineering, Bar-Ilan University, Ramat-Gan, Israel

Ran Gelles
Faculty of Engineering, Bar-Ilan University, Ramat-Gan, Israel

Meghal Gupta
University of California, Berkeley, CA, USA

Abstract

Interactive coding allows two parties to conduct a distributed computation despite noise corrupting a certain fraction of their communication. Dani et al. (Inf. and Comp., 2018) suggested a novel setting in which the amount of noise is unbounded and can significantly exceed the length of the (noise-free) computation. While no solution is possible in the worst case, under the restriction of *oblivious* noise, Dani et al. designed a coding scheme that succeeds with a polynomially small failure probability.

We revisit the question of conducting computations under this harsh type of noise and devise a computationally-efficient coding scheme that guarantees the success of the computation, except with an *exponentially* small probability. This higher degree of correctness matches the case of coding schemes with a bounded fraction of noise.

Our simulation of an N-bit noise-free computation in the presence of T corruptions, communicates an optimal number of $O(N+T)$ bits and succeeds with probability $1-2^{-\Omega(N)}$. We design this coding scheme by introducing an intermediary noise model, where an oblivious adversary can choose the locations of corruptions in a worst-case manner, but the effect of each corruption is random: the noise either flips the transmission with some probability or otherwise erases it. This randomized abstraction turns out to be instrumental in achieving an optimal coding scheme.

2012 ACM Subject Classification Theory of computation → Interactive computation; Mathematics of computing → Coding theory; Computing methodologies → Distributed algorithms

Keywords and phrases Distributed Computation with Noisy Links, Interactive Coding, Noise Resilience, Unbounded Noise, Random Erasure-Flip Noise

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2024.43

Category RANDOM

Related Version *Full Version*: `https://arxiv.org/abs/2407.09463` [10]

Funding *Ran Gelles*: partially supported by a grant from the United States-Israel Binational Science Foundation (BSF), Jerusalem, Israel, Grant No. 2020277.

Acknowledgements E. Fargion and R. Gelles would like to thank Paderborn University for hosting them while part of this research was done. R. Gelles would like to thank CISPA – Helmholtz Center for Information Security for hosting him.

1 Introduction

In many distributed systems nowadays, communication channels suffer various types of noise and interference that may corrupt information exchanged between devices. *Interactive coding*, initiated by the seminal work of Schulman [25, 26] (see also [12]), allows two or more devices to correctly complete their computation despite channel noise, by adding only a moderate amount of redundancy to the computation. The capability of an interactive coding scheme usually depends on the specific type of noise it is designed to withstand. For instance, when

the noise can flip bits, interactive coding schemes withstand up to a fraction of 1/6 of flipped bits [8, 20] (a fraction of 1/4 can be withstood over channels with larger alphabets [5]); when the noise erases bits (i.e., replaces a bit with a special erasure mark $\perp$), then a fraction of 1/2 of bit erasures can be withstood [11, 20], which also applies for larger alphabets [8]. When messages can be inserted and deleted, the maximal corruption rate is again 1/4, see [4, 27].

In a recent work, Dani, Hayes, Movahedi, Saia, and Young [7] suggested a different and interesting model for interactive coding in which the amount of noise is *unbounded*. That is, the number T of corruptions that affects a given execution, can be arbitrary. Note that this number T is unknown to the coding scheme; this is in contrast to the standard model of interactive coding, where a limit on the fraction of corrupted transmissions is known by all devices. The scheme in [7] correctly computes any two-party computation that takes N rounds without noise, by communicating $N + O(T + \sqrt{N(T+1)\log T})$ bits and succeeds with probability $1 - O(1/N \log N)$.

In a nutshell, the idea of the scheme in [7] is as follows. Every message sent between the parties contains the round number it corresponds to and a signature. A device checks that the signature is valid before processing a received message. If the signature does not check out, the device ignores that communication. The coding scheme tracks the progress of both parties via the added information of the round number, so that corrupted messages are re-transmitted until they arrive correctly at the other side.

One significant drawback of the above approach, is that the noise might corrupt a message along with its signature so that the receiver believes that the signature is correct. This occurs with exponentially small probability in the length of the signature, which leads to the polynomially-small failure probability of the scheme. In other words, the scheme in [7] assumes that the noise *never creates a valid signature* and settles with a failure probability of magnitude $1/N \log N$.

In this work we aim to achieve an interactive coding scheme that can withstand an unbounded amount of noise, yet, with failure probability exponentially small in N, similar to most previous work on interactive coding (e.g., [26, 18, 2, 19]). This effectively means that the coding scheme must cope with corrupted messages being processed by some device. That is, the coding scheme must be resilient to the event, that occurs with polynomially small probability in N, where both the message and the signature are corrupted in a matching way.

Our main result is a coding scheme that is resilient to an arbitrary and a priori unknown number T of bit flips, with exponentially small failure probability.

▶ **Theorem 1.1** (Main). *Given any two-party binary interactive protocol π of length N, there exists an efficient randomized protocol Π of length $O(N + T)$ that simulates π with probability $1 - 2^{-\Omega(N)}$ over a binary channel in the presence of an arbitrary and a priori unknown number T of corruptions. The noise is assumed to be independent of the parties' randomness.*

We note that the scheme assumes *oblivious noise* in the sense that the T corrupted transmissions are selected at the beginning of the computation (as a function of the coding scheme and the parties' inputs) and is independent of the parties' (private) randomness. This assumption is crucial, as no coding scheme withstands an unbounded amount of noise that is non-oblivious [7, Theorem 6.1].

1.1 Techniques

Towards an optimal scheme: the code concatenation approach. The immediate approach towards an improved coding scheme for an unbounded amount of corruptions is of *code concatenation*, namely composing two layers of interactive code. The inner layer would be

responsible for transmitting bits over the channel despite the unbounded amount of noise (e.g., [7, 15]). The outer layer would then "see" only a limited amount of noise (which passes the inner layer with polynomially-small probability) and perform a standard interactive coding (e.g., [25, 26, 5, 21, 19, 14, 20]) using these bits.

Unfortunately, such an approach faces severe difficulties. For the inner layer, the scheme of Dani et al. [7] assumes that no corrupted message is accepted by a party. That is, a message can either be correct or marked as invalid. Requiring the parties to process incorrect messages might cause their inner state to differ in a way that could not be recovered by the scheme. That is, a single corrupted message (that is believed to be correct by one of the parties) might cause the parties to "lose sync", so that the parties do not agree anymore on when the next phase of the scheme begins and ends, or whether the scheme has terminated or not. The scheme will not recover from this fault because the synchronization information would be sent by one party at certain rounds but expected by the other party at different rounds. The other option for the inner layer is the scheme by Gelles and Iyer [15] designed to withstand an unbounded amount of *erasures*, and thus, on its surface, does not fit our purpose.

The randomized erasure–flip model. The key towards solving the above conundrum stems from defining a new random noise model that we name the *unbounded probabilistic Erase–Flip noise model* (UPEF). This model (formally defined in Section 2.2) still allows an unbounded number of corruptions determined by the adversary in an oblivious way. However, when the i-th transmission is corrupted by the adversary, the *effect* of the corruption is random: the transmitted bit is flipped with some probability p_i or erased with the complementary probability. The probabilities $\{p_i\}_{i\in\mathbb{N}}$ are parameters of the model and can be determined by the algorithm's designer. In a sense, this type of noise matches the effect oblivious noise has on messages that are protected with a signature: with some probability the corruption is detected (the signature does not verify) and the message is marked as corrupted, i.e., erased. On the other hand, with some small probability the corruption is such that the signature verifies the corrupted message; in this case we have a flip. The probabilities are determined by the length of the signatures in use.[1]

This novel randomized model is much simpler to handle, and facilitates the design and analysis of optimal coding schemes. Furthermore, any scheme designed for this model can be translated into a coding scheme that works in the standard unbounded flips (UF) noise model, by employing signatures of respective length to match the erasure–flip probabilities of the UPEF model. Therefore, this model serves as a crucial tool for obtaining optimal coding schemes in the standard model.

Switching to the UPEF model allows us to use the scheme in [15] as the inner code of our concatenated coding scheme, in an almost as-is fashion: by smartly setting the probabilities $\{p_i\}_{i\in\mathbb{N}}$, we can guarantee, with very high probability, that any execution experiences an unbounded number of erasures but only a bounded number of bit-flips. The scheme in [15] withstands the erasures and delivers the non-erased bits (either correct or not) to the outer layer, which should be able to cope with this limited amount of bit flips.

One problem still remains, but to explain it, we must first explain how the [15] scheme works. In a nutshell, the parties simulate the underlying (noiseless) protocol π bit-by-bit, where the scheme adds to each bit the parity (mod 2) of the corresponding round number.

[1] We use AMD codes [6] to generate signatures, see the full version of this paper [10, Appendix A] for the exact details.

The scheme works in *challenge-response*–style iterations: The first party (Alice) begins by sending the bit of the next round of π along with the parity of that round (*the challenge*). Bob receives this bit, and if the parity corresponds to the round number he expects, he records this bit and replies with the next bit of π along with the parity of that round in π (*the response*). When this reply reaches Alice, and the parity is correct, Alice records the bit from Bob and moves on to the next iteration. In any case of erasures or if the parity mismatches, the receiver ignores the received message. The analysis in [15] shows that this single parity bit suffices to keep track of the progress despite an unbounded amount of erasures.

However, in the UPEF model, a bit flip can either corrupt the content bit (i.e., the next simulated round of π) or the parity bit sent along! Corrupting the parity bit damages the correctness of the [15] scheme, but this is the only way the noise can affect correctness. Throughout a detailed case analysis, we prove that corrupting the parity bit has the sole effect of making the parties *out-of-sync*, in the sense that one party advances to the next round in π, while the other does not. Luckily, this type of out-of-sync corruption was already considered in the interactive-coding community, initiated by the work of Braverman, Gelles, Mao, and Ostrovsky [4], which presented a non-efficient scheme that withstands a noise level of up to 1/18 fraction of the rounds, where "noise" here means insertions and deletions producing out-of-sync events as described above. That work was followed by a work by Sherstov and Wu [27], who showed that a variant of the [4] scheme withstands the optimal level of noise, namely, up to 1/4 of the rounds, and by a work by Haeupler, Shahrasbi, and Vitercik [22], who presented an *efficient* scheme, based on synchronization strings, with noise resilience of 1/44.

Therefore, we can set the scheme in [22] (denoted HSV hereinafter) as the outer layer in our construction, and set the probabilities $\{p_i\}_{i\in\mathbb{N}}$ such that the total number of insertion and deletion errors will not surpass the threshold expected by the HSV scheme, except with an exponentially small probability. This construction achieves our goal of obtaining a coding scheme in the UPEF model, with optimal length of $O(N+T)$, and an exponentially small failure probability.

Unfortunately, once converting this optimal UPEF scheme back to the standard UF model, the overhead increases severely. In particular, the way we set the probabilities $\{p_i\}_{i\in\mathbb{N}}$ implies a logarithmic overhead on the size of the signatures, leading to a sub-optimal scheme of length $O((N+T)\log(N+T))$ in the UF model. To avoid this increase in communication, we must maintain the probabilities $\{p_i\}_{i\in\mathbb{N}}$ "large", and design a new scheme that is still optimal in the UPEF model despite the high values of $\{p_i\}_{i\in\mathbb{N}}$.

Obtaining an optimal scheme: the iterative approach. In order to obtain a UF-optimal scheme, we take a different approach, namely, we execute an increasing-length instances of a "standard" interactive coding [7]. As before, we start by constructing a coding scheme over the UPEF model. Our goal now is to maximize the p_i's as much as possible. The main idea is as follows. Let's fix each p_i to some constant, say, 2/3. Now, any corruption in the UPEF model will cause an erasure with a fixed probability of 1/3. The number of erasures a party observes is a good estimate of the level of noise during the same transmissions. Hence, the parties can continue running the scheme again and again, until they believe the noise level was low enough to produce the correct output.

In more detail, Alice and Bob run an efficient interactive coding scheme resilient to a constant fraction of adversarial flips (e.g., [2, 19]). After executing the scheme, Alice and Bob count the number of erasures observed during the execution and estimate (with high probability) the fraction of corruption they experienced. They communicate this estimate

to each other, and decide how to continue accordingly. If the noise level seems sufficiently low, the resilient scheme must have produced the correct output, and the parties can safely terminate. Otherwise, the parties re-run the interactive scheme, doubling its length. They repeat this action until they reach an execution where the noise level is low enough to guarantee the success of the underlying interactive coding scheme.

With a correct choice of parameters, this results in a UPEF scheme of length $O(N+T)$. However, since all $\{p_i\}_{i\in\mathbb{N}}$ are fixed to a constant, once we translate this scheme into a UF scheme, we keep its length up to a constant and obtain an optimal length of $O(N+T)$ in this case as well.

We note that a communication complexity of $\Theta(N+T)$ is tight for the UF model. A lower bound of $\Omega(N+T)$ is immediate by considering the case where the adversary corrupts the entirety of the communication between Alice and Bob for $\Theta(T)$ rounds, e.g., by flipping each bit with probability $1/2$, thereby not allowing any information to cross the channel during these rounds. After this corruption, N rounds are still needed to complete the protocol without noise.

1.2 Related Work

As mentioned above, the field of interactive coding was initiated by the work of Schulman [25, 26]. Following this work, many two-party interactive coding schemes were developed, with the goal to optimize various properties, such as efficiency, communication rate, and noise resilience [5, 18, 2, 23, 21, 13, 3, 14, 20]. Two-party coding schemes for different types of noise, such as erasures or insertions and deletions, appeared in [11, 13, 8, 4, 22, 27, 9, 20]. See [12] for an extensive survey on this field.

Closest to our work are coding schemes that withstand an unbounded amount of corruption. As mentioned above, Dani et al. [7] developed a randomized scheme that deals with an unbounded amount T of oblivious bit-flips, succeeds with high probability, and simulates any π of length N in $N+O(T+\sqrt{N(T+1)\log T})$ rounds. Gelles and Iyer [15] developed a deterministic scheme that deals with an unbounded amount T of (not necessarily oblivious) erasures in at most $2N+4T$ communication rounds. For the multiparty setting, Aggarwal, Dani, Hayes, and Saia [1] developed a coding scheme that correctly simulates any protocol over an arbitrary network withstanding an unbounded amount of oblivious corruptions in $\tilde{O}(N+T)$ rounds, suppressing logarithmic terms.

1.3 Paper Outline

We set up the UPEF and UF models, recall the insertion-deletion model, and review interactive coding protocols in Section 2. In Section 3 we describe an optimal UPEF coding scheme that follows a code concatenation approach. Its analysis is deferred to the full version of this paper [10, Appendix A]. Finally, in Section 4 we describe an optimal coding scheme in the UPEF model that follows an iterative approach. We then show how to translate it into an optimal UF coding scheme.

2 Preliminaries

Notations. For an integer $n\in\mathbb{N}$ we use $[n]=\{1,2,3,\ldots,n\}$. All logarithms are taken to base 2 unless otherwise mentioned. For two strings a,b we denote by $a\circ b$ their concatenation. We will use $\bigcirc_{k=1,2,\ldots,\ell}\ a_k \triangleq a_1\circ a_2\circ\cdots\circ a_\ell$ to abbreviate the concatenation of multiple strings. We use $O_\varepsilon(\cdot),\Theta_\varepsilon(\cdot)$, etc., to explicitly remind that the constant inside the $O(\cdot)$ may depend on (the constant) ε.

2.1 Interactive Protocols and Coding Schemes

Consider two parties, Alice and Bob, having inputs $x, y \in \{0,1\}^k$ respectively, who wish to compute some function $f(x,y)$ by communicating over a channel with alphabet Σ. Towards that goal, Alice and Bob use an *interactive protocol* composed of two algorithms $\pi = (\pi_a, \pi_b)$ for Alice and Bob, respectively. These algorithms assume a common clock known by both parties (i.e., the protocol is synchronized) and determine, for each party in each round (timestep), whether the party (1) has to send a message in that round, (2) which symbol the party sends, and (3) if the party should terminate in that round and which output should it give.

Each party records all the messages it receives during the execution of the protocol. The collection of these records is the party's *transcript*. We assume that π has *fixed order of speaking*; this means that in each round exactly one party is transmitting a symbol (the other party listens), and the identity of the transmitting party in a given round is predetermined and independent of the parties' inputs. In particular, a protocol in which Alice speaks in odd rounds, and Bob speaks in even rounds is said to be of an *alternating order*. Note that if π is not alternating, then it can be converted to an alternating-order protocol while increasing the communication complexity by a factor of at most 2. We say that a protocol is *k-alternating*, for some $k \in \mathbb{N}$, if during its execution each party transmits bulks of k bits. The *length* of a protocol is defined to be the number of rounds it includes until both parties have terminated.

Noisy channels and coding schemes. Now, assume that the parties are connected by a *noisy channel*. Formally, given an input and output alphabets $\Sigma_{in}, \Sigma_{out}$, respectively, a single utilization of a noisy channel is the (possibly randomized) function $C : \Sigma_{in} \to \Sigma_{out}$.

We can now discuss protocols that perform over noisy channels. We say that a protocol π' *simulates* π over the noisy channel C, if for any inputs (x,y), after executing π' over the noisy channel C, the parties can output their transcripts in an execution of π over a noiseless channel with inputs (x,y). When the channel noise or the algorithm π' are probabilistic, we say that π' simulates π *with probability* p if the probability that the parties' output equal the transcript of π is at least p, for any inputs pair.

A *coding scheme* (for some given noisy channel C) is a function CS, whose input is a noiseless protocol π, and its output is a protocol $\pi' = CS(\pi)$ which simulates π over the channel C. When the channel noise or the scheme are probabilistic, we say that the coding scheme has success probability p, if for any π, the protocol $\pi' = CS(\pi)$ simulates π with probability p.

2.2 Noise Models

As alluded to in the introduction, our scheme is designed to withstand an unbounded amount of (oblivious) bit flips. However, we design the scheme by reducing the unbounded-flip model to a different noise model with unbounded probabilistic erasures and flips. Furthermore, the effect of probabilistic erasures and flips noise on the inner layer of our coding scheme is such that the outer layer "sees" insertion and deletion noise. We will now define these three noise models in turn.

The Unbounded Flip noise model (UF). Our main noise model is the unbounded flip noise model, set forth by Dani et al. [7]. Given a specific execution of π' with inputs (x,y), the adversary sets a *noise corruption pattern* $E \subset \mathbb{N}$ such that the amount of noise, $|E|$, satisfies $|E| = T$ for some number $T \in \mathbb{N}$ decided by the adversary. The noise pattern can

be set as a function of π, x, y but is independent of any randomness the parties might have (i.e., an *oblivious* noise). The noise pattern E determines which bits get flipped during the execution of π. Namely, if $i \in E$, then the i-th transmitted bit in π will be flipped. Otherwise, the bit goes through uncorrupted. Note that T might be arbitrary. When one of the parties terminates, the channel sends zeros to the another party, which may be flipped by the adversary.

The Unbounded Probabilistic Erasure-Flip noise model (UPEF). Our coding scheme in this work is designed and analyzed within the following noise model, that combines both erasure and flip noise. This model naturally appears when executing a protocol in the UF model while each message contains a (probabilistic) signature or a message authentication tag that indicates its validity.

In this model, the parties are connected via a noisy communication channel $C : \{0,1\} \to \{0,1,\perp\}$, which can either flip bits or erase them (denoted by the erasure mark $\perp$). Similar to the UF model, given any specific execution of π' with inputs (x, y), the rounds which are corrupted are predetermined by an adversary that knows π', x, y and the inputs but not the parties' private randomness. This corruption is described via the noise pattern $E \subset \mathbb{N}$, where $i \in \mathbb{N}$ means that the i-th round is corrupted; otherwise, the bit arrives at the other side intact. When a round is corrupted, the effect is as follows: the bit is flipped with some probability p_i or is erased with probability $1 - p_i$. The probabilities $\{p_i\}_{i\in\mathbb{N}}$ are parameters of the model and will be specified later.

Terminating in the UPEF is different from terminating in the UF. When Alice terminates, the channel transmits a special "silence" symbol, namely, "$\square$". Upon the reception of this special symbol, Bob knows that Alice has quit, and terminates as well.

Similar to the UF model, we restrict the discussion to noise patterns in which the total number of corrupted rounds is finite. That is, there exists some number $T \in \mathbb{N}$, unknown to the parties and π, such that $|E| = T$.

The Insertion-Deletion noise model. The insertion-deletion noise model [4], which we briefly describe here, is important for our analysis of the concatenated coding scheme.

In this model we consider *alternating* interactive protocols π', where no common clock is assumed by the parties. Instead, Alice sends the first symbol (round 1), and Bob is idle until receiving this symbol. Once the first symbol is obtained by Bob he transmits a symbol back to Alice (round 2). Alice will execute round 3 once receiving this symbol, and so on. The noise is allowed to either corrupt a symbol (i.e., the receiver will obtain a different symbol from the one sent, *a substitution*), or to completely delete the symbol, so that the receiver receives nothing. In the latter case, the protocol is "stuck" as both parties await an incoming symbol to proceed. To avoid getting stuck, the noise must inject a new symbol towards the sender of the symbol that got deleted. This causes the parties to get *out of sync*, that is, one of them will believe the current round is i, while the other will believe the current round is $i + 2$. See also [4, 27, 22, 9, 16, 17].

In [22], the authors give an efficient constant-rate coding scheme for insertions and deletion noise, which we will use in our construction.

▶ **Theorem 2.1** (Theorem 1.2 in [22]). *For any alternating protocol π of length n and for any $\varepsilon > 0$, there exists an efficient randomized protocol π' simulating π in presence of $\delta = 1/44 - \varepsilon$ fraction of edit-corruptions, whose length is $\Theta_\varepsilon(n)$ and succeeds with probability $1 - 2^{-\Theta(n)}$. The alphabet size of π' is $\Theta_\varepsilon(1)$.*

We assume that at its termination, π' has an output, which equals to the output of π (under the conditions in the theorem).

3 A UPEF-optimal coding scheme via code concatenation

In this section we give an optimal UPEF coding scheme, based on code-concatenation approach (Algorithms 1 and 2). The analysis of the scheme, presented in the full version of this paper [10, Appendix A], proves the following Theorem.

▶ **Theorem 3.1.** *Given any two-party binary interactive protocol π of length N, there exists some constant C and an efficient randomized protocol Π of length $O(N+T)$ that simulates π with probability $1-2^{-\Omega(N)}$ over a binary channel in the presence of an arbitrary and a priori unknown number T of probabilistic Erasure–Flip corruptions, with $p_i = \min\left\{\frac{CN}{i^2}, \frac{1}{2}\right\}$.*

Let π be a binary alternating protocol that assumes noiseless channels of length $|\pi| = N$. Our goal is to simulate π in the UPEF model. Let π' be the randomized protocol obtained from π via Theorem 2.1, by setting $\delta = 1/45$ (i.e., $\varepsilon = 1/1980$). We denote $|\pi'| = N'$. Denote the alphabet of π' by Σ' and note that its size is constant, $|\Sigma'| = \Theta(1)$.

Our coding scheme (Algorithms 1 and 2) simulates the communication of π', symbol by symbol. As the channel in the unbounded probabilistic Erase–Flip noise model is binary, the parties communicate the binary representations of the symbols in π'. Therefore, during each iteration of the simulation, Alice sends a symbol to Bob using $\log|\Sigma'|$ bit transmissions, and expects a symbol reply from Bob.

Algorithm 1 Simulation over Erasure and Substitution Channel with Unbounded Noise (Alice).

Input: An alternating binary protocol π of length N, an input x
Initialize: Let π' be the protocol simulating π given by Theorem 2.1, setting $\varepsilon = 1/1980$. Let $N' = |\pi'|$ and assume Σ' (the alphabet of π') is a power of two.

A.1 $\mathcal{T}_a \leftarrow \emptyset, r_a \leftarrow 0$
A.2 **while** $r_a < \frac{N'}{2}$ **do**
A.3 `// Send Message`
A.4 $r_a \leftarrow r_a + 1$
A.5 $m_{send} \leftarrow \pi'(x \mid \mathcal{T}_a)$
A.6 $\mathcal{T}_a \leftarrow \mathcal{T}_a \circ m_{send}$
A.7 send $(m_{send}, r_a \mod 2)$ ▷ k transmissions
A.8
A.9 `// Receive Message`
A.10 receive $m' = (m_{rec}, r_{rec})$ ▷ k transmissions
A.11 **if** m' does not contain $\perp$ and $r_{rec} = r_a \mod 2$ **then**
A.12 $\mathcal{T}_a \leftarrow \mathcal{T}_a \circ m_{rec}$
A.13 **else**
A.14 delete the last symbol of $\mathcal{T}_a$
A.15 $r_a \leftarrow r_a - 1$
A.16 **end if**
A.17 **end while**
A.18 Output the output given by π'

The simulation of π' employs a challenge-response paradigm, where Alice sends a symbol (the challenge) and expects one back (the response). The parties maintain a counter to track their respective progress, namely, the variables r_a and r_b, which represent the number of successful iterations observed by Alice and Bob, respectively. Every time Alice and Bob send a symbol, they attach to it the parity (mod 2) of their own counter. Hence, the simulation is k-alternating, with $k = \lceil \log|\Sigma'| \rceil + 1$.

Algorithm 2 Simulation over Erasure and Substitution Channel with Unbounded Noise (Bob).

Input: An alternating binary protocol π of length N, an input x
Initialize: Let π' be the protocol simulating π given by Theorem 2.1, setting $\varepsilon = 1/1980$. Let $N' = |\pi'|$ and assume Σ' (the alphabet of π') is a power of two.

B.1 $\mathcal{T}_b \leftarrow \emptyset, r_b \leftarrow 0, err \leftarrow 0, m \leftarrow (0,0)$
B.2 **while** $m' \neq \square$ **do**
B.3 `// Receive Message`
B.4 receive $m' = (m_{rec}, r_{rec})$ ▷ k transmissions
B.5 **if** m' does not contain $\perp$ and $r_{rec} \neq r_b \mod 2$ **then**
B.6 $\mathcal{T}_b \leftarrow \mathcal{T}_b \circ m_{rec}$
B.7 $err \leftarrow 0$
B.8 **else**
B.9 $err \leftarrow 1$
B.10 **end if**
B.11
B.12 `// Send Message`
B.13 **if** $err = 0$ **then**
B.14 $r_b \leftarrow r_b + 1$
B.15 $m_{send} \leftarrow \pi'(y \mid \mathcal{T}_b)$
B.16 $\mathcal{T}_b \leftarrow \mathcal{T}_b \circ m_{send}$
B.17 send $m \leftarrow (m_{send}, r_b \mod 2)$ ▷ k transmissions
B.18 **else**
B.19 send m ▷ k transmissions; m from memory
B.20 **end if**
B.21 **end while**
B.22 Output the output given by π'

When Alice receives a symbol (as a response to the challenge she has previously sent), she checks the counter value attached to it: if it matches her expected counter parity (mod 2), she "believes" this challenge-response iteration, delivers the received symbol to π' and increases r_a by 1; otherwise, she ignores the reply and tries again in the next iteration.

Bob acts in an analogous manner: if the information received from Alice matches the counter parity (mod 2) he is expecting, then he "believes" the received symbol, delivers it to π', increases r_b by 1, obtains from π' the next symbol to communicate to Alice, and sends Alice this symbol and the parity of r_b. If the information from Alice does not match Bob's expectation, he ignores this transmission and replies with the previous symbol computed by π' (along with the parity that corresponds to that symbol). When a party "believes" an iteration, it appends the received and transmitted symbols of the iteration to its transcript, $\mathcal{T}_a$ or $\mathcal{T}_b$, respectively. This transcript records all the symbols communicated so far (by π') during the "successful" iterations of Algorithms 1 and 2.

To summarize, in each iteration of the loop, Alice generates the next message of π', denoted $m \in \Sigma'$, based on her current transcript $\mathcal{T}_a$ and her input x, i.e., $m = \pi'(x \mid \mathcal{T}_a)$. Alice (temporarily) adds m to $\mathcal{T}_a$, and sends its binary representation to Bob, along with the parity of r_a. After receiving a k-bit message (m_{rec}, r_{rec}) from Alice, Bob checks that none of the k bits have been erased (denoted by $\perp$) and that r_{rec} is opposite to his parity (since Alice added a new symbol and he did not, yet). If everything matches, Bob adds m_{rec} to his transcript $\mathcal{T}_b$, increases r_b by 1, computes the next message $m' = \pi'(y \mid \mathcal{T}_b)$ and the new parity of r_b, and transmits them to Alice. On the other hand, if Bob notices any erasures or the mismatch of the parity, he ignores Alice's new symbol and replies with the latest computed (m', r_b) recorded in his memory.

At the end of the iteration, Alice receives a message and a parity from Bob; if there were no erasures and the received parity matches r_a, she adds that message to her transcript. Otherwise, Alice deletes the (temporary) message she added at the beginning of this iteration.

The algorithm ends once the length of $\mathcal{T}_a$ reaches the length of π' (for Alice) or when a special symbol $\square$, sent by the channel when Alice quits, is received by Bob. We discuss termination in this model and the implication of assuming this special symbol in the full version of this paper [10, Appendix B.1].

The complete detailed analysis of the coding scheme (Algorithms 1 and 2) and the proof of Theorem 3.1, are deferred to the full version of this paper [10, Appendix A].

4 A UF Scheme with Optimal Communication

The concatenated scheme in Section 3, while optimal in the UPEF model, implies a UF scheme of length $O((N+T)\log(N+T))$ in which the i-th bit is replaced with an AMD code [6] of length $O(\log(p_i^{-1}))$; see the full version of this paper [10, Appendix B] for details. In this section, we take a different approach towards constructing an optimal coding scheme in the UPEF model, namely by executing increasing-length coding schemes in an iterative fashion. This approach eventually leads to an optimal-communication UF scheme.

Here, Alice and Bob utilize a "standard" interactive coding scheme for substitutions and estimate the experienced level of noise. If the estimated noise level is too high, Alice and Bob repeat the execution with a larger amount of redundancy. When the noise level is low enough, the interactive scheme guarantees the success of the computation, and the parties can terminate. This approach, in addition to leading to a UF scheme with optimal communication, is also much simpler and easier to analyze than the scheme of Section 3. The key difference is that, all we need in order to estimate the noise level well, is that the probabilities $\{p_i\}_{i\in\mathbb{N}}$ are bounded below by a constant, rather than converging to 0. This aligns perfectly with obtaining optimal complexity, as smaller $\{p_i\}_{i\in\mathbb{N}}$ imply longer encodings.

The scheme in this section utilizes a slight variant of the UPEF model, denoted the modified UPEF (mUPEF) model, which we now describe. Let a noise pattern $E \subset \mathbb{N}$ be determined adversarially. As before, if $i \notin E$, the i-th transmitted bit reaches the other party intact. For $i \in E$, the i-th transmitted bit is still erased with probability $1-p_i$, and corrupted with probability p_i. However, the corruption here is not necessarily a bit flip as in the original UPEF. Instead, the adversary determines whether the bit is flipped, erased, or not corrupted at all. The probabilities $\{p_i\}_{i\in\mathbb{N}}$ are parameters of the model. However, we will actually set them all to have the same value. That is, $\forall i,\ p_i = 2/3$.

In our scheme, we set $p_e = 1 - p_i = 1/3$ as a lower bound on the probability that a bit is erased. Similar to the UF model, we will assume that when Alice terminates, the channel implicitly sends Bob a default symbol (e.g., a zero).

In the following sections we describe and analyze a coding scheme in the mUPEF model, with optimal $O(N+T)$ communication that, due to our choice of $p_e = 2/3$, results with a UF scheme with $O(N+T)$ communication as well.

An optimal mUPEF Coding Scheme. For the underlying substitution-resilient interactive coding scheme, we can take any (efficient) 2-party scheme with binary alphabet that is resilient to a constant fraction of adversarial noise, e.g., [2, 19]. In particular, let us assume an interactive coding scheme that simulates any (noiseless) protocol π in the presence of up to 0.1 adversarial substitutions with a constant rate over the binary alphabet. Denote the substitution-resilient version by π', so $|\pi'| = O(|\pi|) = O(N)$. We assume that π' is

alternating. We additionally assume that the communication of π' includes a constant fraction of ones. To be concrete, out of the $|\pi'|/2$ bits Alice sends, at least $|\pi'|/8$ are the bit 1, in any execution of π'. We must have this property, because in our scheme, a long sequence of zeros will indicate termination. For that reason, we want that π' will not send a long sequence of zeros. This can be achieved, for instance by making the parties communicate a 1 every alternate round, or by means of randomization (see, e.g., [13]).

We proceed to describe our mUPEF resilient scheme Π simulating the substitution-resilient π' defined above. The execution of Π consists of iterations, where the i-th iteration, $i = 0, 1, 2, \ldots$, takes $2L_i$ rounds with $L_i = |\pi'|2^i$. The i-th iteration can be broken down into two parts, each of length L_i. In the first part, the parties execute π' from scratch, padded to length L_i; in this padded protocol, each bit of π' is sent 2^i times, and decoding is performed by majority (defaulting to 0 on ties, considering erased copies as zeros). In the second part of iteration i, only Bob speaks. He sends Alice the *success string* 0^{L_i} if and only if he observed less than $0.001p_eL_i$ erasures in the first part; otherwise Bob sends the *error string* 1^{L_i}.

Alice terminates at the end of iteration i if she observed less than $0.001p_eL_i$ erasures in each of the parts of iteration i, and Bob's transmissions at the second part contains more 0's than 1's (i.e., it decodes to the success string rather than to the error string). Alice gives as an output the same output that π' has generated in the iteration in which she terminated. Bob terminates at the end of iteration j if he observed less than $0.001p_eL_j$ erasures in the first part of j and at most $0.001p_eL_j$ of the received bits in the first part of j are 1's. Bob gives as an output the output of the latest iteration k, with $k < j$, in which (1) he observed less than $0.001p_eL_j$ erasures in the first part and (2) he received at least $L_k/40$ ones from Alice in the first part. We call *a valid iteration* any iteration that satisfies these two conditions.

4.1 Analysis

In this section we prove the following theorem.

▶ **Theorem 4.1.** *Given any two-party binary interactive protocol π of length N, there exists an efficient randomized protocol Π of length $O(N + T)$ that simulates π with probability $1 - 2^{-\Omega(N)}$ over a binary channel in the presence of an arbitrary and a priori unknown number T of mUPEF corruptions, with $p_i = 2/3$ for all i.*

We start with proving the correctness of our coding scheme: we begin by demonstrating that Alice's output is correct with high probability. Additionally, we show that Bob's output at Alice's termination is also correct. Then, we prove that Bob terminates after Alice, that Alice terminates in a *valid* iteration, and that there are no valid iterations afterwards. This would imply that Bob gives the right output as well.

Recall that π' is resilient to 0.1-fraction of substitutions. In addition, recall the padding mechanism; in order to cause a bit substitution in π' in some iteration i, at least a half of its 2^i transmitted copies must be flipped or erased. Thus, if there are less than $L_i/20$ corruptions during the first part of iteration i (that is, indices that the adversary puts in E), then π' must give the correct output at the end of iteration i, for both Alice and Bob.

In the following lemma, we show that if there are *more* than $L_i/20$ corruptions in some iteration i, then Alice continues to executing iteration $i + 1$ and does not terminate at the end of iteration i, except with a negligible probability of $2^{-\Omega_{p_e}(L_i)}$.

▶ **Lemma 4.2.** *Assume that in the first part of iteration i there are $c_i \geq L_i/20$ corruptions. Then, the probability that Alice terminates at the end of iteration i is $2^{-\Omega_{p_e}(L_i)}$.*

Proof. We divide the proof into two separate cases: (1) each of Alice and Bob observes less than $0.001p_eL_i$ erasures in the first part, and (2) Bob observes more than $0.001p_eL_i$ erasures but Alice does not. Of course, if more than $0.001p_eL_i$ erasures are observed by Alice during the first part, she does not terminate by definition.

First, we find the probability of case (1) to occur. Denote by E_i the subset of the noise pattern E containing only rounds in the first part of iteration i. Then, $|E_i| \geq c_i \geq L_i/20$. Let e' be the number of rounds during the first part of iteration i that were erased because the event of channel erasure, which happens with probability p_e, occurred. It holds that $\mathbb{E}[e'] = |E_i|p_e \geq 0.05p_eL_i$, and by Chernoff's inequality (Theorem 4.4(2) in [24]),

$$\Pr(e' < 0.002p_eL_i) \leq \Pr(e' < 0.04\mathbb{E}[e']) \leq e^{-\mathbb{E}[e']\frac{0.96^2}{2}} \leq e^{-0.05p_eL_i\frac{0.96^2}{2}} \in 2^{-\Omega_{p_e}(L_i)}.$$

Thus, if $c_i \geq L_i/20$, then $e' < 0.002p_eL_i$ with probability $2^{-\Omega_{p_e}(L_i)}$, which bounds the probability of case (1) to occur.

In case (2), Bob sees a lot of erasures and sends the error string. In order for Alice to terminate, it is necessary for her to decode this message as the success string. For this to happen, it is necessary that the adversary corrupts at least $L_i/2$ bits during the second part of the i-th iteration. Similar to the proof of case (1), the adversary succeeds to corrupt so many bits while causing less than $0.001p_eL_i$ erasures during the second part with probability of at most $2^{-\Omega_{p_e}(L_i)}$. We conclude that Alice terminates at the end of iteration i with probability of at most $2^{-\Omega_{p_e}(L_i)}$. ◀

The above lemma indicates that, in the event of an excessive number of corruptions, Alice will not terminate. The following observation complements this idea and states that if Alice does terminate, the computation of π' is successful with high probability, hence, her output in Π is correct.

▶ **Observation 4.3.** *When Alice terminates, π' gives the correct output, with probability $1 - 2^{-\Omega_{p_e}(N)}$.*

Proof. If during a given iteration there were less than $L_i/20$ corruptions, then the resilience of π' guarantees that it gives the right output, and Alice terminates. The event in which Alice terminates and provides incorrect output can only occur when there are more corruptions in a specific iteration, the probability of which was constrained in Lemma 4.2. A union bound over all possible iterations (while recalling that $L_i = |\pi'|2^i$) bounds the probability for Alice to give an incorrect output, by

$$\sum_{i=0}^{\infty} 2^{-\Omega_{p_e}(L_i)} = 2^{-\Omega_{p_e}(|\pi'|)} = 2^{-\Omega_{p_e}(N)}.$$ ◀

Next, we prove that Bob terminates only after Alice has already terminated, with high probability.

▶ **Lemma 4.4.** *Consider an iteration i in which Alice has not yet terminated. Then, the probability that Bob terminates at the end of iteration i is at most $2^{-\Omega_{p_e}(L_i)}$.*

Proof. In order to terminate at the end of iteration i, Bob must observe less than $0.001p_eL_i$ erasures in the first part of the iteration. In a manner analogous to the argument presented in Lemma 4.2, it can be shown that, with a probability approaching $1 - 2^{-\Omega_{p_e}(L_i)}$, there will be a total of less than $L_i/20$ corrupted messages received by Bob during the first part of iteration i. Recall that as long as Alice has not terminated, at least $L_i/8$ out of the $L_i/2$

bits she sends in iteration i are ones. Consider these transmissions only. Even if all the corruptions during the first part of i occur during these transmissions, then Bob will observe at least $0.001p_eL_i$ ones in the first part of i, and will therefore not terminate by definition. Consequently, Bob terminates in iteration i with probability $2^{-\Omega_{p_e}(L_i)}$. ◀

Bob's output is the output of π' in the last *valid* iteration prior to his termination iteration. The following lemma shows that, with high probability, the last valid iteration is the same iteration in which Alice has terminated. By Observation 4.3, π' gives the correct output in that iteration.

▶ **Lemma 4.5.** *Denote by i the iteration in which Alice terminates. Then, i is a valid iteration with probability $1-2^{-\Omega_{p_e}(L_i)}$. Further, the probability that there is a valid iteration $j>i$ is $2^{-\Omega_{p_e}(N)}$.*

Proof. In order to prove that i is a valid iteration, we have to show that Bob observes less than $0.001p_eL_i$ erasures in the first part of i, and that he receives at least $L_i/40$ ones from Alice in the first part.

First, note that Bob observes less than $0.001p_eL_i$ erasures in the first part of i if and only if he sends the success string in the second part. We demonstrate that with probability $1-2^{-\Omega_{p_e}(L_i)}$ this is the case. As illustrated in case (2) of Lemma 4.2, when Bob transmits the error string to Alice in the second part of i, Alice terminates with probability $2^{-\Omega_{p_e}(L_i)}$. Since Alice terminates in i, there is a probability of $2^{-\Omega_{p_e}(L_i)}$ that Bob sends the error string in the second part of i and observes more than $0.001p_eL_i$ erasures in the first part of i.

We proceed to show that Bob receives at least $L_i/40$ ones from Alice in the first part of i. By Lemma 4.2, during the first part of iteration i there are less than $L_i/20$ corruptions with probability $1-2^{-\Omega_{p_e}(L_i)}$. Additionally, Alice always sends at least $L_i/8$ ones in the first part. Thus, Bob receives at least $L_i/40$ ones during the first part of i with probability $1-2^{-\Omega_{p_e}(L_i)}$, and this is the probability of i to be a valid iteration.

Let j be an iteration such that $j>i$. Recall that after Alice terminates the channel sends Bob zeros, by default. We show that j is not a valid iteration with high probability, by dividing into two cases. If there are at least $L_j/40$ flips in the transmissions towards Bob during the first part of iteration j, then with probability $1-2^{-\Omega_{p_e}(L_j)}$ Bob observes at least $0.001p_eL_j$ erasures, similar to case (1) in Lemma 4.2, thus j is not valid. If there are less than $L_j/40$ flips in these transmissions, then Bob receives less than $L_j/40$ ones and j is not a valid iteration either. Thus, iteration j is valid with probability of at most $2^{-\Omega_{p_e}(L_j)}$. Since $L_j=|\pi'|2^j$, applying a union bound on all the iterations gives a probability of $\sum_{j=i}^{\infty}2^{-\Omega_{p_e}(L_j)}=2^{-\Omega_{p_e}(N)}$ to the event that there is a valid iteration after i. ◀

We may use a union bound to conclude the correctness part for Bob. The overall probability of Bob to terminate after Alice is $1-2^{-\Omega_{p_e}(N)}$. The probability of Bob to declare Alice's termination iteration as valid is $1-2^{-\Omega_{p_e}(N)}$. Bob gives the correct output at this iteration with probability $1-2^{-\Omega_{p_e}(N)}$ by Lemma 4.2, and this iteration is the last valid iteration with probability $1-2^{-\Omega_{p_e}(N)}$. We may use the inclusion-exclusion principle to show that the probability of the intersection of all these four events is $1-2^{-\Omega_{p_e}(N)}$. Recall that for any A,B it holds that $1\geq P(A\cup B)=P(A)+P(B)-P(A\cap B)$. Then, the fact that $P(A),P(B)\in 1-2^{-\Omega_{p_e}(N)}$ implies that $P(A\cap B)\geq 2(1-2^{-\Omega_{p_e}(N)})-1\in 1-2^{-\Omega_{p_e}(N)}$. Thus, the probability of the event in which Bob gives the correct output at his termination is $1-2^{-\Omega_{p_e}(N)}$, and we have completed the correctness part of Theorem 4.1.

The communication complexity of Π is $\sum_{i=1}^{i_B}2L_i$, with i_B being the iteration in which Bob terminates. At any iteration i before Alice terminates, the adversary has to select at least $0.001p_eL_i$ bits to corrupt in order to prevent Alice from terminating. In addition,

at any iteration i after Alice's and before Bob's terminations, the adversary has to select at least $0.001p_eL_i$ bits to corrupt in order to prevent Bob from terminating. Denote the iteration in which Alice terminates by $i_A < i_B$. Then, $\sum_{i=1,i\neq i_A}^{i_B-1} 0.001p_eL_i < T$, so $\sum_{i=1,i\neq i_A}^{i_B-1} 2L_i < 2000/p_e \times T$. Since L_i satisfies $L_i = 2L_{i-1}$ for all i, and since L_{i_A-1}, L_{i_B-1} are included in $\sum_{i=1,i\neq i_A}^{i_B-1} 2L_i < 2000/p_e \times T$, then $\sum_{i=1}^{i_B} 2L_i < 6000/p_e \times T$ and the communication complexity is $O_{p_e}(N+T)$.

4.2 Obtaining a UF-optimal coding scheme

In this section we construct a UF-model scheme based on the mUPEF protocol Π, while maintaining a communication complexity of $O(N+T)$. Recall, we set $p_e = 1/3$. This means that adversarial corruption in the i-th transmission of Π, becomes detectable (an erasure) with probability at least $1/3$. We would like to simulate this property in the UF model.

Towards this goal, we independently encode each bit of Π using a random code of length 5. In particular, we encode a 0 to one of $\{00000, 10000, 01000\}$ with equal probability, and encode a 1 to one of $\{00100, 10010, 01001\}$ with equal probability. A received 5-bit word is decoded to a 0 or a 1 only if it belongs to respective set, or otherwise it is considered as an erasure. It can easily be seen that any pattern of 1 to 5 bit-flips decodes to an erasure with probability at least $1/3$ as desired. We have thus inflated the scheme by only a constant factor. The bit complexity of the resulting UF scheme is then $5 \cdot |\Pi| = O(N+T)$.

References

1 Abhinav Aggarwal, Varsha Dani, Thomas P. Hayes, and Jared Saia. A scalable algorithm for multiparty interactive communication with private channels. In *ICDCN 2020: 21st International Conference on Distributed Computing and Networking, Kolkata, India, January 4-7, 2020*, ICDCN '20, pages 8:1–8:15, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3369740.3369771.

2 Zvika Brakerski, Yael T. Kalai, and Moni Naor. Fast interactive coding against adversarial noise. *J. ACM*, 61(6):35:1–35:30, December 2014. doi:10.1145/2661628.

3 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. doi:10.1137/141002001.

4 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, October 2017. doi:10.1109/TIT.2017.2734881.

5 Mark Braverman and Anup Rao. Toward coding for maximum errors in interactive communication. *IEEE Transactions on Information Theory*, 60(11):7248–7255, November 2014. doi:10.1109/TIT.2014.2353994.

6 Ronald Cramer, Yevgeniy Dodis, Serge Fehr, Carles Padró, and Daniel Wichs. Detection of algebraic manipulation with applications to robust secret sharing and fuzzy extractors. In Nigel P. Smart, editor, *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*, volume 4965, pages 471–488, Berlin, Heidelberg, 2008. Springer. doi:10.1007/978-3-540-78967-3_27.

7 Varsha Dani, Thomas P. Hayes, Mahnush Movahedi, Jared Saia, and Maxwell Young. Interactive communication with unknown noise rate. *Information and Computation*, 261:464–486, 2018. doi:10.1016/j.ic.2018.02.018.

8 Klim Efremenko, Ran Gelles, and Bernhard Haeupler. Maximal noise in interactive communication over erasure channels and channels with feedback. *IEEE Transactions on Information Theory*, 62(8):4575–4588, August 2016. doi:10.1109/TIT.2016.2582176.

9 Klim Efremenko, Elad Haramaty, and Yael Tauman Kalai. Interactive coding with constant round and communication blowup. In Thomas Vidick, editor, *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*, volume 151 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 7:1–7:34, Dagstuhl, Germany, 2020. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ITCS.2020.7.

10 Eden Fargion, Ran Gelles, and Meghal Gupta. Interactive coding with unbounded noise, 2024. arXiv:2407.09463.

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, January 2015. doi:10.1109/TIT.2014.2367094.

12 Ran Gelles. Coding for interactive communication: A survey. *Foundations and Trends® in Theoretical Computer Science*, 13(1-2):1–157, 2017. doi:10.1561/0400000079.

13 Ran Gelles and Bernhard Haeupler. Capacity of interactive communication over erasure channels and channels with feedback. *SIAM Journal on Computing*, 46(4):1449–1472, 2017. doi:10.1137/15M1052202.

14 Ran Gelles, Bernhard Haeupler, Gillat Kol, Noga Ron-Zewi, and Avi Wigderson. Explicit capacity approaching coding for interactive communication. *IEEE Transactions on Information Theory*, 64(10):6546–6560, October 2018. doi:10.1109/TIT.2018.2829764.

15 Ran Gelles and Siddharth Iyer. Interactive coding resilient to an unknown number of erasures. In *23rd International Conference on Principles of Distributed Systems, OPODIS 2019, December 17-19, 2019, Neuchâtel, Switzerland*, volume 153 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 13:1–13:16, 2019. doi:10.4230/LIPIcs.OPODIS.2019.13.

16 Ran Gelles, Yael Tauman Kalai, and Govind Ramnarayan. Efficient multiparty interactive coding – part I: oblivious insertions, deletions and substitutions. *IEEE Transactions on Information Theory*, 67(6):3411–3437, 2021. doi:10.1109/TIT.2021.3066009.

17 Ran Gelles, Yael Tauman Kalai, and Govind Ramnarayan. Efficient multiparty interactive coding - part II: non-oblivious noise. *IEEE Transactions on Information Theory*, 68(7):4723–4749, 2022. doi:10.1109/TIT.2022.3160515.

18 Ran Gelles, Ankur Moitra, and Amit Sahai. Efficient coding for interactive communication. *IEEE Transactions on Information Theory*, 60(3):1899–1913, March 2014. doi:10.1109/TIT.2013.2294186.

19 Mohsen Ghaffari and Bernhard Haeupler. Optimal error rates for interactive coding II: efficiency and list decoding. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 394–403. IEEE Computer Society, 2014. doi:10.1109/FOCS.2014.49.

20 Meghal Gupta and Rachel Yun Zhang. The optimal error resilience of interactive communication over binary channels. In *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, STOC 2022, pages 948–961, 2022. doi:10.1145/3519935.3519985.

21 Bernhard Haeupler. Interactive channel capacity revisited. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 226–235, 2014. doi:10.1109/FOCS.2014.32.

22 Bernhard Haeupler, Amirbehshad Shahrasbi, and Ellen Vitercik. Synchronization strings: Channel simulations and interactive coding for insertions and deletions. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, volume 107 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 75:1–75:14, 2018. doi:10.4230/LIPIcs.ICALP.2018.75.

23 Gillat Kol and Ran Raz. Interactive channel capacity. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 715–724, 2013. doi:10.1145/2488608.2488699.

24 Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomization and Probabilistic Techniques in Algorithms and Data Analysis*. Cambridge University Press, USA, 2nd edition, 2017.

25 Leonard J. Schulman. Communication on noisy channels: A coding theorem for computation. In *33rd Annual Symposium on Foundations of Computer Science, Pittsburgh, Pennsylvania, USA, 24-27 October 1992*, pages 724–733, Los Alamitos, CA, USA, 1992. IEEE Computer Society. doi:10.1109/SFCS.1992.267778.

26 Leonard J. Schulman. Coding for interactive communication. *IEEE Transactions on Information Theory*, 42(6):1745–1756, November 1996. doi:10.1109/18.556671.

27 Alexander A. Sherstov and Pei Wu. Optimal interactive coding for insertions, deletions, and substitutions. *IEEE Transactions on Information Theory*, 65(10):5971–6000, October 2019. doi:10.1109/TIT.2019.2916927.