Formulas Resilient to Short-Circuit Errors

Yael Tauman Kalai [*] Allison Lewko[†] Anup Rao[‡]

Abstract

We show how to efficiently convert any boolean formula F into a boolean formula E that is resilient to *short-circuit* errors (as introduced by Kleitman et al. [KLM94]). A gate has a short-circuit error when the value it computes is replaced by the value of one of its inputs.

We guarantee that E computes the same function as F, as long as at most $(1/10 - \varepsilon)$ of the gates on each path from the output to an input have been corrupted in E. The corruptions may be chosen adversarially, and may depend on the formula E and even on the input. We obtain our result by extending the Karchmer-Wigderson connection between formulas and communication protocols to the setting of adversarial error. This enables us to obtain error-resilient formulas from error-resilient communication protocols.

[*]Microsoft Research, yael@microsoft.com.

[†]Microsoft Research, allew@microsoft.com.

[‡]University of Washington, anuprao@cs.washington.edu. Supported by the National Science Foundation under agreement CCF-1016565.

1 Introduction

In this work, we consider the problem of constructing boolean circuits that are resilient to errors. We seek to (efficiently) convert any circuit C into an error resilient version E that computes the same function as C even if some of E's gates have errors. This is a fundamental question that has received much attention over the years, starting with the seminal work of Von Neumann in 1956 [vN56].

In this work, we consider *short-circuit* errors, which were first studied by Kleitman, Leighton, and Ma [KLM94]. A gate g has a short-circuit error if the value it computes is replaced by the value of some input to g. We consider the setting where these errors are adversarial and are even allowed to depend on the input to the circuit. This is equivalent to an error model where each gate can be replaced by an arbitrary function g, such that $g(0,0) = 0$ and $g(1,1) = 1$. For example, in this error model one may switch any $\wedge$ gate with an $\vee$ gate, and vice versa. Such errors can always be simulated by replacing the output of the gate with the value of one of its inputs in an adversarial way.

Kleitman *et. al.* show that for any error bound e, one can convert a circuit C of size $|C|$ into a circuit E of size $|E|$ that is resilient to e short-circuit errors, such that $|E| \leq O(e \cdot |C| + e^{\log_2 3})$. Their construction involves computing $O(e)$ copies of C in parallel (so that most of these copies must have 0 errors), and then taking a majority of the outputs in a clever way. Thus, they reduce to the problem of computing the majority in a way that is robust to errors. Kleitman *et. al.* also give a negative result, showing that to compute the $\wedge$ function and withstand e short-circuits, one needs a circuit of size at least $\Omega\left(\frac{e \log e}{\log \log e}\right)$. This suggests that one cannot tolerate a constant error rate in this model.

Our main result is the following theorem:

Theorem 1. *For every $\epsilon > 0$, there is a polynomial time computable function that converts any boolean formula F of size s into a boolean formula E of size $\mathsf{poly}(s)$, such that E computes the same function as F as long as at most a $\left(\frac{1}{10} - \epsilon\right)$ fraction of the gates on every input to output path in E experience short-circuit errors.*

We emphasize that Theorem 1 constructs formulas that can be resilient to errors almost everywhere, as long as there are less than 1/10 errors on each path from the root to a leaf. Moreover, the location of these errors may *depend* on the inputs that are fed into the formula. In the case of *random* (short-circuit) errors, Theorem 1 gives formulas that are resilient to a *constant* fraction of random errors with probability of failure that is polynomially small in the *size* of the formula. Indeed, our final error resilient formula will be of depth $O(\log s)$, and if the probability of any gate failing is a small enough constant, the probability that any input-output path experiences too many errors can be made to be polynomially small in s. In contrast, Kleitman *et. al.* [KLM94] construct circuits that are resilient to random errors that occur with rate $O(1/s)$. Note that in the short-circuit error model, it is impossible to be resilient against the corruption of an *entire* path from the root to a leaf, since otherwise the adversary can force the output to equal the value of the corresponding leaf. In this sense, Theorem 1 is optimal, up to the constant 1/10.

The short-circuit error model is different from the standard von Neumann error model, which assumes that the value of each wire in the circuit is flipped independently with some small probability p. These two error models are incomparable. The short-circuit model is weaker in the sense that if all the wires to a gate have the same value, then the gate will produce the same value, even if it is faulty. On the other hand, our model is stronger since it tolerates worst-case faults, whereas the von Neumann model cannot tolerate even a single worst-case fault. This is because the output wire could be faulty, which means that the circuit will be faulty.

Moreover, when the errors occur with probability $p \ll 1/10$, our work gives formulas that fail with probability at most $1/s^{\Omega(1)}$, in contrast to results in von Neumann's model, which give formulas that fail with probability at least p. It is impossible to build fault-tolerant circuits with many outputs in the von Neumann model (making it of dubious value in practice), since for circuits which output m bits, the output of a faulty circuit is correct only with probability $(1-p)^m$. On the other hand, our work gives formulas that are correct with high probability even if there are many outputs. In fact, to the best of our knowledge, none of the previous works could tolerate faults on a set of gates that disconnects the output from the inputs, and our results give a way to tolerate significantly more than this.

In summary, like the von Neumann model, the short-circuit model is restrictive. However, the short-circuit model allows for adversarial faults, and is correct even if the output is large. We refer the reader to [KLM94] for a more elaborate comparison between the two models.

As one might suspect, our error resilient formulas must somehow distribute the computation of the original formula in a *global* way, rather than relying on *local* steps to recover from corruptions (in contrast to prior work), and indeed, our techniques result in formulas that do exactly this. As we elaborate below, the key conceptual idea is to reduce the problem of correcting errors in computation to the (easier) problem of correcting errors in *communication*. We build on the Karchmer-Wigderson [KW88] connection between formulas and communication protocols, and show that this connection can be extended to the setting of adversarial error.

1.1 Our Results

Our main results are the following.

Theorem 2. *There is a polynomial time computable function mapping any balanced formula F of depth d, with gates of fan-in 2, into a formula E of depth $4d+10e$ with gates of fan-in 2, such that E computes $F(x)$ even if up to e of the $\wedge$ gates and e of the $\vee$ gates on every path from input to output in E experience short-circuit errors.*

Note that by setting $e = \frac{4d}{\epsilon}$, the resulting error resilient formula E is of depth at most $4d(\frac{10}{\epsilon}+1)$, and corrupting up to

$$\frac{e}{4d(\frac{10}{\epsilon}+1)} \geq \frac{1}{10+\epsilon} \geq \frac{1}{10} - \epsilon$$

fraction of the gates on any input to output path does not affect the output. Since every formula can be balanced (Lemma 4), one can balance an input formula and then apply Theorem 2 to obtain Theorem 1.

The above constant $1/10$, which bounds the fraction of errors on every path, can be improved to $1/6$ if we allow the error-resilient formula E to have gates with fan-in 3.

Theorem 3. *There is a polynomial time computable function mapping any balanced formula F of depth d, with gates of fan-in 2, into a formula E of depth $2d+6e$ with gates of fan-in 3, such that E computes $F(x)$ even if up to e of the $\wedge$ gates and e of the $\vee$ gates on every path from input to output in E experience short-circuit errors.*

Similarly to the above, by setting $e = \frac{2d}{\epsilon}$, the resulting error resilient formula E is of depth at most $2d(\frac{6}{\epsilon}+1)$, and corrupting up to

$$\frac{e}{2d(\frac{6}{\epsilon}+1)} \geq \frac{1}{6+\epsilon} \geq \frac{1}{6} - \epsilon$$

fraction of the gates on any input to output path does not affect the output.

1.2 Techniques

It is instructive to start with some toy cases. First observe that it is no loss of generality to assume that F only has $\wedge$ and $\vee$ gates, since by DeMorgan's rule we can always assume that all negations are at the variables, and then we may replace each input literal by a new variable. An error resilient version of the resulting monotone formula easily gives an error resilient version of the original formula by substituting the literals back in.

Suppose we wish to compute the formula $x \wedge y$, where x, y are bits. Let E be a balanced depth e tree of $\wedge$ gates, where we assign the input gates of E to x or y as follows. Each input gate of E corresponds to an address string $z \in \{L, R\}^e$, which defines the path from the output gate to the input gate. If the majority of this string is L, set the corresponding input to x, else set it to y (breaking ties arbitrarily). We claim that this circuit can tolerate up to $e/2-1$ corruptions on each root to leaf path. We briefly sketch the argument. Let E' be any circuit obtained from E by doing $\leq e/2-1$ corruptions. The interesting case is when $x = 0, y = 1$. Consider the input to output path that always picks the child corresponding to L in E', unless the corresponding gate has a short-circuit, in which case it follows the input corresponding to the error. Since this path has at most $e/2-1$ corruptions, it must end at an input labeled x, and then one can argue inductively that every gate on this path must evaluate to 0, which proves that the output gate must also evaluate to 0.

This idea can be easily generalized to handle the $\wedge$ of 2^d variables, each identified by a unique d-bit string. We use an error correcting code of a constant sized alphabet $[k]$ that encodes d-bit strings into $n = O(d)$ length strings over $[k]$ such that two valid encodings disagree in at least $(1-\epsilon)$ fraction of the coordinates. We build a depth n tree of $\wedge$ gates of fan-in k, and label every input gate that corresponds to a string at distance at most $(1/2-\epsilon/2)$ from a codeword z by the variable x_z that corresponds to that codeword. All other input gates are labeled arbitrarily. As above, we can argue that even if $(1/2-\epsilon/2)$ fraction of all the gates on any root to leaf path are short-circuited, the circuit will still compute the correct value.

If the formula contains both $\wedge$ and $\vee$ gates, we need a more general approach. We appeal to a connection, first observed by Karchmer and Wigderson [KW88], between formulas computing a boolean function f and communication protocols that solve a game related to f. In the game, Alice is given $x \in f^{-1}(0)$ and Omar is given $y \in f^{-1}(1)$. The two parties communicate using a protocol to compute an index i such that $x_i \neq y_i$. It can be shown that the minimum communication required in the game is exactly equal to the minimum depth of a formula computing f. In this work, we generalize this connection to show that error resilient protocols that solve the game can be used to obtain error resilient formulas that compute the function (see Section 4).

The problem of designing error resilient interactive communication protocols (as opposed to messages) was first studied by Schulman [Sch93], with extensions by [BR11, Bra12, GMS11]. However, in the current paper, we do not require the more complicated ideas of these works, because it turns out that we can obtain our results using simpler protocols that are enough for the error model we consider. Tree code based constructions may be useful to improve the parameters of our construction slightly (one might hope for an error tolerance of $(1/8-\epsilon)$ using the results of [BR11]), but at the additional cost of making the construction more involved. In addition, it is not clear to us that the resulting transformation on formulas will be computable in polynomial time.

The high level outline of our approach is the following. Given a formula F:

1. Apply the Karchmer-Wigderson transformation to F to get a communication protocol for the game (see Section 2 for more details).

2. Convert the protocol into an error resilient version, by proving an error-resilient version of

the Karchmer-Wigderson transformation (see Sections 4 and 5).

3. Apply the error-resilient Karchmer-Wigderson transformation to the error-resilient protocol to get an error-resilient formula (see Section 7).

The above approach shows that an error resilient formula E exists. However, it turns out that showing that the above transformation can be done in polynomial time is more challenging, and requires overcoming some technical hurdles. In Section 6 we show how to make our transformation efficient.

1.3 Related work

We divide the discussion into two parts. First we shall discuss the more classical related work on fault tolerant circuits, which require that the output is *correct* even in the presence of errors. Second, we discuss the "cryptographic" line of work, which only requires *secrecy* (as opposed to correctness); i.e., they require that an adversary that tampers with the circuit does not learn more than he could have learned with only black-box access to the circuit.

Error Resilient Circuits that Preserve Correctness The problem of constructing error resilient circuits dates back to the work of Von Neumann from 1956 [vN56]. Von Neumann studied a model of *random* errors, where each gate (independently) has an (arbitrary) error with small fixed probability. There have been numerous followup papers to this seminal work, including [DO77, Pip85, Pip88, Fed89, ES99, HW91, GG94, ES], who considered the same noise model. For example, these results have shown ([vN56, DO77, Pip85, GG94]) that any circuit of size s can be encoded into a circuit of size $O(s \log s)$ that tolerates a fixed constant noise rate, and that any such encoding must have size $\Omega(s \log s)$.

For formulas, Evans and Schulman [ES] showed that formulas with gates of odd fan-in k, have a tight threshold β_k such that if the error probability of each gate is $p < \beta_k$ then one can construct an error-resilient formula for any boolean function, and if $p > \beta_k$ then such a task is impossible.[1] For fan-in $k = 2$, Evans and Pippenger [EP98] and Unger [Ung08], gave a threshold of $\beta = 3 - \sqrt{7}/4 \approx 8.856\%$ for formulas. Such a threshold for fan-in 2 circuits (as opposed to formulas), or for even fan-in greater than 2 remains open.

Since it is widely believed that a major obstacle to building a quantum computer is dealing with the noise, the issue of building error-resilient quantum circuits has also received a lot of attention. See [KRUdW10] and the references therein. Yet another rich line of work was initiated by Yao and Yao [YY85] (additional works include [AU90, FPRU90, LMP97] and many others), studying how sorting networks can be made resilient to random errors. All of the results mentioned above are for models where the errors are distributed randomly.

Gál and Szegedy [GS95] consider an adversarial error model where the circuit is broken up into layers according to the depth of the gates, and a small constant fraction of gates at each layer may experience errors. They also relax the requirement of correctness to allow their circuit to be incorrect on certain classes of inputs. They show that every symmetric function has a small circuit that is resilient to this kind of error, and also show an interesting connection between their model and probabilistically checkable proofs.

[1] The specific threshold is $\beta_k = \frac{1}{2} - \frac{2^{k-2}}{k\binom{k-1}{k/2-1/2}}$.

Error Resilient Circuits that Preserve Secrecy Recently, the issue of building circuits resilient to errors has resurfaced in the arena of cryptography [GLM+04, IPSW06, KKS11] . All these results consider worst-case adversarial behavior, but do not guarantee correctness, and instead only guarantee *secrecy*. All these results start with a circuit that has a secret key associated with it. The security guarantee is that an adversary who tampers with the circuit does not gain any (useful) information about the secret key, beyond what he can learn via black-box access to the circuit.

Gennaro *et. al.* [GLM+04] construct a general compiler that converts any algorithm into an error-resilient one. Their error-resilient algorithm uses a small (public) trusted hardware. They allow the adversary to tamper *arbitrarily* with the secret key, but do not allow the adversary to tamper with the computation itself (or the hardware). Loosely speaking, the secure hardware detects errors when they occur, and in such cases it erases the secret key.

The result of [KKS11] also consider tampering only with the secret key, and they construct specific cryptographic schemes (such as encryption schemes and signature schemes) that remain secure, even if the secret keys are tampered with arbitrarily. They do not rely on any secure hardware. We note that both [GLM+04, KKS11] allow *arbitrary* tampering with the secret key; i.e., the tampering function can be any poly-time computable function of the secret key.

Ishai *et. al.* [IPSW06] construct a general compiler that converts any circuit into a tamper resilient one. They allow adversarial tampering with any of the wires during the computation. They consider only specific tampering, including setting a wire to 0 or to 1 and flipping the value of the wire. Similarly to [GLM+04, KKS11], they do not offer correctness, but rather only offer security. In particular, they guarantee that whatever an attacker can learn via a tampering attack can be learned via black-box oracle to the circuit.

We note that our error model is incomparable to the error model of [IPSW06], since different types of errors are allowed and disallowed. Moreover, we allow the adversary to see the entire circuit (including secrets that may be stored) before deciding which gates to corrupt, whereas in the [IPSW06] error model the adversary is "blinded", and needs to decide which wire to tamper with, without knowing the secret keys associated with the circuit. Thus, we provide new results even for the case where correctness is not required, but rather only security is required.

2 Preliminaries

Formulas Throughout this work, we focus on boolean formulas computing boolean functions. A formula is a circuit with $\wedge, \vee, \neg$ gates whose underlying graph structure is a tree. Without loss of generality, one may assume that all negations are applied directly to the inputs. The *depth* of the formula is the longest path in it, and the *size* is the number of wires. Unless otherwise specified, we assume that all $\wedge, \vee$ gates have fan-in 2.

We have the following well known lemma:

Lemma 4. *For every formula of size s, there is another formula computing the same function of depth $O(\log s)$.*

Communication Protocols A 2-party communication protocol P, between two parties Alice and Omar, is defined as a directed acyclic graph (usually a tree). The vertices are partitioned into sets $\mathcal{P}_A$ where Alice speaks and $\mathcal{P}_O$ where Omar speaks. Without loss of generality, we assume that the out-degree of each vertex is either 0 or a constant k.[2]

[2]This corresponds to the alphabet being $[k]$.

Each vertex $p \in \mathcal{P}_A$ (resp. $p \in \mathcal{P}_O$) with out-degree greater than 0 is associated with a function h_p mapping the inputs of Alice (resp. Omar) to an out-edge. Every vertex of out-degree 0 is labeled by an output value for the protocol. For a vertex p and number i, we write p_i to denote the vertex reached by following the i'th out-edge of p.

The protocol has a designated root denoted by p_{root}. If Alice is given the input x and Omar is given y, one can compute the value of each vertex p of the protocol, to obtain $p(x, y)$ as follows. If p has out-degree 0, the value is simply the label of p. Otherwise

$$p(x,y) = \begin{cases} p_{h_p(x)}(x,y) & \text{if } p \in \mathcal{P}_A \\ p_{h_p(y)}(x,y) & \text{if } p \in \mathcal{P}_O \end{cases}$$

The outcome of the protocol is $p_{\mathsf{root}}(x, y)$.

Karchmer-Wigderson Games For any boolean function $f : \{0,1\}^n \to \{0,1\}$, the Karchmer-Wigderson game for f is a communication game where Alice gets $x \in f^{-1}(0)$, Omar gets $y \in f^{-1}(1)$, and the goal is for them to use a communication protocol to compute an index $i \in [n]$ such that $x_i \neq y_i$.

Let us restrict our attention to formulas and protocols with fan-in 2. Then Karchmer and Wigderson proved the following theorem.

Theorem 5. *The minimum depth of a formula computing f is equal to the minimum communication of protocols that solve the Karchmer-Wigderson game for f.*

Their proof provides a transformation from a formula to a communication protocol for the Karchmer-Wigderson game, and vice versa. Below we describe how a protocol is obtained from a formula.

We recall that for any boolean function $f : \{0,1\}^n \to \{0,1\}$, the Karchmer-Wigderson game for f is a communication game where Alice gets $x \in f^{-1}(0)$, Omar gets $y \in f^{-1}(1)$, and the goal is for them to use a communication protocol to compute an index $i \in [n]$ such that $x_i \neq y_i$.

Given a formula F (fan-in 2) computing f, one defines the protocol as follows. Alice and Omar start at the top of the formula. At a $\wedge$ gate, Alice will transmit a 1 or 2 to indicate one of the children of the current gate that evaluates to 0 on her input. If she transmits 1, they move to the left child, and if she transmits 2, they move to the right child. At an $\vee$ gate, Omar transmits 1 or 2 to indicate one of the children that evaluates to 1 on his input. Once Alice and Omar reach a leaf, they have identified a variable that is 0 in Alice's input and 1 in Omar's input, as required. Observe that when this protocol is expressed as a tree, the vertices correspond to the gates and leaves of F.

This simple protocol proves:

Lemma 6. *For every formula of depth d computing a boolean function f, there is a binary protocol P solving the Karchmer-Wigderson game for f with communication d.*

For the proof of correctness for this protocol, and the reverse transformation from protocols to formulas, see [KW88].

3 Short-Circuit Errors

Fix a formula F taking n bits as input. Our goal is to find a formula that computes F even in the presence of errors. Without loss of generality, we assume that every $\wedge$ and $\vee$ gate in the formula

has the same constant fan-in k (if not, one can add dummy inputs to the gate to make this so). For the sake of concreteness, throughout this work, one can think of $k = 2$ or $k = 3$. We consider errors that replace any $\wedge$ or $\vee$ gate with an arbitrary function $g : \{0,1\}^k \to \{0,1\}$, as long as $g(0^k) = 0$ and $g(1^k) = 1$. In particular, for every $x = (x_1, \ldots, x_k)$ there exists $i \in [k]$ such that $g(x) = x_i$. In other words, we consider errors that replace a gate with some dictatorship.

We denote the root gate of the formula by g_{root}. For any non-input gate g and $i \in [k]$, we denote by g_i the gate that provides the i'th input to g. Let $\mathcal{G}_A$ (resp. $\mathcal{G}_O$) denote the set of $\wedge$ (resp. $\vee$) gates in F.

An error pattern is defined to be a pair of strings $q = (a, b)$, where a is the string that encodes the errors of the $\wedge$ gates, and b is the string that encodes the errors of the $\vee$ gates. More specifically, $a \in ([k] \cup \{*\})^{|\mathcal{G}_A|}$ is indexed by the gates of $\mathcal{G}_A$, and $b \in ([k] \cup \{*\})^{|\mathcal{G}_O|}$ is indexed by the gates of $\mathcal{G}_O$. Intuitively, if $q_g = i$ (which means that either $g \in \mathcal{G}_A$ and $a_g = i$, or $g \in \mathcal{G}_O$ and $b_g = i$), then the gate g has been *short-circuited*, and the output of the gate is set to be its i'th input.

Given an input $z \in \{0,1\}^n$, an error pattern $q = (a, b)$, and a gate $g \in \mathcal{G}$, we write $g(z, a, b)$ to denote the value computed at g using z as the input and (a, b) as the error pattern. $g(z, a, b)$ is computed as follows. If g is a leaf, then $g(z, a, b)$ is the value of the literal of g on z. If g is an internal gate, then define

$$g(z,a,b) = \begin{cases} g_i(z,a,b) & \text{if } q_g = i \in [k], \\ g_1(z,a,b) \wedge g_2(z,a,b) \wedge \ldots \wedge g_k(z,a,b) & \text{if } q_g = * \text{ and } g \text{ is an } \wedge \text{ gate} \\ g_1(z,a,b) \vee g_2(z,a,b) \vee \ldots \vee g_k(z,a,b) & \text{if } q_g = * \text{ and } g \text{ is an } \vee \text{ gate} \end{cases}$$

If q_g is not $*$, we say that the gate has been *short-circuited*. We shall reserve a^*, b^* to denote strings that encode no errors.

The following observation is useful:

Claim 7. *Let g be a gate, z be an input, and (a, b) be any error pattern. Then $g(z, a, b^*) = 0 \Rightarrow g(z, a, b) = 0$, and $g(z, a^*, b) = 1 \Rightarrow g(z, a, b) = 1$.*

Proof. Short-circuiting an $\vee$ gate can only change its output value from 1 to 0, which can only change the outcome of the entire formula from 1 to 0. The case of $\wedge$ gates is symmetric. □

3.1 Communication Protocols with Errors

One can define a communication protocol with short-circuit errors similarly to the way we defined a formula with short-circuit errors. More specifically, recall that a protocol P between two parties, Alice and Omar, is defined to be a directed acyclic graph, where the vertices in the graph are partitioned into two sets $\mathcal{P}_A$ and $\mathcal{P}_O$, depending on whether it is Alice's turn to speak or Omar's turn to speak.

Given an input x to Alice, an input y to Omar, and an error pattern $q = (a, b)$, where $a \in ([k] \cup \{*\})^{|\mathcal{P}_A|}$ is indexed by the vertices of $\mathcal{P}_A$, and $b \in ([k] \cup \{*\})^{|\mathcal{P}_O|}$ is indexed by the vertices of $\mathcal{P}_O$, the outcome of the protocol (with errors), denoted by $p_{\mathsf{root}}(x, a, y, b)$, is computed as follows. For any vertex p in the graph, if p is a leaf then the outcome is the label of p. Otherwise,

$$p(x,a,y,b) = \begin{cases} p_i(x,a,y,b) & \text{if } q_g = i \in [k] \\ p_{h_p(x)}(x,a,y,b) & \text{if } q_g = * \text{ and } g \in \mathcal{P}_A \\ p_{h_p(y)}(x,a,y,b) & \text{if } q_g = * \text{ and } g \in \mathcal{P}_O \end{cases}$$

The main difference between our error model, and the standard error model in communication, is that here a party *knows* if a message it sent was "tampered with" by the adversary. We emphasize, however, that like the standard error model, the adversary can replace any message sent by Alice or Omar with an arbitrary symbol in the alphabet, which in our case is $[k]$. The difference between the models allows us, in Section 4, to construct relatively simple communication protocols that are resilient to error, and to avoid the use of complicated objects such as tree codes.

4 Karchmer Wigderson games with Errors

In this section we prove an error-resilient version of the Karchmer-Wigderson result. Namely, we prove that if the KW protocol is resilient to errors (as defined in Section 3.1), then when applying the Karchmer-Wigderson transformation to the protocol, we obtain a formula that is resilient to short-circuit errors.

Lemma 8. *Let f be a boolean function, and let P be a protocol with root p_{root}. Let $T \subset f^{-1}(0) \times ([k] \cup \{*\})^{\mathcal{P}_A}$ and $U \subset f^{-1}(1) \times ([k] \cup \{*\})^{\mathcal{P}_O}$ be two nonempty sets such that the protocol rooted at p_{root} solves the game on every pair of inputs in $T \times U$, and every vertex that is a descendent of p_{root} can be reached using the inputs $T \times U$. Then there is a formula with root g that is obtained by replacing every vertex where Alice speaks with a $\wedge$ gate, every vertex where Omar speaks with an $\vee$ gate and every leaf with a literal, such that for every $(x,a) \in T, (y,b) \in U$, $g(x,a,b^*) = 0$ and $g(y,a^*,b) = 1$.*

Remark 9. *We note that Lemma 8, together with Claim 7, implies that the formula g obtained by Lemma 8 satisfies that for every $(x,a) \in T, (y,b) \in U$, and for any error strings a', b' it holds that $g(x,a,b') = 0$ and $g(y,a',b) = 1$.*

Proof. We prove the lemma by induction on the depth of the communication protocol. If the depth is 0, then there must exist an index i such that for every $(x,a,y,b) \in T \times U$, $x_i \neq y_i$. Since $T \times U$ is a rectangle, this must mean that either $x_i = 0, y_i = 1$ for all such x, y or $x_i = 1, y_i = 0$ for all such x, y. Thus, either the literal z_i or $\overline{z_i}$ satisfies the hypothesis in this case. (The error patterns are irrelevant in this case.)

For the general case, assume without loss of generality that the root of the protocol p is in $\mathcal{P}_A$. Let $T_1, \ldots, T_k$ be the partition of the set T according to the message that Alice sends. Since every vertex is reachable, all of these sets are non-empty. By induction, the corresponding formulas $g_1, \ldots, g_k$ satisfy the hypothesis for the sets $T_1 \times U, \ldots, T_k \times U$ respectively.

Let $(x,a),(y,b)$ be any elements of $T \times U$. Assume without loss of generality that $(x,a) \in T_1$. Then by induction $g_1(x,a,b^*) = 0$. Now a_g must be in $\{1,*\}$ (since $(x,a) \in T_1$). If $a_g = 1$, then $g(x,a,b^*) = g_1(x,a,b^*)$. If $a_g = *$, then $g(x,a,b^*) = g_1(x,a,b^*) \wedge \ldots \wedge g_k(x,a,b^*)$. In either case, $g(x,a,b^*) = 0$. By induction, $g_1(y,a^*,b) = \cdots = g_k(y,a^*,b) = 1$. Thus $g(y,a^*,b) = 1$.

□

5 Protocols Resilient to Short-Circuits

In this section, we show how to transform any communication protocol into a protocol that is resilient to short-circuit errors. Let $\mathcal{P}'$ be a protocol with root p_{root} and depth $2d$ using the alphabet [2]. We assume without loss of generality that Alice and Omar alternate transmissions in $\mathcal{P}'$. This is without loss of generality, since any protocol can be modified so that it alternates transmissions, by increasing the communication complexity by at most a factor of two. We rely on this property of alternating transmissions for the success of the resulting error-resilient protocol.

We define a new protocol $\mathcal{S}$ that yields the same result as $\mathcal{P}'$ even in the presence of errors.[3] We focus on the case where the alphabet of $\mathcal{S}$ is [3]. We will later discuss the case where the alphabet is [2].

Intuition. The error resilient protocol $\mathcal{S}$ will try to run $\mathcal{P}'$, but allows the parties to backtrack when an error is detected. More precisely, the parties will traverse vertices of $\mathcal{P}'$, trying to go along the path they would follow in the error-free case. This is reminiscent of the techniques of Schulman [Sch93], though things are much simpler for us because the parties always agree on the current location of the simulation in $\mathcal{P}'$. When a party detects that an error has occurred, he or she will attempt to return to a previous vertex in $\mathcal{P}'$. It is convenient to use the three symbol alphabet $\{1, 2, R\}$. A transmission of 1 causes the simulation to move to the left child of the current vertex in $\mathcal{P}'$, while a transmission of 2 moves to the right child. A transmission of R moves back from the current node to its grandparent. By only allowing small movements at a time, we bound the effect of an error on the progress of the simulation.

By limiting the total number of errors, we can ensure that after a certain number of steps, the parties have made quantifiable progress towards the goal of simulating $\mathcal{P}'$. To make sure that even the final steps of the simulated path in $\mathcal{P}'$ are correct, we artificially extend paths in $\mathcal{P}'$ to have extra "padding" vertices at the end. After a sufficient number of steps, we can guarantee that the simulated path matches the true path in $\mathcal{P}'$ for the meaningful steps, and only differs beyond in this padding region where the exact path taken is unimportant. We define the action of the protocol on the padding vertices similarly to that on the vertices of $\mathcal{P}'$: if a party detects an error in the current simulated path, he or she will transmit R to move back. Otherwise, a party always transmits 1 at a padding vertex. This simply sets the convention that the "error-free" way to traverse the padding vertices is to always go forward to the left child.

5.1 The protocol

We start by defining a protocol $\mathcal{P}$ simulating $\mathcal{P}'$, that has $6e$ padding steps at the end, and that preserves the alternating transmissions property of $\mathcal{P}'$.

Obtaining Protocol $\mathcal{P}$ from Protocol $\mathcal{P}'$

1. Replace every leaf v of $\mathcal{P}'$ with a path of vertices $v^1, \ldots, v^{6e}$. All edges from v^i point to v^{i+1}, and v^{6e} is a leaf labeled by the same value as v was labeled with, in such a way that these vertices continue to alternate transmissions. For each $1 \leq i < 6e$ and for every possible input (x, y) set $h_{v^i}(x) = 1$ if Alice owns v^i, and set $h_{v^i}(y) = 1$ if Omar owns it. The protocol now has depth $D = 2d + 6e$.

2. Ensure that the graph of the protocol is a tree possibly by duplicating vertices.

Observe that $\mathcal{P}$ simulates $\mathcal{P}'$. Let p_{root} be the root of $\mathcal{P}$.

We obtain our final protocol by simulating $\mathcal{P}$ and adding in the ability to backtrack. Intuitively, in $\mathcal{S}$, the parties simulate the transmissions of $\mathcal{P}$, unless their input is inconsistent with the vertex in $\mathcal{P}$. We say that an input x (resp. y) is consistent with a vertex w in the protocol tree if x (resp. y) is consistent with each step that leads to w from the root.

[3] We mention that the resulting error-resilient protocol $\mathcal{S}$ does not have the alternating transmissions property.

Protocol $\mathcal{S}$

Vertices. For every $t \in \{0, 1, \ldots, D\}$ and every vertex $w \in \mathcal{P}$ at distance at most t from p_{root}, define the vertex $s^{w,t}$ of $\mathcal{S}$. The root of the new protocol is $s^{p_{\mathsf{root}},0}$.

Edges. The leaves of the protocol are all vertices $s^{w,t}$ with $t = D$. Every other vertex $s^{w,t}$ has 3 out-edges, corresponding to the symbols $\{1, 2, R\}$, called $s_1^{w,t}, s_2^{w,t}, s_R^{w,t}$. If w has out-edges to w_1, w_2, the first two children are $s_1^{w,t} = s^{w_1,t+1}$ and $s_2^{w,t} = s^{w_2,t+1}$. The vertex $s_R^{w,t}$ is the one that corresponds to backtracking in the original protocol. Let v be the grandparent of w, or $v = w$ if w has no grandparent. Then set $s_R^{w,t} = s^{v,t+1}$.

Transmission Functions. Next we define the functions associated with each vertex. If w is owned by Alice on input x, the transmission function $h_{s^{w,t}}$ is defined as follows

$$h_{s^{w,t}}(x) = \begin{cases} h_w(x) & \text{if } x \text{ is consistent with } w, \\ R & \text{else.} \end{cases}$$

We define the functions for vertices owned by Omar in the same way.

Output Values. Given a leaf $s^{w,D}$, if w is a descendent of a leaf in $\mathcal{P}'$, we label $s^{w,D}$ with the same label as w. Otherwise, we label $s^{w,D}$ by an arbitrary value.

5.2 Analysis

We now prove the following theorem:

Theorem 10. *Let $p_{\mathsf{root}}, s_{\mathsf{root}}$ denote the roots of the protocols $\mathcal{P}, \mathcal{S}$ as defined above. Then for every input (x, y) and every error string (a, b) containing at most $2e$ errors per path, $s_{\mathsf{root}}(x, a, y, b) = p_{\mathsf{root}}(x, y)$.*

Proof. We claim that $\mathcal{S}$ simulates $\mathcal{P}'$ even if there are up to $2e$ total short-circuit errors per path in $\mathcal{S}$. Fix any input (x, y) to the protocol. Then (x, y) determines a path from the root of $\mathcal{P}$ to a leaf, that we call the *correct path*. Recall that the graph of $\mathcal{P}$ is a tree. For every vertex w in $\mathcal{P}$, the path from p_{root} to w must follow the correct path until some vertex z, and then (possibly) deviate from it. Let C_w be the depth of z, and E_w be the number of error free transmission steps it will take for the protocol to return to z. We shall consider how the function $C_w - E_w$ evolves as the protocol runs through a path of $\mathcal{S}$. We have the following claims.

Claim 11. *Any short-circuit error can decrease the value of $C_w - E_w$ by at most 2.*

Observe that if v is a grandparent of w, then $C_v \geq C_w - 2$ and $E_v \leq E_w$. On the other hand, if v is a child of w, $C_v \geq C_w$, and $E_v \leq E_w + 2$.

Claim 12. *Any error free step must increase the value of $C_w - E_w$ by at least 1.*

If $z = w$, an error free transmission will increase C_w by 1 and keep $E_w = 0$. If not, then an error free step will decrease E_w by 1 and leave C_w unchanged.

Now suppose the number of errors is at most $2e$. Then, after $D = r + 6e$ steps, we must have that the protocol ends at a vertex $s^{w,D}$ such that $C_w - E_w \geq (r + 4e) - 2(2e) = r \Rightarrow C_w \geq r$. Indeed, this implies that the vertex z that corresponds to w on the correct path is at depth at least r. Thus, the protocol must output the same value as the original protocol. □

6 Producing the Error Resilient Formula Efficiently

Recall that our transformation starts by balancing the input formula, then then converting it into a communication protocol, making the protocol error resilient and finally converting the protocol back into a formula. Each of the steps before the final step is easily seen to be polynomial time computable. The functions labeling the nodes of the initial communication protocol are computable by polynomial sized circuits that can be computed from the input formula in polynomial time. Next, the protocol is converted into an error resilient version, and this transformation is also easily seen to be polynomial time computable. The only step that is hard to make efficiently computable is the final one, when the error resilient protocol is converted into a formula.

The reason the last step is not trivially in polynomial time is that the Karchmer-Wigderson transformation (Lemma 8) only works when every node of the protocol tree is reachable. Thus, we need to show how to remove the unreachable nodes from the protocol tree in polynomial time. This is the problem that we focus our attention on in this section.

For convenience, we do this for our non-binary protocol with fan-in 3, though the same argument works for our binary protocol as well. We assume for the sake of simplicity that we start with a formula F of depth d represented as a full binary tree with alternating levels of $\vee$ nodes and $\wedge$ nodes. We also assume that all leaves of F are labeled with independent variables.

We define the protocols $\mathcal{P}$, $\mathcal{P}'$ and $\mathcal{S}$ as in Section 5.1. We let $V(\mathcal{S})$ denote the set of vertices of $\mathcal{S}$.

We let e denote the maximal number of errors per path that will be tolerated for each of Alice and Omar. We let $\mathcal{E}_A$ denote the subset of strings in $([k] \cup \{*\})^{\mathcal{P}_A}$ with at most e non-$*$ symbols per path, and $\mathcal{E}_O$ denote the analogous strings in $([k] \cup \{*\})^{\mathcal{P}_O}$.

Let $\mathit{Recognize} : V(\mathcal{S}) \to \{0,1\}$ be the function that takes as input a vertex γ and outputs 1 if and only if there exist inputs in $F^{-1}(0) \times \mathcal{E}_A$ and $F^{-1}(1) \times \mathcal{E}_O$ which cause Alice and Omar to reach the vertex γ in $\mathcal{S}$.

Lemma 13. *The function Recognize is computable in time poly*(2^d).

Before we prove Lemma 13, we fix some notation. For every pair of inputs $(x, y) \in F^{-1}(0) \times F^{-1}(1)$, we let $H_{x,y} : V(\mathcal{S}) \to V(\mathcal{S})$ denote the function that takes each vertex to the next (in an "error-free" manner) assuming the inputs are x and y. In other words, for each $v \in V(\mathcal{S})$, the vertex $H_{x,y}(v)$ is the child of v corresponding to $h_v(x)$ if v is owned by Alice and to $h_v(y)$ is v is owned by Omar.

Intuition of the proof. Fix any vertex $\gamma \in V(\mathcal{S})$. The straightforward way of checking whether $\mathit{Recognize}(\gamma) = 1$ is to iterate over all possible inputs $(x, y) \in F^{-1}(0) \times F^{-1}(1)$, and check how many errors are required to reach the vertex γ with inputs x and y. Unfortunately this takes time poly(2^n), and thus is far from being efficient.

The main observation in the proof is that we do not need to iterate over all possible inputs. Instead, for each γ, we consider the sequence of $\leq D + 1$ vertices $(\gamma_0, \gamma_1, \ldots, \gamma_J = \gamma)$ on the path from the root γ_0 of $\mathcal{S}$ to γ. As before, the straightforward way of testing whether $\mathit{Recognize}(\gamma) = 1$ is to iterate over all functions $H_{x,y}$, and compute the number of i's for which $\gamma_i \neq H_{x,y}(\gamma_{i-1})$. Each such inequality corresponds to an error in the communication protocol.

However, the key observation is that we do not actually have to iterate over *all* functions $H_{x,y}$, but rather it suffices to consider only the functions $H_{x,y}$ restricted to $(\gamma_0, \gamma_1, \ldots, \gamma_{J-1})$.

Proof. Fix any $\gamma \in V(\mathcal{S})$. Compute $\mathit{Recognize}(\gamma)$ as follows.

1. Obtain the sequence of $\leq D+1$ vertices $(\gamma_0, \gamma_1, \ldots, \gamma_J = \gamma)$, as above.

2. Consider the set of all functions $H : \{\gamma_0, \gamma_1, \ldots, \gamma_{J-1}\} \to V(\mathcal{S})$,[4] such that for every $i \in \{0, \ldots, J-1\}$, the vertex $H(\gamma_i)$ is a child of γ_i. (Note that the number of such functions is at most $k^D = \text{poly}(2^d)$.)

   Before we proceed, let us add a definition: For any pair of inputs $(x, y) \in F^{-1}(0) \times F^{-1}(1)$, we say that H and $H_{x,y}$ are consistent on $\{\gamma_0, \gamma_1, \ldots, \gamma_{J-1}\}$ if $H(\gamma_i) = H_{x,y}(\gamma_i)$ for every $i \in \{0, 1, \ldots, J-1\}$.

   For each such function H, do the following.

   (a) Check if there exists a pair of inputs $(x, y) \in F^{-1}(0) \times F^{-1}(1)$ such that H and $H_{x,y}$ are consistent on $\{\gamma_0, \gamma_1, \ldots, \gamma_{J-1}\}$. The fact that this can be done in polynomial time follows from the following claim.

   Claim 14. *For every function $H : \{\gamma_0, \gamma_1, \ldots, \gamma_{J-1}\} \to V(\mathcal{S})$ there is a polynomial time algorithm that tests whether there exists an input pair $(x, y) \in F^{-1}(0) \times F^{-1}(1)$ such that H is consistent with $H_{x,y}$ on $\{\gamma_0, \gamma_1, \ldots, \gamma_{J-1}\}$.*

   (b) If H is consistent with some $H_{x,y}$ on $\{\gamma_0, \gamma_1, \ldots, \gamma_{J-1}\}$, then compute the number of errors in γ with respect to H. Namely, compute the set

   $$E = \{i \in [J] : H(\gamma_{i-1}) \neq \gamma_i\}.$$

   Let $E_A = \{i \in E : \text{Alice owns } \gamma_{i-1}\}$ and let $E_O = \{i \in E : \text{Omar owns } \gamma_{i-1}\}$. If both $|E_A| \leq e$ and $|E_O| \leq e$, then output $\mathit{Recognize}(\gamma) = 1$.

3. Otherwise, output $\mathit{Recognize}(\gamma) = 0$.

It is rather straightforward to see that $\mathit{Recognize}(\gamma) = 1$ if and only if γ is reachable.
Thus it remains to prove Claim 14.
We recall that each vertex $\gamma \in V(\mathcal{S})$ has an associated vertex $\in \mathcal{P}$, which we will denote by w_γ.

Proof of Claim 14. Fix any function $H : \{\gamma_0, \gamma_1, \ldots, \gamma_{J-1}\} \to V(\mathcal{S})$ that maps each γ_i to one of its children. In what follows we present a polynomial time algorithm for deciding whether there exists an input pair $(x, y) \in F^{-1}(0) \times F^{-1}(1)$ such that H is consistent with $H_{x,y}$ on the set $\{\gamma_0, \gamma_1, \ldots, \gamma_{J-1}\}$.

1. Let $H' : \{\gamma_0, \ldots, \gamma_{J-1}\} \to [3]$ be the function such that $H'(\gamma_i) = b$ where $H(\gamma_i)$ is the child of γ_i indicated by b. We identify 3 with R.

2. If there exists γ_i for which the associated vertex $w_{\gamma_i} \in \mathcal{P}$ is of depth $\geq d$ (i.e. at or below the location of the original leaves of $\mathcal{P}'$) and $H'(\gamma_i) = 2$, then output 0 (meaning that H is not consistent with any pair of inputs (x, y)).

   The reason such H cannot be consistent with any $H_{x,y}$ is that according to the protocol, when the protocol reaches vertices in $\mathcal{P}$ below the original leaves of $\mathcal{P}'$, then the parties either try to transmit 1 (if the current location is consistent with their input), or try to go back (i.e. transmit R).

3. Check that $H'(\gamma_i)$ only depends on w_{γ_i}: i.e. if some $w_{\gamma_i} = w_{\gamma_j}$ and $H'(\gamma_i) \neq H'(\gamma_j)$, then output 0.

[4] Recall that γ_0 always denotes the root.

4. Output 1 if the following condition holds for every $i \in \{0, 1, \ldots, J-1\}$: $H'(\gamma_i) = R$ iff $\exists \gamma_j \in \{\gamma_1, \ldots, \gamma_J\}$ s.t. w_{γ_j} is an ancestor of w_{γ_i} with the same owner but $H(\gamma_j)$ does not correspond to an ancestor of w_{γ_i}.

   We refer to this condition as the consistency condition.

5. Otherwise output 0.

It remains to prove that this algorithm is correct. It is easy to see that if the above algorithm outputs 0 then it must be the case that H is not consistent with any $H_{x,y}$ on the set $\{\gamma_0, \ldots, \gamma_{J-1}\}$. We thus focus on the case that the algorithm outputs 1, and we prove that in this case there exists $(x, y) \in F^{-1}(0) \times F^{-1}(1)$ such that H is consistent with $H_{x,y}$ on the set $\{\gamma_0, \ldots, \gamma_{J-1}\}$.

We prove this by induction on the depth d of the formula F. Note that the consistency condition above can be thought of as two separate conditions: one for vertices owned by Alice and one for vertices owned by Omar. We will refer to these separated conditions as "Alice consistency" and "Omar consistency." We note that $H_{x,y}$ only depends on x for vertices owned by Alice and only depends on y for vertices owned by Omar. We formulate our inductive hypothesis as:

Inductive hypothesis for depth $d-1$. For H satisfying conditions 2 and 3 above, and satisfying Alice consistency, there exists an input x such that for every y, $H_{x,y}$ is consistent with H on the vertices in the $\{\gamma_0, \ldots, \gamma_{J-1}\}$ owned by Alice. Similarly, for H satisfying conditions 2 and 3 and Omar consistency, there exist an input y such that for every x, $H_{x,y}$ is consistent with H on the vertices in $\{\gamma_0, \ldots, \gamma_{J-1}\}$ owned by Omar.

Base case. When $d = 0$, the formula is simply a leaf, labeled by some variable. In this case, we choose arbitrary x and y such that $F(x) = 0$ and $F(y) = 1$, and prove that H is consistent with $H_{x,y}$. Note that $\mathcal{P}'$ is a single vertex, and $\mathcal{P}$ is a tree of depth $6e$, with all leaves labeled by the same value. We consider the behavior of the protocol $\mathcal{S}$ with inputs x, y: for each non-leaf vertex v in S corresponding to a vertex $w \in \mathcal{P}$ owned by Alice, $H_{x,y}(v)$ will be the first child of v if and only if the path from the root to w in $\mathcal{P}$ includes only the first children of all vertices owned by Alice. Otherwise, $H_{x,y}(v)$ will be the third child of v.

We observe that this agrees with any H satisfying Alice consistency on the vertices among $\gamma_0, \ldots, \gamma_{J-1}$ owned by Alice. This follows from the consistency condition on these vertices and the additional check that H can never specify the second child for the nodes in $\mathcal{P}$ replacing the leafs of $\mathcal{P}'$. We rely here on the fact that if H is defined on a vertex γ_i, then it is also defined on all of its ancestors. This follows from the fact that $\gamma_0, \ldots, \gamma_{J-1}$ corresponds to a path in $\mathcal{S}$, and thus cannot visit a vertex of $\mathcal{P}$ without traversing a path in $\mathcal{P}$ from the root to this vertex. An analogous argument applies to H satisfying Omar consistency. Thus our base case holds.

Induction step. Now, suppose $d \geq 1$ and the top gate of F is $\wedge$ (the case of $\vee$ is analogous). The first and second children of this $\wedge$ gate correspond to formulas F_1 and F_2 respectively of depth $d-1$, with disjoint variables at their leaves. We will think of an input x for Alice as being composed of an input x_1 for F_1 and an input x_2 for F_2 (note that these are independent). We similarly consider an input y for Omar as y_1, y_2. For an H satisfying Alice consistency, we seek to define x_1, x_2 appropriately. We consider $H'(\gamma_0)$ (recall γ_0 is the root), which must be either 1 or 2. Without loss of generality, we suppose that $H'(\gamma_0) = 1$, and we set the input variables x_2 for Alice all to be 1. This ensures that $F_2(x_2) = 1$, so in the error-free Karchmer-Wigderson protocol, Alice will go from the root of F to F_1. We set the input variables x_1 for Alice using our inductive hypothesis as follows (we will ensure that $F_1(x_1) = 0$).

We let $\mathcal{P}_1, \mathcal{S}_1$ and $\mathcal{P}_2, \mathcal{S}_2$ denote the communication protocol trees associated with F_1 and F_2 respectively. We note that the vertices of $\mathcal{P}_1, \mathcal{P}_2$ can also be viewed as vertices in $\mathcal{P}$. (Recall that vertices in the error-free Karchmer-Wigderson protocol correspond to gates and leaves in F, so the vertices of $\mathcal{P}_1$ correspond to F_1 plus padding vertices.) The path $\gamma_0, \ldots, \gamma_{J-1}$ in $\mathcal{S}$ corresponds to a bi-directional walk in the tree $\mathcal{P}$, sometimes traversing vertices in $\mathcal{P}_1$ and sometimes traversing in $\mathcal{P}_2$. The time spent traversing in $\mathcal{P}_1$ corresponds to a path in $\mathcal{S}_1$. We restrict H to this path and let H_1 denote the resulting function (recall here that $H'(\gamma_i)$ only depends on w_{γ_i}, so there is no ambiguity introduced). It is easy to see that H_1 satisfies the necessary conditions to apply our inductive hypothesis (in particular, Alice consistency is maintained). This gives us an input x_1 for Alice on F_1, such that if we take $x = (x_1, x_2)$ we get that $H_{x,y}$ agrees with H_1 on its domain for vertices owned by Alice, for any setting of y. This concludes our setting of inputs x for Alice, and we note that $F(x) = 0$ is ensured.

We observe that (for any y) $H_{x,y}(\gamma_i) = H(\gamma_i) = R$ for any γ_i owned by Alice whose associated w_{γ_i} is in $\mathcal{P}_2$. This holds by the Alice consistency condition for H since $H'(\gamma_0) = 1$ and by the fact that F_2 evaluates to 1 for Alice's input. Finally, $H_{x,y}(\gamma_0) = H(\gamma_0) = 1$ (for all y), so H and $H_{x,y}$ agree on the vertices owned by Alice in the domain $\gamma_0, \ldots, \gamma_{J-1}$.

For an H satisfying Omar consistency, we consider H_1 and H_2, the restrictions of H to $\mathcal{S}_1$ and $\mathcal{S}_2$ as above. These restrictions inherit all of the required features to apply our inductive hypothesis, including Omar consistency. Therefore, there exist y_1 and y_2 such that for all x_1 and all x_2, $H_{x_1,y_1} = H$ on all vertices γ_i owned by Omar in $\mathcal{S}_1$, and $H_{x_2,y_2} = H$ on all vertices γ_i owned by Omar in $\mathcal{S}_2$. We note that $F_1(y_1) = F_2(y_2) = 1$. Combining these, we obtain $y = (y_1, y_2)$ such that $F(y) = 1$ and for all x, $H_{x,y} = H$ for all vertices owned by Omar in $\gamma_0, \ldots, \gamma_{J-1}$.

□

7 Proofs of Theorems 2 and 3

We now combine the results of the previous sections to prove our main theorems from Section 1.1.

Proof of Theorem 3 We start with a balanced formula F of depth d (fan-in 2). We let F' denote a formula of depth $\leq 2d$ which is also balanced and alternates $\wedge$ and $\vee$ gates (this can be obtained easily from F by inserting extra gates as necessary). We relabel the leaves of F' with independent variables, recording the mapping between these new variables and the original ones. We then let $\mathcal{P}'$ denote the corresponding communication protocol for the (error-free) Karchmer-Wigderson game for the boolean function computed by F', promised by Lemma 6. We note that this protocol alternates between vertices owned by Alice and Omar.

Applying Theorem 10, we can efficiently obtain the error-resilient protocol $\mathcal{S}$ of depth $\leq 2d+6e$ using alphabet [3] that simulates $\mathcal{P}'$ correctly when there are at most $2e$ total errors on each path. In particular, $\mathcal{S}$ is correct whenever each path contains at most e errors for Alice and at most e errors for Omar. Using Lemma 13, we can efficiently compute which vertices of $\mathcal{S}$ are reachable from a pair of inputs (x, a) and (y, b), such that $F'(x) = 0$, $F'(y) = 1$, and a and b are error strings containing at most e short-circuit errors per path each. We remove the unreachable vertices from $\mathcal{S}$ and call the resulting protocol $\mathcal{S}'$. Applying Lemma 8, we obtain an error-resilient formula E' of depth $\leq 2d + 6e$ and fan-in 3. Here, we have set $T = (F')^{-1}(0) \times \mathcal{E}_A$ and $U = (F')^{-1}(1) \times \mathcal{E}_0$ (recall $\mathcal{E}_A$ denotes the set of error strings a encoding at most e errors per path and $\mathcal{E}_O$ is defined analogously).

We note that the variables labeling the leaves of E' can be computed efficiently. To see this, recall that the vertices of our $\mathcal{S}'$ correspond to vertices in $\mathcal{P}$, and the leaves of $\mathcal{S}'$ will correspond

to vertices in $\mathcal{P}$ that have replaced leaves in $\mathcal{P}'$. The leaves of the error-free Karchmer-Wigderson protocol $\mathcal{P}'$ correspond to the leaves of the formula F', and hence we can use the labels of the leaves in F' to efficiently label the leaves of E'. Finally, we replace the variables labeling the leaves of E' with the corresponding original variables for F. This gives us our final error-resilient formula E.

Proof of Theorem 2 This is exactly like the proof of Theorem 3, replacing the alphabet [3] protocol $\mathcal{S}$ with the binary protocol $\mathcal{S}_{[2]}$ defined below (making the appropriate minor adjustments to the proof of Lemma 13).

The idea is to replace each vertex s with children s_1, s_2, s_3 in $\mathcal{S}$ by a binary tree in $\mathcal{S}_{[2]}$ consisting of a root with two children and two grandchildren descending from the right child. We will call these vertices $w, w_1, v = w_2, v_1, v_2$ (w denotes the root, with children w_1 and $w_2 = v$, and v_1, v_2 denote the children of v). The root vertex w corresponds to s, and the vertex w_1 corresponds to s_R. The vertices v_1, v_2 correspond to the children s_1, s_2. If s is owned by Alice, then both w, v are owned by Alice (and vice versa). The idea is that the transmission of R is encoded by the string 1, and the transmission of $1, 2$ are encoded by $21, 22$. We truncate the resulting protocol at depth $2r + 10e$.

Theorem 15. *Let $s_{\mathsf{root}}, p_{\mathsf{root}}$ denote the roots of the protocols $\mathcal{P}, \mathcal{S}_{[2]}$ defined above. Then for every input (x, y), and every error pattern (a, b) with at most $2e$ errors per path, $s_{\mathsf{root}}(x, a, y, b) = p_{\mathsf{root}}(x, y)$.*

Proof. For the analysis, note that every vertex q in the binary protocol corresponds to the tree of some $s^{w,t}$ in the non-binary protocol. As before, we let z denote vertex on the correct path that is closest to w. If $w = z$ and q is an internal vertex, i.e. q is neither the root or a leaf of the tree corresponding to $s^{w,t}$, we set C_q to be $2C_w + 1$ (i.e. twice the depth of z plus one). Otherwise, we set $C_q = 2C_w$. As before, we set E_q to be the number of error-free steps required to get the vertex back to a point where $w = z$. Now we consider the quantity $C_q - E_q$ along a path in $\mathcal{S}_{[2]}$.

Claim 16. *A short-circuit step can decrease $C_q - E_q$ by at most 4.*

If the short circuit is to take a backtracking step, then E_q can only decrease, and C_q can decrease by at most 4. If the short-circuit is to take a non-backtracking step, then the worst case is when q is the root of the tree corresponding to $s^{w,t}$, in which case E_w can increase by at most 4, and C_w cannot decrease. It is easy to check the following claim:

Claim 17. *An error free step must increase $C_q - E_q$ by at least 1.*

If $E_q = 0$, C_q must increase by 1 and E_q will remain 0. Otherwise, C_q will remain at least as large, and E_q will decrease by 1.

Thus, if the number of errors is $2e$, we have that after $2r + 10e$ steps, the protocol reaches a vertex q for which $C_q - E_q \geq 2r + 8e - 8e = 2r$. This implies that $C_w \geq r$, and the protocol must be correct. □

8 Open Problems

The fraction of errors tolerated per path by our construction is a constant, bounded by $\frac{1}{10}$ for fan-in 2, and by $\frac{1}{6}$ for fan-in 3. We leave it as an open problem to determine the *optimal* constant fraction of errors that can be tolerated per path. Extending our result to circuits is a very interesting open problem.

References

[AU90] Shay Assaf and Eli Upfal. Fault tolerant sorting network. In *FOCS*, pages 275–284, 1990.

[BR11] Mark Braverman and Anup Rao. Towards coding for maximum errors in interactive communication. In *STOC*, pages 159–166, 2011.

[Bra12] Mark Braverman. Towards deterministic tree code constructions. In *ITCS*, pages 161–167, 2012.

[DO77] R. L. Dobrushin and S. I. Ortyukov. Upper bound for the redundancy of self- correcting arrangements of unreliable functional elements. 13:203–218, 1977.

[EP98] William S. Evans and Nicholas Pippenger. On the maximum tolerable noise for reliable computation by formulas. *IEEE Transactions on Information Theory*, 44(3):1299–1305, 1998.

[ES] W. Evans and L. Schulman. On the maximum tolerable noise of k-input gates for reliable computation by formulas.

[ES99] W. Evans and L. Schulman. Signal propagation and noisy circuits. In *IEEE Trans. Inform. Theory, 45(7)*, page 23672373, 1999.

[Fed89] T. Feder. Reliable computation by networks in the presence of noise. In *IEEE Trans. Inform. Theory, 35(3)*, page 569571, 1989.

[FPRU90] Uriel Feige, David Peleg, Prabhakar Raghavan, and Eli Upfal. Computing with unreliable information (preliminary version). In *STOC*, pages 128–137, 1990.

[GG94] Péter Gács and Anna Gál. Lower bounds for the complexity of reliable boolean circuits with noisy gates. *IEEE Transactions on Information Theory*, 40(2):579–583, 1994.

[GLM+04] Rosario Gennaro, Anna Lysyanskaya, Tal Malkin, Silvio Micali, and Tal Rabin. Algorithmic tamper-proof (atp) security: Theoretical foundations for security against hardware tampering. In *TCC*, pages 258–277, 2004.

[GMS11] R. Gelles, Ankur Moitra, and Amit Sahai. Efficient and explicit coding for interactive communication. In *FOCS*, pages 159–166, 2011.

[GS95] A. Gal and M. Szegedy. Fault tolerant circuits and probabilistically checkable proofs. In *Proc. of 10th IEEE Structure in Complexity Theory Conference*, pages 65–73, 1995.

[HW91] Bruce E. Hajek and Timothy Weller. On the maximum tolerable noise for reliable computation by formulas. *IEEE Transactions on Information Theory*, 37(2):388–391, 1991.

[IPSW06] Yuval Ishai, Manoj Prabhakaran, Amit Sahai, and David Wagner. Private circuits ii: Keeping secrets in tamperable circuits. In *EUROCRYPT*, pages 308–327, 2006.

[KKS11] Yael Tauman Kalai, Bhavana Kanukurthi, and Amit Sahai. Cryptography with tamperable and leaky memory. In *CRYPTO*, pages 373–390, 2011.

[KLM94] Daniel J. Kleitman, Frank Thomson Leighton, and Yuan Ma. On the design of reliable boolean circuits that contain partially unreliable gates. In *FOCS*, pages 332–346, 1994.

[KRUdW10] Julia Kempe, Oded Regev, Falk Unger, and Ronald de Wolf. Upper bounds on the noise threshold for fault-tolerant quantum computing. *Quantum Information & Computation*, 10(5&6):361–376, 2010.

[KW88] Mauricio Karchmer and Avi Wigderson. Monotone circuits for connectivity require super-logarithmic depth. In *STOC*, pages 539–550, 1988.

[LMP97] Frank Thomson Leighton, Yuan Ma, and C. Greg Plaxton. Breaking the theta (n $\log^2$ n) barrier for sorting with faults. *J. Comput. Syst. Sci.*, 54(2):265–304, 1997.

[Pip85] Nicholas Pippenger. On networks of noisy gates. In *FOCS*, pages 30–38. IEEE, 1985.

[Pip88] N. Pippenger. Reliable computation by formulas in the presence of noise. In *IEEE Trans. Inform. Theory, 34(2)*, page 194197, 1988.

[Sch93] Leonard J. Schulman. Deterministic coding for interactive communication. In *STOC*, pages 747–756, 1993.

[Ung08] Falk Unger. Noise threshold for universality of two-input gates. *IEEE Transactions on Information Theory*, 54(8):3693–3698, 2008.

[vN56] J. von Neumann. *Probabilistic logics and the synthesis of reliable organisms from unreliable components*, volume 3, pages 43–99. Princeton University Press, 1956.

[YY85] Andrew Chi-Chih Yao and F. Frances Yao. On fault-tolerant networks for sorting. *SIAM J. Comput.*, 14(1):120–128, 1985.