EFFICIENT SCALABLE OVERSIGHT VIA REWINDING

Anonymous authors
Paper under double-blind review

ABSTRACT

Scalable oversight asks how a weak verifier can check the outputs of computations performed by more powerful AI systems. One proposed approach uses relativizing interactive proofs, in which a weak verifier checks the output of a complex oracle-aided computation by interacting with one or more powerful provers. A prominent example is debate, which is highly efficient but assumes that one of two powerful AI models, or provers, is truthful. For certain classes of oracle-aided computations, single-prover relativizing proofs remove this assumption, but at substantially greater computational cost.

In this work, we present a new approach to scalable oversight based on relativizing multi-prover interactive proofs (MIPs). We construct a relativizing MIP for computations whose outputs are robust to changing a small fraction of oracle answers. Our construction improves the efficiency of prior single-prover relativizing proofs without assuming that any prover is truthful. Instead, it uses two provers that may coordinate beforehand but cannot communicate during verification. This separation can be realized by interacting with isolated copies of an AI model, or by effectively "rewinding" a single model, provided its state can be reset so that each interaction has access only to its prescribed history.

In our MIP construction the honest provers run in nearly the time of the original computation, and the verifier (after a one-time preprocessing step) runs in near-linear time in the input length. We implement our MIP and report results on a preliminary MNIST experiment.

1 INTRODUCTION

As AI capabilities grow rapidly, a critical problem is: how can we ensure that AI systems remain safe and behave as intended? One major concern is that, as we are training AI models to perform increasingly complex tasks, we may no longer be able to *efficiently* verify the accuracy of their outputs. As a result, we may not be able to verify that they are behaving in a way that aligns with our intentions. This problem of providing accurate training supervision for increasingly powerful AI models is known as scalable oversight.

For a motivating example, consider the scenario of training an LLM to determine laws a company has violated based on hundreds of pages of records. The LLM outputs one bit for each law (representing yes or no), and now we need to verify whether these bits are correct or not. The straightforward way of doing this would be to have a human expert carefully read through the documents, but this seems unreasonably expensive, especially to produce just one training label. In a setting like this, it is important to have a more efficient training protocol. As another example, we may consider a scenario where we train an LLM on a large public dataset, such as the web; we would like to be able to verify that an output of this LLM is correct efficiently, making few queries to the dataset.

Various approaches to scalable oversight have been proposed in the literature Christiano et al. (2018); Leike et al. (2018); Irving et al. (2018). One approach makes use of interactive proofs, which are a foundational concept in theoretical computer science Goldwasser et al. (2019); Babai (1985). The idea is that a proof does not have to be something that we write down on a piece of paper, but rather can involve interaction; a powerful but untrusted "prover" can convince a weak "verifier" of the output of some hard computation by interacting with the verifier. Since their inception, interactive proofs have been studied extensively, giving rise to various proof models Ben-Or et al. (1988); Feige & Kilian (1997); Arora & Safra (1998); Arora et al. (1998); Goldwasser et al. (2015); Rothblum et al. (2013); Ben-Sasson et al. (2016); Reingold et al. (2016). One extremely efficient model is a

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

multi-prover interactive proof (MIP), where there are two provers and a verifier, and the provers are allowed to talk with each other (and plot against the verifier) only before the interaction with the verifier starts; then they are placed in separate rooms and the verifier asks them questions (like a detective interrogating suspects to make sure that their stories are consistent). While MIPs offer strong efficiency guarantees, the assumption that the provers cannot communicate with each other during the protocol can be hard to enforce in practice Kalai (2018). This has motivated constructions that use cryptographic assumptions to emulate multiple non-communicating provers with a single prover Bitansky & Chiesa (2012).

The starting point of this work is the simple observation that in our scalable oversight setting, this assumption becomes much more natural because the provers are AI models. In our setting the verifier can interact with isolated copies of an AI model, or even with a single model if he can save a snapshot of the model's state before the interaction begins. Then the verifier can run one interaction, "rewind" the model by restoring this snapshot, and run another interaction, so that each interaction has access only to its own history. In this way, rewinding allows a single AI model to play the role of multiple noncommunicating provers. Thus, in this work we focus on the MIP model, which allows us to get a significant improvement in efficiency over previous single-prover protocols.

For scalable oversight, we are particularly interested in proof systems for computations that may have access to an oracle, which are also called relativizing interactive proofs. An oracle is a "black-box" function, meaning that we cannot peer inside to see how it gets from an input to an output. This makes it a natural choice to model sources of information such as human judgment or an external database; in the examples above, we would like the verifier to make only a small number of queries to such an oracle. Relativization is believed to be essential for AI safety purposes: because we do not know what sorts of capabilities AI may have in the future, we would like our results to hold relative to any arbitrary oracle. The UK AISI team even writes that "in order to be relevant to AI safety, an interactive proof result must relativize" Irving & Marshall (2025).

One prominent example of this "relativizing interactive proofs" approach to scalable oversight is debate, which was introduced by Irving, Christiano, and Amodei Irving et al. (2018). In an AI debate, there are two powerful AI models, or provers, who claim different outputs, and the provers debate each other to try to convince a verifier that their output is the correct one. The verifier is extremely efficient, and the authors show that DEBATE, the class of problems which admit a debate protocol, is powerful: in the language of computational complexity theory, DEBATE $=$ PSPACE (where PSPACE is the class of problems solvable in polynomial space). Moreover, this result relativizes.

However, debate makes the strong assumption that *one of the provers is truthful*, i.e., one of the AI models is guaranteed to argue for the correct answer. Intuitively, this means that the verifier has one of the powerful AI models on his side. But scalable oversight is especially important precisely in a setting where we do not have this guarantee, i.e., we do not know whether we can trust any AI model. Recently, for certain classes of oracle computations, Chen, Kalai, and Xi presented single-prover interactive proofs that remove this assumption, but at a substantially greater computational cost Chen et al. (2026). A natural question, then, is whether we can obtain a more efficient relativizing interactive proof without assuming that any prover is truthful.

In this work, we present a new approach to scalable oversight based on relativizing multi-prover interactive proofs (MIPs). The MIP model was first introduced by Ben-Or, Goldwasser, Kilian, and Wigderson Ben-Or et al. (1988) within the field of theoretical cryptography, where it has been studied extensively Bitansky & Chiesa (2012); Blumberg et al. (2014); Kalai et al. (2014); Holmgren & Rothblum (2018); Boneh et al. (2018); Arnon et al. (2025b). These classical results do not relativize, however, so they do not carry over readily to our setting. This brings us to the following question:

Can we construct a multi-prover interactive proof that relativizes?

Unfortunately, MIPs with efficient verification do not relativize for arbitrary oracle computations Chiesa & Liu (2020); Barbara et al. (2025). For example, intuitively, if the computation is "unstable" in the sense that changing an answer to one oracle query may change the output of the entire computation, then the provers can agree to lie about one such answer, and our super-efficient verifier cannot hope to catch them (since he cannot make all of the queries himself). However, the output of such an unstable computation may not be useful, especially in a setting where the oracle already represents something error-prone like human judgment—in such a setting, as a safeguard,

the computation should be *robust* to a few errors. This notion of robust oracle computation was introduced by Chen, Kalai, and Xi Chen et al. (2026) for single-prover interactive proofs, and we adopt their definition.

1.1 Our results

In this work, we present a two-prover doubly-efficient protocol in the MIP model for verifying the output of any robust oracle computation, i.e., a computation where the entire output string does not change if a small fraction of answers to oracle queries are changed.

Theorem 1.1 (Informal; see Theorem 5.1). Let C^O be an ε-robust oracle circuit with n input bits, m output bits, and S gates. After a one-time preprocessing step taking $\widetilde{O}(n+m+S)$ time, there is a two-prover MIP for verifying the output of C^O with the following guarantees:

- Honest provers always convince the verifier of a correct output, while cheating provers (even computationally unbounded ones) can convince the verifier of an incorrect output with probability at most $1/3$.
- The verifier runs in $\widetilde{O}(n+m)$ time and makes $O(1/\varepsilon)$ oracle queries.
- Each honest prover runs in $\widetilde{O}(n+m+S)$ time.
- The number of rounds and the total commuication are both $\mathrm{polylog}(n+m+S)$.

Here $\widetilde{O}(\cdot)$ hides factors polylogarithmic in $n+m+S$.

We note that the preprocessing step is natural when the same circuit is evaluated on many inputs. For a general circuit, the verifier must read the circuit description at least once; our protocol does this during preprocessing, and then this can be reused for every subsequent evaluation. So the preprocessing cost is amortized over evaluations (while each evaluation takes the verifier only $\widetilde{O}(n+m)$ time). Such preprocessing is standard in efficient proof systems Chiesa et al. (2020); Setty (2020); Chen et al. (2023a).

We complement our theoretical result by implementing our protocol and running a proof-of-concept experiment inspired by the MNIST debate experiment of Irving, Christiano, and Amodei Irving et al. (2018). We evaluate our protocol on robust oracle circuits with between 256 and $1{,}048{,}576$ oracle gates. Over this range, the median number of oracle queries made by the verifier remains between 57 and 60; at the largest circuit size, the verifier makes $17{,}476\times$ fewer oracle queries than direct evaluation and its online running time is $0.45\times$ that of direct evaluation.

1.2 Related work

Interactive proofs for scalable oversight. There has been a recent line of work studying interactive proofs as an approach to scalable oversight, starting with a paper by Irving, Christiano, and Amodei on "AI safety via debate" Irving et al. (2018). Their model assumes that the debating provers are computationally unbounded; later, Brown-Cohen, Irving, and Piliouras introduced doubly-efficient debate, where the provers are guaranteed to be efficient Brown-Cohen et al. (2024). More recent works on debate have focused on its limitations, e.g, the "obfuscated arguments problem," where a lying debating prover can make it very hard for the other prover to find the flaw in her argument Barnes (2020); Brown-Cohen et al. (2025); Buhl et al. (2025). Very recently, Chen, Kalai, and Xi initiated the study of relativizing single-prover interactive proofs for scalable oversight, constructing doubly-efficient protocols for certain classes of oracle computations Chen et al. (2026). Our work considers the setting of robust computation that they introduce; we leverage multiple provers to improve the efficiency of their protocols.

Relativizing proof systems. Relativization has been studied extensively as a barrier in computational complexity theory, and notably the classical results on interactive proofs, e.g., the celebrated $\mathsf{IP} = \mathsf{PSPACE}$ and $\mathsf{MIP} = \mathsf{NEXP}$ theorems, do not relativize Chang et al. (1994). On the positive side, prior work has shown that $\mathsf{IP} = \mathsf{PSPACE}$ and $\mathsf{MIP} = \mathsf{NEXP}$ do algebrize, meaning that they relativize if the proof system is also given a low-degree extension of the oracle Aaronson & Wigderson (2009), and there has also been work on constructing relativizing succinct cryptographic proof systems in some structured oracle models Chiesa & Tromer (2010); Chen et al. (2022; 2023b); Chiesa et al.

(2024). On the negative side, recent results show that efficient verification is impossible for arbitrary oracle-aided computations (even when the security guarantee is relaxed to hold only against efficient dishonest provers) Chiesa & Liu (2020); Barbara et al. (2025). Our work circumvents these barriers by restricting to the setting of robust oracle computation.

1.3 Outline of the paper

We begin in Section 2 with an overview of the key ideas behind our results. In Section 3 we construct an encoded-witness IOP, the main technical tool behind our theoretical result. In Section 4, we construct a relativizing IOP, which we use in Section 5 to construct a relativizing MIP. In Section 6 we put everything together to prove our main result. We defer preliminaries, omitted proofs, and our implementation and experiments to the appendices.

2 Technical Overview

Here, let us briefly outline the key ideas behind our construction.

We model the computation that the verifier wishes to verify as an oracle Boolean circuit C^O, i.e., a Boolean circuit that may contain gates that make calls to an arbitrary oracle $O : \{0,1\}^* \to \{0,1\}$. We assume that the oracle circuit is ε-robust, meaning that its output does not change if at most an ε-fraction of answers to its oracle queries are changed (see Definition B.14 for a formal definition). Given an ε-robust oracle circuit C^O, an input x, a claimed output a, and access to O, two provers and the verifier engage in a protocol to prove that $C^O(x) = a$. Recall that the two provers may coordinate before the protocol begins but cannot communicate during the interaction.

Our goal is to design a protocol that is efficient enough to use in practice. In classical cryptography, typically practical proof systems are constructed by first constructing an interactive oracle proof (IOP), which is a single-prover interactive proof where the prover sends long messages but the verifier reads only a few bits of each message, and then compiling the IOP into the desired proof system. We follow a similar blueprint: we construct a relativizing IOP and then show how to turn it into a relativizing MIP.

A relativizing IOP. To start, a natural strategy is to try to turn the oracle circuit C^O into a circuit without oracle gates and then use standard IOPs. To "deoraclize" the circuit, suppose that the prover provides a string w containing a claimed query-answer pair for each oracle gate. We can then define a circuit C_w without oracle gates, which is C^O except with each oracle gate replaced by the corresponding query and answer in w. The basic idea is to have the prover send the string w in the first round of the IOP and then prove that $C_w(x) = a$. Then the verifier can check that the query-answer pairs in w are correct with respect to O.

The verifier cannot check that all the pairs in w are correct, since this would require making all the oracle calls needed in order to run the computation, which he cannot afford. But the verifier does not actually need to do this, since we assume C^O is ε-robust: if $C^O(x) \neq a$, then any w satisfying $C_w(x) = a$ must contain more than an ε-fraction of incorrect oracle answers, so the verifier can catch an incorrect oracle answer with good probability by sampling $O(1/\varepsilon)$ pairs in w.

To obtain stronger efficiency guarantees, instead of w itself, we have the prover send an encoding of w. We design the protocol so that the verifier can later ask the prover for a few bits of w and check that her answers are consistent with this encoding. In particular, the verifier can sample a few oracle gates, ask for the query-answer pair in w for each oracle gate, and check that the pair is correct using O. To achieve this, we construct a variant of an IOP that we call an encoded-witness IOP.

Relativizing IOP $\to$ relativizing two-prover MIP. Now we turn our relativizing IOP into a relativizing two-prover protocol. The idea is similar to the one behind the standard proof for turning a (non-relativizing) IOP into an MIP Ben-Sasson et al. (2016). Recall that in an IOP, the prover sends the verifier long messages, but the verifier reads only a few bits of these. We have the MIP verifier simulate the IOP verifier and one of the two provers simulate the IOP prover, except that the "IOP" prover does not send long messages. Instead, whenever the verifier wants to read a bit in some message, he sends the bit to the "IOP" prover, and the "IOP" prover sends back an answer.

However, now that the "IOP" prover does not fix each message before the verifier queries bits of it, she may lie by answering the verifier's queries adaptively, e.g., the answer that she provides to the verifier's second query to some message depends on the verifier's first query. To deal with this, we also have the verifier ask the second prover a random query and check that the second prover's answer matches the first prover's answer to this query. Because the second prover only gets this one query, in some sense she answers nonadaptively, so we catch the first prover if she changed her answer based on the verifier's other queries. Since the verifier only checks one random query, this gives only a weak security guarantee (e.g., if only a few of the first prover's answers differ from the second prover's answers, we are unlikely to catch such an answer). To improve the security guarantee, we run this protocol several times and accept if only if the verifier accepts in all of the runs.

Naively repeating the protocol would multiply the number of queries that the verifier makes to the oracle O, which we would like to avoid. So instead, we separate the interactive part of the protocol from the oracle checks and have the verifier repeat only the interaction, without making any oracle checks. Then at the end the verifier chooses one repetition at random and performs oracle checks for only that repetition.

Experiments. As a proof of concept, we implement our protocol and evaluate it on a task inspired by the MNIST debate experiment of Irving, Christiano, and Amodei Irving et al. (2018). We use a sparse MNIST classifier, which predicts a digit from six revealed nonzero pixels of an MNIST image, to define our oracle. Given a "sparse view" of an MNIST image and a candidate digit, the oracle outputs whether the classifier predicts that digit. Our circuits query this oracle on many random sparse views of an image and output whether a majority of the views vote for the candidate digit.

We run the protocol on 50 MNIST test images while increasing the number q_C of oracle gates from 256 to $1{,}048{,}576$. The robustness parameter remains nearly constant across these circuit sizes, and the median number of oracle queries made by the verifier stays between 57 and 60. At the largest circuit size, the verifier makes $17{,}476\times$ fewer oracle queries than are needed to evaluate the circuit directly. The verifier's online running time also grows substantially more slowly than direct evaluation, and at $q_C = 1{,}048{,}576$ is $0.45\times$ the time required to evaluate the circuit directly.

3 Encoded-Witness IOPs

In this section we define the notion of encoded-witness IOPs, which serve as a building block in our relativizing IOP in Section 4. An encoded-witness IOP is an IOP in which the prover's first message is an encoding of its witness. Soundness is stated in terms of this first message Ben-Sasson et al. (2016; 2018). In addition, the encoding is required to be interactively locally decodable (Section C).

Definition 3.1 (Encoded-witness IOP). *Let R be a relation whose witnesses are strings $\mathbb{w} \in \{0,1\}^k$, and let $\mathsf{Enc} : \Gamma^k \to \Sigma^\ell$ be an encoding as in Definition C.1. An* **encoded-witness IOP** *for R with encoding Enc is an IOP with preprocessing $(\mathcal{P}, \mathcal{V})$ for R (Definition B.24 and Remark B.28) in which the prover's first message is a proof string $\pi_1 \in \Sigma^\ell$, and the honest prover sends $\pi_1 := \mathsf{Enc}(\mathbb{w})$, where the encoding Enc is interactively locally decodable, with decoder $(\mathcal{P}_{\text{dec}}, \mathbf{D})$ (Definition C.1). We write*

$$(\pi_1, b) \leftarrow \langle \mathcal{P}(\mathbb{w}), \mathcal{V} \rangle(\mathbb{x})$$

for the first message and the decision bit of the verifier. The decoder has the completeness and decoding error of Definitions C.2 and C.3; the IOP has the completeness and soundness properties defined below.

Definition 3.2 (Completeness). *An encoded-witness IOP has* **perfect completeness** *if, for every $(\mathbb{x}, \mathbb{w}) \in R$, the honest prover satisfies*

$$\Pr\left[b = 1 \mid (\pi_1, b) \leftarrow \langle \mathcal{P}(\mathbb{w}), \mathcal{V} \rangle(\mathbb{x})\right] = 1.$$

Definition 3.3 (Soundness). *For an instance $\mathbb{x}$, let $\mathcal{C}_{\mathbb{x}} := \{\mathsf{Enc}(\mathbb{w}) : (\mathbb{x}, \mathbb{w}) \in R\}$ be the set of encodings of valid witnesses. An encoded-witness IOP has* **soundness error** *$\epsilon_{\mathsf{IOP}} : (0,1) \to [0,1]$ if, for every* **proximity parameter** *$\delta \in (0,1)$, every instance $\mathbb{x}$, and every possibly computationally unbounded cheating prover $\widetilde{\mathcal{P}}$,*

$$\Pr\left[b = 1 \ \wedge \ \Delta\big(\pi_1, \mathcal{C}_{\mathbb{x}}\big) > \delta \ \middle| \ (\pi_1, b) \leftarrow \langle \widetilde{\mathcal{P}}, \mathcal{V} \rangle(\mathbb{x})\right] \leq \epsilon_{\mathsf{IOP}}(\delta).$$

Lemma 3.4 (Encoded-witness IOP for circuits, adapted from Chen et al. (2023a); Chiesa et al. (2020); Bünz et al. (2020); Arnon et al. (2025a)). *' Consider instances of R_{CSAT} (Definition B.15) whose circuit has a public input of length n, a private input of length n_w, m outputs, and S gates. Let $N' \coloneqq S + n + n_w + m$ and $K \geq 0$, and let $\mathbb{F}$ and $\mathsf{Enc} : \mathbb{F}^{n_w+S} \to \mathbb{F}^\ell$ be the field and the encoding of Lemma C.4 for $k \coloneqq n_w + S$ and $M \coloneqq c_{\mathbb{F}} \cdot (N' + K)$ for a sufficiently large constant $c_{\mathbb{F}}$. Then R_{CSAT} has an encoded-witness IOP with encoding Enc with the following parameters, where $\widetilde{O}(\cdot)$ hides factors polylogarithmic in N' and $\delta \coloneqq 1/8$.*

- **Perfect completeness**.
- **Soundness error**: $\epsilon_{\mathsf{IOP}}(\delta) \leq 1/12$.
- **Round complexity**: $O(\log N')$.
- **Proof query complexity**: $O(\log N' \cdot \log\log N')$.
- **Proof length**: $O(N')$.
- **Prover time**: $\widetilde{O}(N')$.
- **Verifier preprocessing time**: $\widetilde{O}(N')$.
- **Verifier online time**: $\widetilde{O}(n + m)$.

4 Relativizing IOPs for Robust Oracle Circuits

In this section we construct a relativizing IOP (Definition B.27) for the oracle-circuit-evaluation relation R^O_{CVAL} of Definition B.16, that is, for proving that $C^O(x) = a$.

Theorem 4.1 (relativizing IOP for robust oracle circuits). *Fix an oracle $O : \{0,1\}^* \to \{0,1\}$ and $0 < \varepsilon \leq 1$. Let C^O be an ε-robust oracle circuit with an input of length n, m outputs, and S gates, whose oracle queries have length polylogarithmic in $S+n+m$. Then R^O_{CVAL}, for instances $(\langle C^O\rangle, x, a)$ with this circuit, has a relativizing IOP with preprocessing (Remark B.28) over a prime field of size polynomial in $S + n + m$ with the following properties, where $\widetilde{O}(\cdot)$ hides factors polylogarithmic in $S + n + m$, running times count field operations, and proof lengths count field elements.*

- **Perfect completeness**.
- **Soundness error**: *at most* $1/3$.
- **Round complexity**: $O(\log(S + n + m))$.
- **Proof query complexity**: $O(\log(S + n + m) \cdot \log\log(S + n + m))$.
- **Verifier oracle queries**: $O(1/\varepsilon)$.
- **Proof length**: $\widetilde{O}(S + n + m)$.
- **Prover time**: $\widetilde{O}((S + n + m)/\varepsilon)$.
- **Verifier preprocessing time**: $\widetilde{O}(S + n + m)$.
- **Verifier online time**: $\widetilde{O}(n + m + 1/\varepsilon)$.

The construction combines the encoded-witness IOP for circuits of Lemma 3.4 with random sampling of oracle gates. The prover encodes the evaluation of the de-oraclized circuit of C^O (Definition B.13) on x and the oracle answers recorded in its transcript, and proves that the de-oraclized circuit outputs a. The verifier then samples a few oracle gates, decodes the query and the recorded answer at each of them from the encoded witness, and checks that each answer agrees with the oracle. By robustness (Definition B.14), a wrong output forces more than an ε fraction of the recorded answers to be wrong, so $O(1/\varepsilon)$ samples suffice. Section 4.1 gives the protocol and Section 4.2 proves the theorem.

4.1 Construction

We describe the protocol $(\mathcal{P}, \mathcal{V})$ for R^O_{CVAL}. Fix O, ε, and a field $\mathbb{F}$ as in Theorem 4.1, an instance $\mathbb{x} = (\langle C^O\rangle, x, a)$ with $\mathbb{x}_{\mathrm{pre}} = \langle C^O\rangle$ and $\mathbb{x}_{\mathrm{post}} = (x, a)$, and the witness $z = \mathrm{Tr}_{C^O}(x)$. Let C be the de-oraclized circuit of C^O (Definition B.13), let $w \in \{0,1\}^{q_C}$ be the values of z at the oracle gates, and let

$$u \coloneqq w \parallel \mathrm{Tr}_C(x, w),$$

be the witness of R_{CSAT} (Definition B.15) for the instance $(\langle C\rangle, x, a)$. The answer at oracle gate j is the bit w_j of u, and each bit of its query is a bit of x or of u, at a position determined by $\langle C^O\rangle$. We use the following parameters.

- w_O: one plus the maximum fan-in of an oracle gate.
- $t \coloneqq \min\{\mathsf{q}_C, \lceil(\ln 6)/\varepsilon\rceil\}$: the number of oracle gates the verifier samples.
- $K \coloneqq t \cdot w_O$: the number of positions decoded.
- $\mathbb{F}$, Enc, and $\delta = 1/8$: the field, the encoding, and the proximity parameter of Lemma 3.4 for R_{CSAT} with K positions.
- $(\mathcal{P}_{\mathrm{ew}}, \mathcal{V}_{\mathrm{ew}})$ with $\mathcal{V}_{\mathrm{ew}} = (\mathcal{V}_{\mathrm{ew,pre}}, \mathcal{V}_{\mathrm{ew,post}})$: the encoded-witness IOP of Lemma 3.4.
- $(\mathcal{P}_{\mathrm{dec}}, \mathbf{D})$: the decoder of Enc (Lemma C.4).

1. **Preprocessing**: $\mathcal{V}_{\mathrm{pre}}(\mathbb{x}_{\mathrm{pre}})$ computes the de-oraclized circuit $\langle C\rangle$, runs $\mathcal{V}_{\mathrm{ew,pre}}$ on it, and outputs the resulting state together with, for each oracle gate, the positions in u of its answer and of its query bits.
2. **IOP**: $\mathcal{P}$ and $\mathcal{V}_{\mathrm{post}}$ run $(\mathcal{P}_{\mathrm{ew}}, \mathcal{V}_{\mathrm{ew,post}})$ on $(\langle C\rangle, x, a)$ with witness u, which gives the first message $\pi_1 = \mathsf{Enc}(u)$ and the decision bit b.
3. **Decoding**: $\mathcal{V}_{\mathrm{post}}$ samples a uniformly random set $\{j_1, \ldots, j_t\} \subseteq [\mathsf{q}_C]$ of t oracle gates and reads from its state, for each sampled gate, the position of its answer and the positions of its query bits. This gives a list $\mathcal{I}$ of K positions. Then $\mathcal{P}$ and $\mathcal{V}_{\mathrm{post}}$ run the decoder $(\mathcal{P}_{\mathrm{dec}}, \mathbf{D})$ on the first message π_1 and $\mathcal{I}$, which gives $\zeta \in \{0,1\}^K \cup \{\perp\}$. If $\zeta \neq \perp$, then for each $k \in [t]$ it contains a decoded answer $\widetilde{\mathsf{ans}}_k$ at the j_k-th oracle gate and, together with the bits of x, a decoded query $\widetilde{\mathsf{qry}}_k$.
4. **Decision**: $\mathcal{V}$ accepts if and only if $b = 1$, $\zeta \neq \perp$, and $O(\widetilde{\mathsf{qry}}_k) = \widetilde{\mathsf{ans}}_k$ for every $k \in [t]$.

4.2 Proof of Theorem 4.1

Completeness. Suppose that $C^O(x) = a$. Then w holds the answers of O in the evaluation of C^O on x, so $C(x, w) = a$ and $(w, \mathrm{Tr}_C(x, w))$ is a valid witness. Hence $b = 1$ with probability 1 by the completeness of the encoded-witness IOP, and $\zeta = u|_{\mathcal{I}}$ with probability 1 by the completeness of the decoder (Lemma C.4). The decoded bits are bits of u, so $\widetilde{\mathsf{qry}}_k$ is the query at the j_k-th oracle gate and $\widetilde{\mathsf{ans}}_k = O(\widetilde{\mathsf{qry}}_k)$.

Soundness. Suppose that $C^O(x) \neq a$ and fix a cheating prover. Since the encoding has relative distance at least $1/2$ and $\delta = 1/8$, the first message π_1 is δ-close to the encoding of at most one string. Let (w^*, z^*) be this string if it is a valid witness of R_{CSAT} for the instance $(\langle C\rangle, x, a)$, and $\perp$ otherwise. Note that (w^*, z^*) is determined by π_1.

- If $(w^*, z^*) = \perp$, then $b = 1$ with probability at most $\epsilon_{\mathsf{IOP}}(\delta) \leq 1/12$ by the soundness of the encoded-witness IOP (Definition 3.3 and Lemma 3.4).
- If $\zeta \notin \{(w^*, z^*)|_{\mathcal{I}}, \perp\}$, then, since π_1 is δ-close to $\mathsf{Enc}(w^*, z^*)$, this happens with probability at most $\epsilon_{\mathrm{dec}}(\delta, K) \leq 1/12$ by the decoding error of the decoder (Definition C.3 and Lemma C.4).
- Otherwise, the verifier rejects if $\zeta = \perp$, so suppose $\zeta = (w^*, z^*)|_{\mathcal{I}}$. Then $C(x, w^*) = a \neq C^O(x)$, and the k-th sampled pair is $(\mathsf{qry}_{j_k}(w^*), w^*_{j_k})$, with $\mathsf{qry}_j(\cdot)$ as in Definition B.14. By robustness, $O(\mathsf{qry}_j(w^*)) \neq w^*_j$ for more than $\varepsilon \mathsf{q}_C$ indices j. A uniformly random set of t oracle gates avoids all of them with probability at most $(1-\varepsilon)^t$, which is at most $e^{-\varepsilon t} \leq 1/6$ if $t = \lceil(\ln 6)/\varepsilon\rceil$, and with probability 0 if $t = \mathsf{q}_C$.

If $\mathsf{q}_C = 0$, then $C(x) = C^O(x) \neq a$, so there is no valid witness and only the first case occurs. In total, the verifier accepts with probability at most $1/12 + 1/12 + 1/6 = 1/3$.

Efficiency. The circuit C has $\mathsf{q}_C \leq S$ private input bits and $S - \mathsf{q}_C$ gates, so $N' = S + n + m$ in Lemma 3.4, and $K \leq \lceil(\ln 6)/\varepsilon\rceil \cdot w_O = \widetilde{O}(1/\varepsilon)$, since w_O is polylogarithmic by hypothesis. The round complexity, proof query complexity, proof length, and prover time are the sums of those of the IOP in Lemma 3.4 and of the decoder in Lemma C.4, and the verifier preprocessing time is that of the encoded-witness IOP plus the time $\widetilde{O}(S)$ to compute $\langle C\rangle$ and the positions from $\langle C^O\rangle$. The verifier makes $t = O(1/\varepsilon)$ oracle queries, and its online time is that of the IOP plus that of the decoder plus $\widetilde{O}(K)$ to read the positions and form the queries, for a total of $\widetilde{O}(n + m + 1/\varepsilon)$.

5 Relativizing Two-Prover Interactive Proofs for Robust Oracle Circuits

Here we turn a relativizing IOP into a relativizing two-prover interactive proof.

Theorem 5.1 (relativizing two-prover protocol). *Fix $O : \{0,1\}^* \to \{0,1\}$ and $0 < \varepsilon \leq 1$, and let C^O be an ε-robust oracle circuit with an input of length n, m outputs, and S gates, whose oracle queries have length polylogarithmic in $S + n + m$. For every $\sigma \geq 1$, R^O_{CVAL}, for instances $(\langle C^O \rangle, x, a)$ with this circuit, has a relativizing two-prover interactive proof with preprocessing (Remark B.21) with the following properties, where $\widetilde{O}(\cdot)$ hides factors polylogarithmic in $S + n + m$.*

- **Perfect completeness**.
- **Soundness error**: *at most* $2^{-\sigma}$.
- **Round complexity**: $O(\sigma \log^2(S + n + m) \cdot \log^2 \log(S + n + m))$.
- **Communication complexity**: $\widetilde{O}(\sigma/\varepsilon)$ *field elements.*
- **Verifier oracle queries**: $O(\sigma \min\{\mathsf{q}_C, 1/\varepsilon\})$.
- **Verifier preprocessing time**: $\widetilde{O}(S + n + m)$.
- **Verifier online time**: $\widetilde{O}(\sigma(n + m + 1/\varepsilon))$.
- **Prover time**: $\widetilde{O}(\sigma(S + n + m)/\varepsilon)$.

Running times use the field-operation and access model of Theorem 4.1. Bit complexity also charges arithmetic over $\mathbb{F}$ and the encoding of communicated field elements.

We defer the proof to the appendix.

6 Putting Everything Together

We apply Theorem 5.1 to circuit families to obtain the following doubly-efficient protocol.

Theorem 6.1 (Doubly-efficient two-prover protocol). *Fix an oracle $O : \{0,1\}^* \to \{0,1\}$. Let $\{C_n\}_{n \in \mathbb{N}}$ be a family of oracle circuits, where C_n has n inputs, $m(n)$ outputs, $S(n)$ gates, and $\mathsf{q}_C(n)$ oracle gates, and is $\varepsilon(n)$-robust. Write $H = n + m + S$ for the total size. Assume that, for every n:*

1. **Polynomial size**: $S \leq n^{O(1)}$.
2. **Short oracle queries**: *every oracle query of C_n has length at most* $\log^{O(1)} H$.
3. **Robustness**: $1/\varepsilon \leq \log^{O(1)} H$, *and the number $\varepsilon \mathsf{q}_C$ of tolerated wrong oracle answers grows without bound as q_C does.*

Then for every $\sigma \geq 1$ there is a relativizing two-prover interactive proof with preprocessing for the claim $C^O_n(x) = a$ with the following properties, where running times count bit operations and $\widetilde{O}(\cdot)$ hides factors polynomial in $\log H$*:*

- **Perfect completeness**.
- **Soundness error**: *at most* $2^{-\sigma}$.
- **Round complexity**: $\sigma \cdot \log^{O(1)} H$.
- **Verifier oracle queries**: $O(\sigma \cdot \min\{\mathsf{q}_C, 1/\varepsilon\})$.
- **Verifier preprocessing time**: $\widetilde{O}(H)$, *once per circuit.*
- **Verifier online time**: $\widetilde{O}(\sigma(n + m))$.
- **Prover time**: $\widetilde{O}(\sigma H)$.
- **Communication complexity**: $\sigma \cdot \log^{O(1)} H$ *bits.*

In particular, for every fixed σ the protocol is a quasi-linear doubly-efficient MIP for R^O_{CVAL} (Definition B.22).

Proof of Theorem 6.1. Fix n. We apply Theorem 5.1 to the circuit C_n with soundness parameter σ. Note that we can do this since C_n is ε-robust and its oracle queries have length polylogarithmic in H (hypothesis 2). We use a field of size $H^{O(1)}$ as in Theorem 4.1, so a field operation costs $\log^{O(1)} H$ bit operations.

For the parameters, completeness, soundness, and the number of verifier oracle queries come directly from Theorem 5.1. The round complexity is $O(\sigma \log^2 H \cdot \log^2 \log H) = \sigma \cdot \log^{O(1)} H$. The communication complexity is $\widetilde{O}(\sigma/\varepsilon) = \widetilde{O}(\sigma)$ field elements, which is $\sigma \cdot \log^{O(1)} H$ bits, since $1/\varepsilon \leq \log^{O(1)} H$ by hypothesis 3 and a field element has $O(\log H)$ bits. The running times are counted in field operations, which cost only a polylogarithmic factor to convert to bit operations. The verifier online time and the prover time also carry a factor of $1/\varepsilon$, which is at most $\log^{O(1)} H$ by hypothesis 3. The $\widetilde{O}(\cdot)$ notation absorbs both factors, which gives the stated times.

Finally, fix σ. By hypothesis 1, $\log H = O(\log n)$, so these times are quasi-linear as Definition B.22 requires. The verifier makes Rt oracle queries, where $R = O(\sigma)$ and $t = \min\{\mathsf{q}_C, \lceil (\ln 6)/\varepsilon \rceil\}$. Since $\varepsilon \mathsf{q}_C \to \infty$ (hypothesis 3),

$$\frac{t}{\mathsf{q}_C} \leq \frac{\lceil (\ln 6)/\varepsilon \rceil}{\mathsf{q}_C} \longrightarrow 0,$$

so the number of oracle queries is sublinear in q_C.

□

AI USE STATEMENT

We used generative AI tools to write large portions of the code for our implementation and experiments, draft and edit sections of this paper, proofread the paper to catch potential bugs and suggest writing improvements, do a literature review, and help with numerical calculations. All AI-generated text and code was manually reviewed and verified by the authors. For example, we tested AI-generated code and checked consistency against the theoretical protocol, and we did a manual literature survey. The theoretical model and construction were developed solely by the authors. We take full responsibility for the content of this work.

REPRODUCIBILITY STATEMENT

Proofs are detailed in Sections 3 to 6, all data sources are publicly available, and code for reproducing the experiments is provided in the supplementary material.

REFERENCES

Scott Aaronson and Avi Wigderson. Algebrization: A new barrier in complexity theory. *ACM Transactions on Computation Theory*, 1(1):2:1–2:54, 2009. doi: 10.1145/1490270.1490272. URL https://www.scottaaronson.com/papers/alg.pdf.

Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 2087–2104. Association for Computing Machinery, 2017. doi: 10.1145/3133956.3134104. URL https://doi.org/10.1145/3133956.3134104.

Gal Arnon, Alessandro Chiesa, Giacomo Fenzi, and Eylon Yogev. WHIR: Reed–Solomon proximity testing with super-fast verification. In *Advances in Cryptology – EUROCRYPT 2025*, volume 15604 of *Lecture Notes in Computer Science*, pp. 214–243. Springer, 2025a. doi: 10.1007/978-3-031-91134-7_8. URL https://link.springer.com/chapter/10.1007/978-3-031-91134-7_8.

Gal Arnon, Jesko Dujmovic, and Yuval Ishai. Designated-verifier SNARGs with one group element. In *Advances in Cryptology – CRYPTO 2025*, volume 16006 of *Lecture Notes in Computer Science*, pp. 192–224. Springer, 2025b. doi: 10.1007/978-3-032-01907-3_7. URL https://eprint.iacr.org/2025/517.

Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of np. *Journal of the ACM (JACM)*, 45(1):70–122, 1998.

Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM (JACM)*, 45(3):501–555, 1998.

László Babai. Trading group theory for randomness. In *Proceedings of the seventeenth annual ACM symposium on Theory of computing*, pp. 421–429, 1985.

Annalisa Barbara, Alessandro Chiesa, and Ziyi Guan. Relativized succinct arguments in the ROM do not exist. In *TCC (1)*, volume 16268 of *Lecture Notes in Computer Science*, pp. 400–416. Springer, 2025. doi: 10.1007/978-3-032-12287-2_14. URL https://eprint.iacr.org/2024/728.

Beth Barnes. Debate update: Obfuscated arguments problem, Dec 2020. URL https://www.alignmentforum.org/posts/PJLABqQ962hZEqhdB/debate-update-obfuscated-arguments-problem. AI Alignment Forum post.

Michael Ben-Or, Shafi Goldwasser, Joe Kilian, and Avi Wigderson. Multi-prover interactive proofs: how to remove intractability assumptions. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pp. 113–131. Association for Computing Machinery, 1988.

Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil Vadhan. Robust pcps of proximity, shorter pcps and applications to coding. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pp. 1–10, 2004.

Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In *Theory of Cryptography Conference*, pp. 31–60. Springer, 2016. Full version: Cryptology ePrint Archive, Paper 2016/116, https://eprint.iacr.org/2016/116.

Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Fast Reed-Solomon interactive oracle proofs of proximity. In *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*, volume 107 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pp. 14:1–14:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi: 10.4230/LIPIcs.ICALP.2018.14. URL https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ICALP.2018.14.

Nir Bitansky and Alessandro Chiesa. Succinct arguments from multi-prover interactive proofs and their efficiency benefits. In *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pp. 255–272. Springer, 2012. doi: 10.1007/978-3-642-32009-5_16. URL https://www.iacr.org/archive/crypto2012/74170253/74170253.pdf.

Andrew J. Blumberg, Justin Thaler, Victor Vu, and Michael Walfish. Verifiable computation using multiple provers. Cryptology ePrint Archive, Paper 2014/846, 2014. URL https://eprint.iacr.org/2014/846.

Dan Boneh, Yuval Ishai, Amit Sahai, and David J. Wu. Quasi-optimal SNARGs via linear multi-prover interactive proofs. In *Advances in Cryptology – EUROCRYPT 2018*, pp. 222–255. Springer, 2018. doi: 10.1007/978-3-319-78372-7_8. URL https://eprint.iacr.org/2018/133.

Jonah Brown-Cohen, Geoffrey Irving, and Georgios Piliouras. Scalable ai safety via doubly-efficient debate. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 4585–4602, 2024.

Jonah Brown-Cohen, Geoffrey Irving, and Georgios Piliouras. Avoiding obfuscation with prover-estimator debate. *arXiv preprint arXiv:2506.13609*, 2025.

Marie Davidsen Buhl, Jacob Pfau, Benjamin Hilton, and Geoffrey Irving. An alignment safety case sketch based on debate. *arXiv preprint arXiv:2505.03989*, 2025.

Benedikt Bünz, Ben Fisch, and Alan Szepieniec. Transparent SNARKs from DARK compilers. In *Advances in Cryptology – EUROCRYPT 2020*, volume 12105 of *Lecture Notes in Computer Science*, pp. 677–706. Springer, 2020. doi: 10.1007/978-3-030-45721-1_24. URL https://link.springer.com/chapter/10.1007/978-3-030-45721-1_24.

Benedikt Bünz, Ron D. Rothblum, and William Wang. Flock: Fast proving for batch boolean computations. Cryptology ePrint Archive, Paper 2026/1329, 2026. URL https://eprint.iacr.org/2026/1329.

Richard Chang, Benny Chor, Oded Goldreich, Juris Hartmanis, Johan Håstad, Desh Ranjan, and Pankaj Rohatgi. The random oracle hypothesis is false. *Journal of Computer and System Sciences*, 49(1):24–39, 1994. doi: 10.1016/S0022-0000(05)80084-4. URL https://userpages.cs.umbc.edu/chang/papers/roh/.

Binyi Chen, Benedikt Bünz, Dan Boneh, and Zhenfei Zhang. HyperPlonk: Plonk with linear-time prover and high-degree custom gates. In *Advances in Cryptology – EUROCRYPT 2023*, volume 14005 of *Lecture Notes in Computer Science*, pp. 499–530. Springer, 2023a. doi: 10.1007/978-3-031-30617-4_17. URL https://link.springer.com/chapter/10.1007/978-3-031-30617-4_17.

Liyan Chen, Yael Tauman Kalai, and Zoe Xi. How to avoid debate: Scalable ai safety via doubly-efficient interactive proofs. *arXiv preprint arXiv:2607.03561*, 2026.

Megan Chen, Alessandro Chiesa, and Nicholas Spooner. On succinct non-interactive arguments in relativized worlds. In *Advances in Cryptology – EUROCRYPT 2022*, volume 13276 of *Lecture Notes in Computer Science*, pp. 336–366. Springer, 2022. doi: 10.1007/978-3-031-07085-3_12. URL https://iacr.org/archive/eurocrypt2022/132760105/132760105.pdf.

Megan Chen, Alessandro Chiesa, Tom Gur, Jack O'Connor, and Nicholas Spooner. Proof-carrying data from arithmetized random oracles. In *Advances in Cryptology – EUROCRYPT 2023*, volume 14005 of *Lecture Notes in Computer Science*, pp. 379–404. Springer, 2023b. doi: 10.1007/978-3-031-30617-4_13. URL https://eprint.iacr.org/2023/587.

Alessandro Chiesa and Siqi Liu. On the impossibility of probabilistic proofs in relativized worlds. In *11th Innovations in Theoretical Computer Science Conference (ITCS 2020)*, volume 151 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pp. 57:1–57:30. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi: 10.4230/LIPIcs.ITCS.2020.57. URL https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ITCS.2020.57.

Alessandro Chiesa and Eran Tromer. Proof-carrying data and hearsay arguments from signature cards. In *Proceedings of the 1st Symposium on Innovations in Computer Science*, ICS '10, pp. 310–331, 2010.

Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Psi Vesely, and Nicholas P. Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In *Advances in Cryptology – EUROCRYPT 2020*, volume 12105 of *Lecture Notes in Computer Science*, pp. 738–768. Springer, 2020. doi: 10.1007/978-3-030-45721-1_26. URL https://eprint.iacr.org/2019/1047.

Alessandro Chiesa, Ziyi Guan, Shahar Samocha, and Eylon Yogev. Security bounds for proof-carrying data from straightline extractors. In *Theory of Cryptography – TCC 2024*, volume 15365 of *Lecture Notes in Computer Science*, pp. 464–496. Springer, 2024. doi: 10.1007/978-3-031-78017-2_16. URL https://eprint.iacr.org/2023/1646.

Paul Christiano, Buck Shlegeris, and Dario Amodei. Supervising strong learners by amplifying weak experts. *arXiv preprint arXiv:1810.08575*, 2018.

Graham Cormode, Michael Mitzenmacher, and Justin Thaler. Practical verified computation with streaming interactive proofs. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pp. 90–112, 2012. doi: 10.1145/2090236.2090245. URL https://arxiv.org/abs/1105.2003.

Benjamin E. Diamond and Jim Posen. Succinct arguments over towers of binary fields. In *Advances in Cryptology – EUROCRYPT 2025*, volume 15604 of *Lecture Notes in Computer Science*, pp. 93–122. Springer, 2025. doi: 10.1007/978-3-031-91134-7_4. URL https://link.springer.com/chapter/10.1007/978-3-031-91134-7_4.

Uriel Feige and Joe Kilian. Making games short. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pp. 506–516, 1997.

Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: interactive proofs for muggles. *Journal of the ACM (JACM)*, 62(4):27:1–27:64, 2015.

Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems. In *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, pp. 203–225. ACM, 2019.

Justin Holmgren and Ron D. Rothblum. Delegating computations with (almost) minimal time and space overhead. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 124–135. IEEE, 2018. doi: 10.1109/FOCS.2018.00021. URL https://eprint.iacr.org/2018/861.

Geoffrey Irving and Simon Marshall. The need to relativise in debate, June 2025. URL https://www.alignmentforum.org/posts/XycoFucvxAPhcgJQa/the-need-to-relativise-in-debate-1. AI Alignment Forum post.

Geoffrey Irving, Paul Christiano, and Dario Amodei. Ai safety via debate. *arXiv preprint arXiv:1805.00899*, 2018.

Yuval Ishai, Eyal Kushilevitz, and Rafail Ostrovsky. Efficient arguments without short PCPs. In *22nd Annual IEEE Conference on Computational Complexity, CCC 2007*, pp. 278–291, 2007.

Yael Tauman Kalai. Delegating computation via no-signaling strategies. In *Proceedings of the International Congress of Mathematicians (ICM 2018)*, 2018. URL https://gilkalai.wordpress.com/wp-content/uploads/2018/01/cacm-delegation.pdf.

Yael Tauman Kalai, Ran Raz, and Ron D. Rothblum. How to delegate computations: The power of no-signaling proofs. In *Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing*, pp. 485–494. Association for Computing Machinery, 2014. doi: 10.1145/2591796.2591809.

Jan Leike, David Krueger, Tom Everitt, Miljan Martic, Vishal Maini, and Shane Legg. Scalable agent alignment via reward modeling: a research direction. *arXiv preprint arXiv:1811.07871*, 2018.

Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. Constant-round interactive proofs for delegating computation. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pp. 49–62, 2016.

Guy N. Rothblum, Salil Vadhan, and Avi Wigderson. Interactive proofs of proximity: delegating computation in sublinear time. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pp. 793–802, 2013.

Srinath Setty. Spartan: Efficient and general-purpose zkSNARKs without trusted setup. In *Advances in Cryptology – CRYPTO 2020*, volume 12172 of *Lecture Notes in Computer Science*, pp. 704–737. Springer, 2020. doi: 10.1007/978-3-030-56877-1_25. URL https://link.springer.com/chapter/10.1007/978-3-030-56877-1_25.

Justin Thaler. Time-optimal interactive proofs for circuit evaluation. In *Advances in Cryptology – CRYPTO 2013, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pp. 71–89. Springer, 2013. doi: 10.1007/978-3-642-40084-1_5. URL https://link.springer.com/chapter/10.1007/978-3-642-40084-1_5.

Tiancheng Xie, Jiaheng Zhang, Yupeng Zhang, Charalampos Papamanthou, and Dawn Song. Libra: Succinct zero-knowledge proofs with optimal prover computation. In *Advances in Cryptology – CRYPTO 2019, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pp. 733–764. Springer, 2019. doi: 10.1007/978-3-030-26954-8_24. URL https://link.springer.com/chapter/10.1007/978-3-030-26954-8_24.

A Appendix

These appendices contain the preliminaries (Section B), a section on interactive local decoding, which we use to construct encoded-witness IOPs, the main technical tool behind our theoretical result (Section C), a description of our implementation and our experimental results (Section D), and omitted proofs (Section E).

B PRELIMINARIES

Here we introduce some definitions and tools that we use.

Definition B.1 (Language). *A* **language** $\mathcal{L} \subseteq \{0,1\}^*$ *is a set of binary strings. We think of a language as a decision problem: given an* **instance** $\mathbb{x} \in \{0,1\}^*$*, we want to determine whether* $\mathbb{x} \in \mathcal{L}$*.*

Definition B.2 (Relation). *A* **relation** $R \subseteq \{0,1\}^* \times \{0,1\}^*$ *is a set of pairs* $(\mathbb{x}, \mathbb{w})$*, where* $\mathbb{x}$ *is an instance and* $\mathbb{w}$ *is a* **witness** *for* $\mathbb{x}$*. The language associated with* R *is* $L_R \coloneqq \{\mathbb{x} : \exists \mathbb{w}(\mathbb{x}, \mathbb{w}) \in R\}$*.*

Definition B.3 (Oracle and oracle access). *An* **oracle** *is a function* $O : \{0,1\}^* \to \{0,1\}$*. We write that an algorithm has* **oracle access** *to* O *if it if it can send a query* $\mathsf{qry} \in \{0,1\}^*$ *and receive the answer string* $O(\mathsf{qry})$ *in one timestep.*

Definition B.4 (Relativized language). *For an oracle* O*, a* **relativized language** $\mathcal{L}^O$ *is a language whose membership may depend on* O*.*

Definition B.5 (Relativized relation). *For an oracle* O*, a* **relativized relation** R^O *is a relation whose membership may depend on* O*. Its associated relativized language is* L_{R^O}*.*

Definition B.6 (Equality polynomial). *Let* $\mathbb{F}$ *be a finite field and* $h \geq 0$ *an integer. The* **equality polynomial** $\mathsf{eq} : \mathbb{F}^h \times \mathbb{F}^h \to \mathbb{F}$ *is*

$$\mathsf{eq}(r, e) \coloneqq \prod_{i=1}^{h} \big(r_i e_i + (1 - r_i)(1 - e_i)\big).$$

For $b, e \in \{0,1\}^h$*,* $\mathsf{eq}(b, e)$ *is* 1 *if* $b = e$ *and* 0 *otherwise.*

Definition B.7 (Multilinear extension). *Let* $\mathbb{F}$ *be a finite field and* $h \geq 0$ *an integer. We interpret every vector* $v \in \mathbb{F}^{2^h}$ *as a function* $v : \{0,1\}^h \to \mathbb{F}$*. The* **multilinear extension** *of* v *is the unique polynomial* $\widehat{v} : \mathbb{F}^h \to \mathbb{F}$ *of degree at most one in each variable satisfying* $\widehat{v}(b) = v(b)$ *for every* $b \in \{0,1\}^h$*. Explicitly,*

$$\widehat{v}(r) = \sum_{b \in \{0,1\}^h} v(b)\, \mathsf{eq}(r, b),$$

where eq *is the equality polynomial (Definition B.6). For a binary string of length* 2^h*, we view its bits as elements of* $\mathbb{F}$ *and use the same definition. For a vector or string of any other length, the multilinear extension is that of its zero-padding to the next power-of-two length* 2^h*.*

B.1 ORACLE-AIDED COMPUTATION AND ROBUSTNESS

Following Chen et al. (2026), we model the computation that the verifier verifies as an oracle Boolean circuit. Informally, this is a diagram that shows how to obtain an output string from an input string by applying some sequence of OR ($\vee$), AND ($\wedge$), and NOT ($\neg$) operations together with calls to an oracle. More formally, we have:

Definition B.8 (Boolean circuit). *An* n*-input,* m*-output Boolean circuit is a directed acyclic graph with* n *input vertices and an ordered list of* m *output vertices. Input vertices have no incoming edges; every other vertex is a* **gate** *labeled* $\vee$ *or* $\wedge$ *with two incoming edges, or* $\neg$ *with one. Any vertex may be an output. A circuit* C *is given by its* **description** $\langle C \rangle$*, a binary string that lists its gates in a fixed order, each with its label and its incoming edges, followed by the list of output vertices; we refer to gates by their position in this list.*

On input $x \in \{0,1\}^n$*, the value of the* i*th input vertex is the* i*th bit of* x*, and each gate applies its operation to the values of its* **children***, i.e., the vertices with edges going into the gate. The output of the circuit on* x *consists of the values of the output vertices read in order. The* **size** *of a circuit is its number of gates.*

Definition B.9 (Oracle Boolean circuit). *For an oracle* $O : \{0,1\}^* \to \{0,1\}$*, an oracle Boolean circuit* C^O *is a Boolean circuit that may also contain* **oracle gates***. Each oracle gate has an arbitrary number of incoming edges. We view the input to an oracle gate as the string given by the values of its children read in order. The value of an oracle gate is* O *applied to its input string.*

Definition B.10 (Family of circuits). *A family of circuits* $\{C_n\}_{n \in \mathbb{N}}$ *consists of one circuit* C_n *with an* n*-bit input and an* $m(n)$*-bit output for each* n*. It* **computes** *a function* $f : \{0,1\}^* \to \{0,1\}^*$ *if* $C_n(x) = f(x)$ *for every* n *and every* $x \in \{0,1\}^n$*.*

Definition B.11. *Let C^O be an oracle circuit of size S. We define the computation transcript on input x to be the string that lists the values of all the gates in C when run on input x in order (using O to evaluate oracle gates), and denote the computation transcript by*

$$z = \mathrm{Tr}_{C^O}(x) \in \{0,1\}^S.$$

Definition B.12. *Suppose C^O has q_C oracle gates, and consider running C on an input x. We use*

$$\mathcal{Q}_{C^O}(x) = \mathcal{Q} = ((\mathsf{qry}_1, \mathsf{ans}_1), \ldots, (\mathsf{qry}_{\mathsf{q}_C}, \mathsf{ans}_{\mathsf{q}_C}))$$

for the string of queries and answers that C^O makes to O on input x, where qry_i (resp. ans_i) is the input to (resp. output of) oracle gate i, and call $\mathcal{Q}$ a query-answer trace. We write $\mathcal{Q}_{\mathrm{in}}$ (resp. $\mathcal{Q}_{\mathrm{out}}$) for the string of queries (resp. answers) in $\mathcal{Q}$.

Definition B.13 (De-oraclized circuit.). *Let C^O be an oracle circuit with an input of length n and q_C oracle gates. We say that its* **de-oraclized circuit** *is the (non-oracle) circuit C with input $(x, w) \in \{0,1\}^n \times \{0,1\}^{\mathsf{q}_C}$ obtained by replacing, for every $j \in [\mathsf{q}_C]$, oracle gate j with an input vertex with value w_j.*

Recall that the output of a robust oracle circuit should not change if at most a small fraction of answers to its oracle queries are incorrect. Formally, we define robustness as follows, following the definition that Chen, Kalai, and Xi use Chen et al. (2026):

Definition B.14 (Robustness). *Let C^O be an oracle circuit with an input of length n and q_C oracle gates, and let C be its de-oraclized circuit (Definition B.13). For an input (x, w), where $x \in \{0,1\}^n$ and $w \in \{0,1\}^{\mathsf{q}_C}$, and an index $j \in [\mathsf{q}_C]$, let $\mathsf{qry}_j(w)$ denote the input to oracle gate j in C^O when we run C^O on x except with the value of each oracle gate replaced with the corresponding bit in w. For $\varepsilon \in (0,1]$, we say that C^O is* **ε-robust** *if, for every $x \in \{0,1\}^n$, for every $w \in \{0,1\}^{\mathsf{q}_C}$ satisfying*

$$\left|\{j \in [\mathsf{q}_C] : w_j \neq O(\mathsf{qry}_j(w))\}\right| \leq \varepsilon \mathsf{q}_C,$$

we have that $C(x, w) = C^O(x)$.

Definition B.15 (Circuit-satisfiability relation). *The* **circuit-satisfiability relation** *R_{CSAT} is the relation defined as follows. The instance is a triple $\mathbb{x} = (\langle C \rangle, x, a)$, where $\langle C \rangle$ is the description of a Boolean circuit with a public input of length n, a private input of length n_w, an output of length m, and S gates, $x \in \{0,1\}^n$ is a public input, and $a \in \{0,1\}^m$ is a claimed output. The witness is a pair $\mathbb{w} = (w, z) \in \{0,1\}^{n_w} \times \{0,1\}^S$ of a private input and a transcript. Then*

$$(\mathbb{x}, \mathbb{w}) \in R_{\mathsf{CSAT}} \iff z = \mathrm{Tr}_C(x, w) \ \wedge \ C(x, w) = a,$$

where Tr_C is the computation transcript of Definition B.11.

Definition B.16 (Oracle-circuit-evaluation relation). *Fix an oracle O. The* **oracle-circuit-evaluation relation** *R^O_{CVAL} is the relativized relation (Definition B.5) defined as follows. The instance is a triple $\mathbb{x} = (\langle C^O \rangle, x, a)$, where $\langle C^O \rangle$ is the description of an oracle circuit with an input of length n, m outputs, and S gates, $x \in \{0,1\}^n$ is an input, and $a \in \{0,1\}^m$ is a claimed output. The witness is a transcript $z \in \{0,1\}^S$. Then*

$$(\mathbb{x}, z) \in R^O_{\mathsf{CVAL}} \iff z = \mathrm{Tr}_{C^O}(x) \ \wedge \ C^O(x) = a,$$

where Tr_{C^O} is the computation transcript of Definition B.11.

B.2 Multi-prover interactive proofs

A multi-prover interactive proof (MIP) Ben-Or et al. (1988) lets several powerful provers try to convince a weak verifier that an instance has a valid witness.

Definition B.17 (Multi-prover interactive proof system). *A k-prover interactive proof for a relation R is an interactive protocol between a probabilistic polynomial-time verifier $\mathbf{V}$ and k computationally unbounded provers $\mathbf{P}_1, \ldots, \mathbf{P}_k$. The parties receive a common input $\mathbb{x}$, and the honest provers additionally receive a witness $\mathbb{w}$ with $(\mathbb{x}, \mathbb{w}) \in R$.*

The provers may agree on a strategy beforehand but cannot communicate during the protocol: each prover's messages depend only on $\mathbb{x}$, its agreed strategy, and its own interaction with the verifier. At the end of the interaction, the verifier accepts or rejects.

The protocol satisfies the completeness and soundness properties defined below.

We write $\langle \mathbf{P}_1(\mathbb{w}), \ldots, \mathbf{P}_k(\mathbb{w}), \mathbf{V}\rangle(\mathbb{x}) = 1$ when the verifier accepts after interacting with the provers on common input $\mathbb{x}$. The probabilities below are over the verifier's private random bits (we assume without loss of generality that the provers are deterministic).

Definition B.18 (Completeness). *An MIP for R has* **perfect completeness** *if, for every $(\mathbb{x}, \mathbb{w}) \in R$, the honest provers satisfy*

$$\Pr\left[\langle \mathbf{P}_1(\mathbb{w}), \ldots, \mathbf{P}_k(\mathbb{w}), \mathbf{V}\rangle(\mathbb{x}) = 1\right] = 1.$$

Definition B.19 (Soundness). *An MIP for R has* **soundness error** ϵ_{MIP} *if, for every $\mathbb{x} \notin L_R$ and every tuple of possibly computationally unbounded cheating provers $\widetilde{\mathbf{P}}_1, \ldots, \widetilde{\mathbf{P}}_k$,*

$$\Pr\left[\langle \widetilde{\mathbf{P}}_1, \ldots, \widetilde{\mathbf{P}}_k, \mathbf{V}\rangle(\mathbb{x}) = 1\right] \leq \epsilon_{\mathsf{MIP}}(|\mathbb{x}|).$$

Definition B.20 (relativizing MIP). *Fix an oracle O and a relativized relation R^O as in Definition B.5. A* **relativizing MIP** *for R^O is an MIP for R^O in which the verifier and all provers, including cheating provers, have black-box access to the same oracle O. The protocol* **relativizes** *if its completeness, soundness, and efficiency guarantees hold for R^O for every oracle O.*

Remark B.21 (Preprocessing). *An instance may have the form $\mathbb{x} = (\mathbb{x}_{\mathrm{pre}}, \mathbb{x}_{\mathrm{post}})$, where $\mathbb{x}_{\mathrm{pre}}$ is long and is the same for many instances. In our setting $\mathbb{x}_{\mathrm{pre}}$ is the description of a circuit and $\mathbb{x}_{\mathrm{post}}$ is an input together with a claimed output (Definition B.16). An MIP* **with preprocessing** *has a verifier that is a pair $\mathbf{V} = (\mathbf{V}_{\mathrm{pre}}, \mathbf{V}_{\mathrm{post}})$. First $\mathbf{V}_{\mathrm{pre}}$ receives $\mathbb{x}_{\mathrm{pre}}$ and outputs a state $\mathsf{st}_{\mathbf{V}}$. Then $\mathbf{V}_{\mathrm{post}}$ receives $\mathsf{st}_{\mathbf{V}}$ and $\mathbb{x}_{\mathrm{post}}$ and interacts with the provers. The verifier reads $\mathbb{x}_{\mathrm{pre}}$ only in the preprocessing, which is done once and whose state is reused for every $\mathbb{x}_{\mathrm{post}}$. Completeness and soundness are as in Definitions B.18 and B.19.*

We track the following worst-case efficiency parameters of a (relativizing preprocessing) MIP:

- **Round complexity** $\mathsf{r}_{\mathsf{MIP}}$: the number of back-and-forth rounds between the verifier and the provers. Parallel exchanges with different provers count as one round.
- **Communication complexity**: the total number of bits that the verifier and the provers send to each other.
- **Verifier oracle query complexity** $\mathsf{q}_{\mathbf{V}}$: the number of verifier calls to an external oracle O, when present.
- **Verifier preprocessing time** $t_{\mathbf{V}_{\mathrm{pre}}}$: for an MIP with preprocessing (Remark B.21), the running time of $\mathbf{V}_{\mathrm{pre}}$, which bounds $|\mathsf{st}_{\mathbf{V}}|$.
- **Verifier online time** $t_{\mathbf{V}}$: the running time of $\mathbf{V}_{\mathrm{post}}$, with random access to $\mathsf{st}_{\mathbf{V}}$.
- **Prover running time** $t_{\mathbf{P}}$: the maximum, over the honest provers, of that prover's total running time.

In our setting, the honest provers should also be efficient, and the verifier should do substantially less work than evaluating the circuit itself. To verify the evaluation of an oracle circuit C^O, we use an MIP with preprocessing for the relation R^O_{CVAL} of Definition B.16: the parties receive the instance $\mathbb{x} = (\langle C^O\rangle, x, a)$, where $\mathbb{x}_{\mathrm{pre}} = \langle C^O\rangle$ and $\mathbb{x}_{\mathrm{post}} = (x, a)$, the honest provers receive the transcript z, and the claim is that $C^O(x) = a$.

Definition B.22 (Quasi-linear doubly-efficient MIP). *A function $g : \mathbb{N} \to \mathbb{N}$ is* **quasi-linear** *if $g(N) = O(N\,\mathsf{polylog} N)$. An MIP with preprocessing for R^O_{CVAL} is* **quasi-linear doubly-efficient** *if the following hold for every instance $(\langle C^O\rangle, x, a)$, where C^O has an n-bit input, m-bit output, S gates, and q_C oracle gates.*

- *The prover running time $t_{\mathbf{P}}$ is quasi-linear in $n + m + S$.*
- *The verifier preprocessing time $t_{\mathbf{V}_{\mathrm{pre}}}$ is quasi-linear in $n + m + S$.*
- *The verifier online time $t_{\mathbf{V}}$ is quasi-linear in $n + m$.*
- *The verifier oracle query complexity $\mathsf{q}_{\mathbf{V}}$ is sublinear in q_C.*

Remark B.23. *For the verifier's online time to be sublinear in the size of the circuit, we need to preprocess the circuit, unless the circuit has a short description Goldwasser et al. (2015); Cormode et al. (2012); Thaler (2013); Xie et al. (2019) or is evaluated on a batch of inputs Bünz et al. (2026). Intuitively, without preprocessing the verifier must read the entire circuit, since two circuits that differ in one gate can produce different outputs on the same input. Preprocessing is standard in deployed proof systems Chiesa et al. (2020); Setty (2020); Chen et al. (2023a). Since we typically fix a circuit and evaluate it on many inputs, the one-time cost of preprocessing is amortized over all the evaluations.*

B.3 INTERACTIVE ORACLE PROOFS

An interactive oracle proof (IOP), introduced in Ben-Sasson et al. (2016); Reingold et al. (2016), lets a powerful prover convince a weak verifier that an instance has a valid witness by sending long proof strings that the verifier checks at a few locations. The parties may interact over several rounds.

Definition B.24 (Interactive oracle proof). *An interactive oracle proof for a relation R is an interactive protocol between a probabilistic polynomial-time verifier $\mathcal{V}$ and a possibly computationally unbounded prover $\mathcal{P}$. The parties receive a common input $\mathbb{x}$, and the honest prover additionally receives a witness $\mathbb{w}$ with $(\mathbb{x}, \mathbb{w}) \in R$.*

In round i, the prover sends a proof string π_i over a finite alphabet Σ, which is a finite field in our constructions. The verifier has oracle access (Definition B.3) to every proof string sent so far.

The protocol satisfies the completeness and soundness properties defined below.

We write $\langle \mathcal{P}(\mathbb{w}), \mathcal{V} \rangle(\mathbb{x}) = 1$ when the verifier accepts after interacting with the prover on common input $\mathbb{x}$. The probabilities below are over the verifier's private random bits (we assume without loss of generality that the prover is deterministic).

Definition B.25 (Completeness). *An IOP for R has* **perfect completeness** *if, for every $(\mathbb{x}, \mathbb{w}) \in R$, the honest prover satisfies*

$$\Pr\left[\langle \mathcal{P}(\mathbb{w}), \mathcal{V} \rangle(\mathbb{x}) = 1\right] = 1.$$

Definition B.26 (Soundness). *An IOP for R has* **soundness error** ϵ_{IOP} *if, for every $\mathbb{x} \notin L_R$ and every possibly computationally unbounded cheating prover $\widetilde{\mathcal{P}}$,*

$$\Pr\left[\langle \widetilde{\mathcal{P}}, \mathcal{V} \rangle(\mathbb{x}) = 1\right] \leq \epsilon_{\mathsf{IOP}}(|\mathbb{x}|).$$

Definition B.27 (relativizing IOP). *Fix an oracle O and a relativizing relation R^O as in* **??**. *A* **relativizing IOP** *for R^O is an IOP for R^O in which the verifier and prover, including a cheating prover, have black-box access to the same oracle O. The protocol* **relativizes** *if its completeness, soundness, and efficiency guarantees hold for R^O for every oracle O.*

Remark B.28 (Preprocessing). *As in Remark B.21, an IOP* **with preprocessing** *for instances of the form $\mathbb{x} = (\mathbb{x}_{\mathrm{pre}}, \mathbb{x}_{\mathrm{post}})$ has a verifier that is a pair $\mathcal{V} = (\mathcal{V}_{\mathrm{pre}}, \mathcal{V}_{\mathrm{post}})$. First $\mathcal{V}_{\mathrm{pre}}$ receives $\mathbb{x}_{\mathrm{pre}}$ and outputs a state $\mathsf{st}_{\mathcal{V}}$. Then $\mathcal{V}_{\mathrm{post}}$ receives $\mathsf{st}_{\mathcal{V}}$ and $\mathbb{x}_{\mathrm{post}}$ and interacts with the prover. Completeness and soundness are as in Definitions B.25 and B.26.*

We track the following worst-case efficiency parameters of a (relativizing preprocessing) IOP:

- **Proof length** ℓ: the total number of alphabet symbols in the proof strings.
- **Proof query complexity** Q: the total number of verifier queries to the proof strings.
- **Round complexity** $\mathsf{r}_{\mathsf{IOP}}$: the number of back-and-forth rounds between the verifier and the prover. Queries to proof strings do not count as interaction rounds.
- **Verifier oracle query complexity** $\mathsf{q}_{\mathcal{V}}$: the number of verifier calls to the oracle O, when present. These calls are counted separately from proof queries.
- **Verifier preprocessing time** $t_{\mathcal{V}_{\mathrm{pre}}}$: for an IOP with preprocessing (Remark B.28), the running time of $\mathcal{V}_{\mathrm{pre}}$, which bounds $|\mathsf{st}_{\mathcal{V}}|$.
- **Verifier online time** $t_{\mathcal{V}}$: the running time of $\mathcal{V}_{\mathrm{post}}$, with random access to $\mathsf{st}_{\mathcal{V}}$.
- **Prover running time** $t_{\mathcal{P}}$: the honest prover's total running time.

Definition B.29 (Distance). *Let Σ be a finite alphabet and ℓ a positive integer. The* **distance** $\Delta(\pi, \pi')$ *between $\pi, \pi' \in \Sigma^\ell$ is the fraction of positions in which they differ. The distance between π and a set $S \subseteq \Sigma^\ell$ is*

$$\Delta(\pi, S) := \min_{\pi' \in S} \Delta(\pi, \pi'),$$

and $\Delta(\pi, S) := 1$ if S is empty. For $\delta \in (0, 1)$, we say that π is **δ-close** *to S if $\Delta(\pi, S) \leq \delta$, and* **$\delta$-far** *from S otherwise.*

C INTERACTIVE LOCAL DECODING

A locally decodable code lets one recover any symbol of an encoded string from a few queries to a corrupted encoding, and a relaxed locally decodable code Ben-Sasson et al. (2004) lets the decoder

refuse to answer instead. We use an interactive analogue of the latter, in which a prover helps the decoder Ben-Sasson et al. (2016); Reingold et al. (2016).

Definition C.1 (Interactive local decoder). *Let $\Gamma \supseteq \{0,1\}$ be a finite alphabet and let $\mathsf{Enc} : \Gamma^k \to \Sigma^\ell$ be an injective* **encoding***. An* **interactive local decoder** *for Enc is an interactive protocol between a probabilistic polynomial-time* **decoder** $\mathbf{D}$ *and a possibly computationally unbounded prover $\mathcal{P}_{\mathrm{dec}}$. The parties receive a common input $\mathcal{I} = (i_1, \ldots, i_{|\mathcal{I}|}) \in [k]^*$, a list of* **positions***. The decoder has oracle access to a string $\pi \in \Sigma^\ell$, and the honest prover receives $\mathbb{w} \in \Gamma^k$ with $\pi = \mathsf{Enc}(\mathbb{w})$. The prover sends proof strings as in an IOP (Definition B.24), and at the end the decoder outputs $\zeta \in \Gamma^{|\mathcal{I}|} \cup \{\perp\}$.*

We write $\zeta \leftarrow \langle \mathcal{P}_{\mathrm{dec}}(\mathbb{w}), \mathbf{D}^\pi \rangle(\mathcal{I})$ for the output of the decoder, and $\mathbb{w}|_{\mathcal{I}} \coloneqq (\mathbb{w}_{i_1}, \ldots, \mathbb{w}_{i_{|\mathcal{I}|}})$ for the symbols of $\mathbb{w}$ at the positions in $\mathcal{I}$. We track the efficiency parameters of an interactive local decoder as for an IOP (Section B.3), with the queries to π counted as proof queries.

Definition C.2 (Completeness). *An interactive local decoder for Enc has* **perfect completeness** *if, for every $\mathbb{w} \in \Gamma^k$ and every list $\mathcal{I}$ of positions, the honest prover satisfies*

$$\Pr\left[\zeta = \mathbb{w}|_{\mathcal{I}} \mid \zeta \leftarrow \langle \mathcal{P}_{\mathrm{dec}}(\mathbb{w}), \mathbf{D}^{\mathsf{Enc}(\mathbb{w})} \rangle(\mathcal{I})\right] = 1.$$

Definition C.3 (Decoding error). *An interactive local decoder for Enc has* **decoding error** $\epsilon_{\mathrm{dec}} : (0,1) \times \mathbb{N} \to [0,1]$ *if, for every* **proximity parameter** $\delta \in (0,1)$*, every list $\mathcal{I}$ of positions, every $\mathbb{w} \in \Gamma^k$, every $\widetilde{\pi} \in \Sigma^\ell$ that is δ-close to $\mathsf{Enc}(\mathbb{w})$ (Definition B.29), and every possibly computationally unbounded cheating prover $\widetilde{\mathcal{P}}_{\mathrm{dec}}$,*

$$\Pr\left[\zeta \notin \{\mathbb{w}|_{\mathcal{I}}, \perp\} \;\middle|\; \zeta \leftarrow \langle \widetilde{\mathcal{P}}_{\mathrm{dec}}, \mathbf{D}^{\widetilde{\pi}} \rangle(\mathcal{I})\right] \le \epsilon_{\mathrm{dec}}(\delta, |\mathcal{I}|).$$

The encoding Enc is **interactively locally decodable** *if it has an interactive local decoder with perfect completeness and some decoding error.*

Lemma C.4 (Arnon et al. (2025a)). *Let $k \ge 0$ and $M \ge k$. There is a prime field $\mathbb{F}$ of size at least M and at most polynomial in M, and an injective encoding $\mathsf{Enc} : \mathbb{F}^k \to \mathbb{F}^\ell$ with $\ell = O(k)$ and relative distance at least $1/2$, such that Enc is interactively locally decodable and on a list $\mathcal{I}$ of at most M positions its decoder has the following parameters, where $\widetilde{O}(\cdot)$ hides factors polylogarithmic in M and $\delta \coloneqq 1/8$.*

- **Perfect completeness**.
- **Decoding error**: $\epsilon_{\mathrm{dec}}(\delta, |\mathcal{I}|) \le 1/12$.
- **Round complexity**: $O(\log k)$.
- **Proof query complexity**: $O(\log k \cdot \log\log k)$.
- **Proof length**: $O(k)$.
- **Prover time**: $\widetilde{O}(k \cdot (|\mathcal{I}| + 1))$.
- **Verifier time**: $\widetilde{O}(|\mathcal{I}| + 1)$.

D Implementation and Experiments

As a proof of concept, we implement our protocol and evaluate it on an MNIST task inspired by the AI debate experiment of Irving, Christiano, and Amodei Irving et al. (2018). Briefly, in their experiment, there are two provers, or agents, and a verifier which is trained to classify an MNIST image from a few nonzero pixels. A random MNIST image is shown to the agents but not to the verifier. One agent claims the correct label and the other agent claims an incorrect label, e.g., the image contains a 5, and one claims 5 and the other claims 6. They take turns revealing a nonzero pixel in the image to the verifier until four or six pixels are revealed. Then the verifier chooses which agent wins based on which of their labels has the higher logit.

Following their experiment, we train a sparse MNIST classifier using the same architecture; it correctly classifies 59.4% of random six-pixel views. We use its predictions to define an oracle: given a sparse view and a candidate digit y, the oracle outputs 1 if the classifier predicts y and 0 otherwise. Our circuits query this oracle on many sparse views of an image and output whether a majority of the views vote for the candidate digit. We vary the number of sparse views to study how the cost of verification scales with the size of the circuit.

D.1 IMPLEMENTATION

Our implementation is written in Rust.[1] It follows the construction of Section 6, building the encoded-witness IOP of Section 3 from components of Binius Diamond & Posen (2025), Flock Bünz et al. (2026), and Ligero Ames et al. (2017), with one change: the verifier checks the second prover on all proof queries of a run at once, using a random linear combination, rather than on a single query. This avoids the factor Q lost in Lemma E.1, so the protocol needs only as many repetitions as the IOP: 9 or 10 here, instead of 95 to 103. This is sound because the IOP's checks on its proof strings are linear, so a random linear combination pins the second prover to a single proof string before the remaining checks run, as in the linear decommitment of Ishai et al. (2007).

The verifier uses the preprocessing of Remark B.21 only when the circuit's transcript is at least 2^{15} bits long, which here means $\mathsf{q}_C \geq 4{,}096$; on smaller circuits it reads the circuit description directly. The classifier's answers are computed in advance, so both the protocol and direct evaluation of the circuit query the oracle by table lookup. All timings are from AWS Graviton4 instances, one single-threaded process per core, with transparent huge pages enabled.[2] Absolute times are specific to that core; every ratio below is between measurements on the same core type.

D.2 EXPERIMENTS

Concretely, call an MNIST image with six nonzero pixels revealed a view. For each circuit, we fix an MNIST image together with q_C randomly sampled views of the image; the image and views are hardwired into the description of the circuit. On input a candidate digit y, the circuit queries the oracle on each of its q_C views and outputs 1 if more than half of the views vote for y. We can compute the robustness parameters for these circuits exactly: for each digit y, we compute the minimum number of oracle answers that would need to be changed in order to flip the output of the circuit on y. Let m be the minimum of this quantity over all ten digits. Then changing at most $m-1$ oracle answers cannot change the output of the circuit on any input, so the circuit is $(m-1)/\mathsf{q}_C$-robust.

To choose instances to run our circuit on, we scan the MNIST test set in order. For each image, we sample 256 random views and construct the corresponding circuit. We select the first 50 images for which this circuit accepts the correct digit and has robustness parameter at least $1/8$. We use these same 50 images throughout the experiment, and consider

$$\mathsf{q}_C \in \{256, 1024, 4096, 16384, 65536, 262144, 1048576\},$$

sampling additional views of each image as q_C increases. For each image and circuit size, we recompute the robustness parameter and run the complete two-prover protocol with soundness error 2^{-20}. The median robustness parameter remains between 0.31 and 0.32 at every circuit size.

All runs with the honest provers accepted, as expected from perfect completeness. Table 1 reports detailed costs, and Fig. 1 shows how the verifier scales relative to direct evaluation. All values are medians over the 50 images.

In each repetition, the verifier samples t oracle gates, the smallest number with $(1-\varepsilon)^t \leq 1/8$, and it uses $R = 9$ or 10 repetitions to reach soundness error 2^{-20}. Since the robustness parameter remains nearly constant, the median number of oracle queries made by the verifier stays between 57 and 60 as q_C grows from 256 to 1,048,576. At the largest size, the verifier makes $17{,}476\times$ fewer oracle queries than the circuit. The verifier's online time, summed over all repetitions and excluding a one-time preprocessing of the circuit, also grows much more slowly than direct evaluation: it grows from 1.7 ms to 40.3 ms, while direct evaluation grows from 19.7 μs to 89.4 ms. The verifier matches direct evaluation at $\mathsf{q}_C = 262{,}144$, and at $\mathsf{q}_C = 1{,}048{,}576$ it takes $0.45\times$ as long.

At small q_C, the verifier is slower than direct evaluation because protocol overhead dominates: the verifier performs the IOP interaction and opening checks in each repetition, whereas direct evaluation requires only q_C table lookups. As q_C grows, direct evaluation grows linearly, while the verifier's time grows empirically as approximately $q_C^{0.37}$, so their ratio falls by a factor of about 2.4 per fourfold increase in q_C. The verifier time and communication grow because our implementation

[1] Repository URL omitted for anonymity.

[2] These matter at the larger sizes: in a control at $\mathsf{q}_C = 262{,}144$, the verifier's online time was 21 ms with huge pages and 55 ms without.

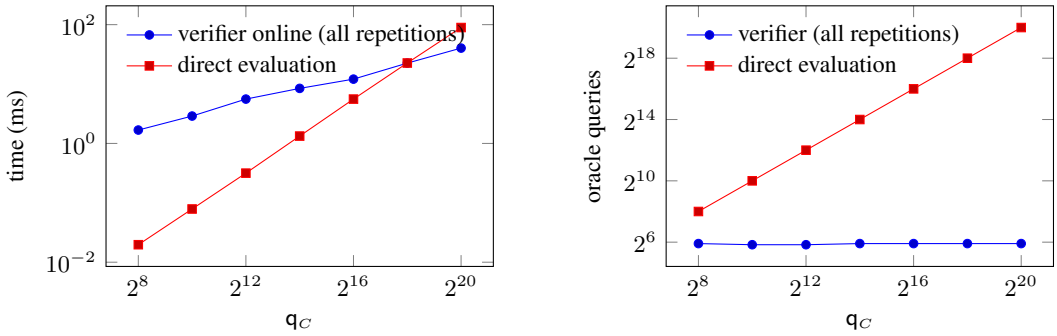

Figure 1: Verifier cost compared with direct circuit evaluation as the number q_C of oracle gates increases. Curves show medians over the 50 selected MNIST images. Left: verifier online time, summed over all repetitions needed for soundness error 2^{-20}, compared with one direct circuit evaluation. Right: total verifier oracle queries over those repetitions, compared with the q_C queries made by direct evaluation. Both axes are logarithmic.

uses the Ligero layout Ames et al. (2017), whose openings have size about $\sqrt{N}$, rather than the polylogarithmic cost in Theorem 7.1.

The verifier's one-time preprocessing takes 31 ms to 24 s, or 97–294× direct evaluation. Prover cost is dominated by P_1: at the largest size, P_1 takes about two hours ($\sim$ 80,000× direct evaluation), while P_2 remains 17–21× direct evaluation across all sizes. Communication reaches 8.7 MiB. The jump in P_1 time at $q_C = 4{,}096$ occurs when the verifier starts using preprocessing.

			multiple of direct eval.			
q_C	**V** oracle queries	direct eval.	**V** online	$\mathbf{P}_1$	$\mathbf{P}_2$	comm.
256	60	19.7 μs	85×	211×	21×	198 KiB
1,024	57	78.7 μs	37×	191×	20×	312 KiB
4,096	57	316 μs	18×	36,000×	19×	873 KiB
16,384	60	1.33 ms	6.4×	44,700×	18×	1.4 MiB
65,536	60	5.58 ms	2.2×	52,800×	18×	2.5 MiB
262,144	60	22.6 ms	1.0×	66,500×	17×	4.6 MiB
1,048,576	60	89.4 ms	0.45×	81,600×	17×	8.7 MiB

Table 1: Cost of the two-prover protocol on the MNIST circuits at seven circuit sizes (medians over 50 images), repeated to get soundness error down to 2^{-20}. "Direct eval." is the time to evaluate the circuit. Verifier online time and the two provers' times are reported as multiples of the time it takes to evaluate the circuit directly; verifier online time excludes the one-time preprocessing cost.

For comparison, we also compile the same IOP into a non-interactive single-prover protocol using the Fiat–Shamir transformation. We compare against Fiat–Shamir rather than against the IOP itself, since the IOP is an idealized model in which the prover's messages are proof strings far longer than the verifier reads, so in practice it is always compiled into a protocol with succinct communication. Fig. 2 compares the two compilations on the first of the 50 images, across six circuit sizes, with both configured for soundness error 2^{-20}. To achieve this soundness error, our two-prover protocol requires 9 repetitions compared to 59 for Fiat–Shamir, resulting in 14.5–17.3× lower verifier time, 6.6–15.7× lower prover time, and 6.6× fewer oracle queries. The tradeoff is that the Fiat–Shamir protocol is non-interactive but provides soundness only against computationally bounded provers, whereas our protocol is interactive and is sound against computationally unbounded provers.

D.3 Limitations

Our experiments are intended as a proof of concept rather than as a realistic scalable-oversight benchmark. The circuits are simple majority circuits over random sparse views, and the 50 images are selected so that their $\mathsf{q}_C = 256$ circuits have robustness parameter at least $1/8$; thus, the experiments do not show that arbitrary MNIST instances or more complex oracle computations satisfy our

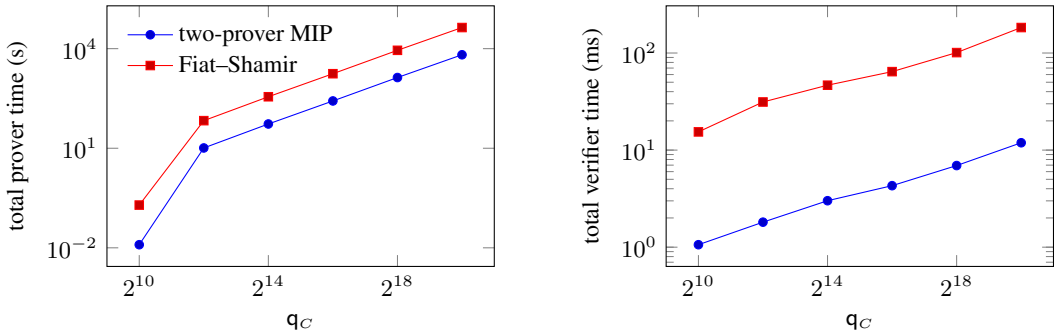

Figure 2: Comparison of our two-prover protocol with the Fiat–Shamir compilation of the same IOP, on the first selected MNIST image and at soundness error 2^{-20}. Left: total prover time (for the MIP, this is the sum of the two provers' time). Right: verifier online time. Times are summed over 9 repetitions for the two-prover protocol and 59 repetitions for Fiat–Shamir.

robustness assumption. Moreover, unlike in the MNIST debate experiment, the sparse views are sampled randomly instead of chosen adversarially (by debating agents).

Because the image and views are hardwired into the circuit, each image and circuit size requires its own preprocessing, so this experiment does not demonstrate amortizing preprocessing across different images. Even though the verifier becomes cheaper than direct evaluation on the largest circuits, the prover cost remains substantial: the first prover takes about two hours on our largest instance. The prover's running time nevertheless scales quasi-linearly with the circuit size, as predicted by Theorem 5.1. The large practical overhead comes from performing field arithmetic over the entire encoded witness in each repetition, whereas direct evaluation of these circuits consists of only q_C table lookups.

E Omitted Proofs

Here we give the omitted proof of Theorem 5.1.

E.1 From IOPs to two provers

To convert a relativizing IOP into a relativizing two-prover MIP, we have the MIP verifier simulate the IOP with one of the provers and check a random answer from this prover against the second prover. This is very similar to the idea behind the standard proof for turning a (non-relativizing) IOP into an MIP (Ben-Sasson et al., 2016, Appendix A).

Here, note that all parties have access to the same fixed oracle O.

Lemma E.1 (IOP-to-MIP conversion). *Suppose a relativizing IOP has perfect completeness, soundness error $\eta < 1$, $\mathsf{r}_{\mathsf{IOP}}$ rounds, and at most $Q \geq 1$ proof queries. Then it has a perfectly complete relativizing two-prover simulation with soundness error at most*

$$1 - \frac{1-\eta}{Q},$$

$O(\mathsf{r}_{\mathsf{IOP}} + Q)$ rounds, and the same verifier oracle-query bound.

Proof of Lemma E.1. **The protocol.** Call the two MIP provers $\mathbf{P}_1$ and $\mathbf{P}_2$. The verifier simulates the IOP verifier with $\mathbf{P}_1$ as the IOP prover, except that the prover does not send long proof strings to the verifier. Instead, whenever the verifier wishes to query a bit in a proof string, he sends the query to the prover, and the prover sends back an answer. The verifier records all the queries made and the answers from $\mathbf{P}_1$. If the IOP verifier accepts and didn't make any queries, the verifier accepts. Otherwise, the verifier picks an IOP query uniformly at random. Suppose the selected query reads location q^* in string j, or all of string j, and $\mathbf{P}_1$'s answer to this query is ζ. The verifier sends

$$(j, q^*, \mu_1, \ldots, \mu_j)$$

to $\mathbf{P}_2$, where $\mu_1, \ldots, \mu_j$ are the IOP verifier messages available when string j was fixed. The second prover returns $\gamma \in \Sigma$. If $\gamma = \zeta$ and the IOP verifier accepts, then the verifier accepts; otherwise the verifier rejects.

Completeness. The honest provers share the same witness and prover randomness. Given the verifier-message prefix, the second prover reproduces the honest IOP prover up to string j and returns the requested symbol. The answers agree, so perfect completeness is preserved.

Soundness. Fix a false instance and the provers' strategies and randomness. For every verifier-message prefix, the second prover defines a string

$$s_j(q) \coloneqq \widetilde{\mathbf{P}}_2(j, q, \mu_1, \ldots, \mu_j).$$

Sending these strings is a legal IOP prover strategy: each string depends only on messages available when it is sent. Let D be the event that some answer from $\mathbf{P}_1$ differs from the corresponding string. Couple the actual simulation to the IOP verifier using these strings and the same verifier randomness. Outside D, the two executions agree. IOP soundness gives

$$1 - \eta \le \Pr[\mathsf{D}] + \Pr[\neg\mathsf{D} \wedge b_{\text{sim}} = 0].$$

On D, either the simulation rejects or the consistency check detects a differing answer with probability at least $1/Q$. Therefore

$$\Pr[b_{\text{MIP}} = 0] \ge \frac{\Pr[\mathsf{D}]}{Q} + \Pr[\neg\mathsf{D} \wedge b_{\text{sim}} = 0] \ge \frac{1 - \eta}{Q}.$$

Averaging over the provers' randomness proves the bound.

Efficiency. Each proof query becomes at most one additional exchange, and the consistency check adds one exchange. The verifier makes the same external oracle calls as the simulated IOP verifier. The second prover reconstructs at most one honest IOP execution. Beyond the IOP's computation, the parties only process query addresses, replies, and one verifier-message prefix.

□

The factor $1/Q$ is inherent to this black-box conversion, which knows nothing about the IOP's proof strings. Our implementation avoids it for the specific IOP of this paper by auditing all query positions at once; see Section D.1.

E.2 Amplification

Lemma E.2 (Amplification with deferred oracle checks). *Let Π be a relativizing two-prover protocol with perfect completeness in which the verifier makes all of its queries to the oracle after the interaction: it records at most t query-answer pairs during the interaction, and after the interaction checks each recorded pair against O. Call these checks* oracle checks*, and call all the other checks that the verifier makes* internal checks*. The verifier accepts if and only if all internal checks and all oracle checks pass.*

Let A be the event that all internal checks pass, and let B be the event that some recorded query-answer pair is incorrect. Suppose that for some integer $k \ge 1$ and some $\delta > 0$, for every false instance and every pair of cheating provers,

$$k \Pr[\neg\mathsf{A}] + \Pr[\mathsf{A} \wedge \mathsf{B}] \ge \delta.$$

Then there is a relativizing two-prover protocol with perfect completeness and soundness error at most $1 - \delta/(2 + \delta)$ that runs k copies of Π but performs the oracle checks of only one copy. In particular, its verifier makes at most t oracle queries in total.

Proof. **The protocol.** The verifier runs k copies of Π sequentially. In each copy it performs the internal checks but not the oracle checks; it only records the query-answer pairs for these checks. If some internal check in some copy fails, the verifier rejects. Otherwise, the verifier picks a copy

$i^* \in [k]$ uniformly at random and performs the oracle checks for this copy. It accepts if and only if all of the oracle checks pass.

Completeness. With honest provers, all internal checks and all oracle checks pass in every copy of Π, so the verifier always accepts.

Soundness. Fix a false instance and a pair of cheating provers, and let r be the probability that the verifier rejects. Let A_i be the event that copy i passes its internal checks, let B_i be the event that some recorded pair in copy i is incorrect, and let $\mathsf{H}_i \coloneqq \mathsf{A}_1 \wedge \cdots \wedge \mathsf{A}_i$ be the event that the first i copies all pass their internal checks (so H_0 always holds).

Fix a copy i and a history h of the first $i-1$ copies, that is, the verifier's randomness and all the messages exchanged in these copies. Applying the hypothesis, we get $k \Pr[\neg\mathsf{A}_i \mid h] + \Pr[\mathsf{A}_i \wedge \mathsf{B}_i \mid h] \geq \delta$. We now average this over all the histories h in which the first $i-1$ copies pass their internal checks. These histories are disjoint and together they make up the event H_{i-1}, so we write $h \in \mathsf{H}_{i-1}$ for them. We get

$$\begin{aligned} k \Pr[\mathsf{H}_{i-1} \wedge \neg\mathsf{A}_i] + \Pr[\mathsf{H}_i \wedge \mathsf{B}_i] &= \sum_{h \in \mathsf{H}_{i-1}} \Big(k \Pr[\neg\mathsf{A}_i \wedge h] + \Pr[\mathsf{A}_i \wedge \mathsf{B}_i \wedge h] \Big) \\ &= \sum_{h \in \mathsf{H}_{i-1}} \Pr[h] \cdot \Big(k \Pr[\neg\mathsf{A}_i \mid h] + \Pr[\mathsf{A}_i \wedge \mathsf{B}_i \mid h] \Big) \\ &\geq \sum_{h \in \mathsf{H}_{i-1}} \Pr[h] \cdot \delta \\ &= \delta \Pr[\mathsf{H}_{i-1}] \\ &\geq \delta(1-r). \end{aligned} \tag{1}$$

We now sum Eq. (1) over all $i \in [k]$, and compare each of the two sums on the left to r.

- The event $\mathsf{H}_{i-1} \wedge \neg\mathsf{A}_i$ says that copy i is the first copy to fail its internal checks. These k events are disjoint, and the verifier rejects in each of them. So $\sum_i \Pr[\mathsf{H}_{i-1} \wedge \neg\mathsf{A}_i] \leq r$.
- Consider the event that $\mathsf{H}_i \wedge \mathsf{B}_i$ holds and $i^* = i$. The verifier rejects in this event: either some later copy fails its internal checks, or the verifier gets to the oracle checks of copy i and finds the incorrect pair. These k events are disjoint, and each has probability $\Pr[\mathsf{H}_i \wedge \mathsf{B}_i]/k$ because i^* is uniform and independent of the interaction. So $\sum_i \Pr[\mathsf{H}_i \wedge \mathsf{B}_i] \leq kr$.

Therefore $kr + kr \geq \delta k(1-r)$, which gives $r \geq \delta/(2+\delta)$.

Efficiency. The verifier performs the oracle checks of a single copy of Π, so it makes at most t oracle queries. Everything else is k runs of Π without its oracle checks.

□

E.3 Proof of Theorem 5.1

By Theorem 4.1, the IOP has soundness error at most $1/3$; all its oracle queries occur after the interaction and check at most t query-answer pairs fixed before these queries begin, and it accepts if and only if its remaining checks pass and every checked pair is correct. Apply Lemma E.1 to it, performing the oracle checks after the interaction. The result is a protocol of the form required by Lemma E.2, whose internal checks are the simulated non-oracle checks together with the consistency check. Fix a false instance and arbitrary cheating provers, and let A and B be as in Lemma E.2. The coupling in Lemma E.1 gives

$$Q\Pr[\neg\mathsf{A}] + \Pr[\mathsf{A} \wedge \mathsf{B}] \geq 2/3.$$

Indeed, on D the consistency check fails with probability at least $1/Q$, so $\Pr[\mathsf{D}] \leq Q \Pr[\mathsf{D} \wedge \neg\mathsf{A}]$. Outside D the consistency check passes, so there A is the event that the simulated non-oracle checks pass, and rejection by the full IOP verifier means a failed internal check or an incorrect recorded oracle answer; hence $\Pr[\neg\mathsf{D} \wedge \text{IOP rejects}] \leq \Pr[\neg\mathsf{D} \wedge \neg\mathsf{A}] + \Pr[\mathsf{A} \wedge \mathsf{B}]$. Adding the two bounds

and using $1 - \eta \le \Pr[\mathsf{D}] + \Pr[\neg\mathsf{D} \wedge \text{IOP rejects}]$ from the proof of Lemma E.1 with $\eta \le 1/3$ gives the displayed inequality. If there are no proof queries, the consistency check is omitted and the same inequality holds. Lemma E.2 with $k = Q$ and $\delta = 2/3$ therefore gives a group of Q copies with soundness error at most $1 - (2/3)/(8/3) = 3/4$ and at most t verifier oracle queries. Run

$$R \coloneqq \left\lceil \frac{\sigma \ln 2}{\ln(4/3)} \right\rceil$$

groups sequentially, using fresh verifier randomness in each group. The verifier makes each group's private copy selection and performs the selected copies' oracle checks only after the last group, and accepts if and only if every group's internal checks and every selected copy's oracle checks pass. Postponing the selections and oracle checks does not change the provers' view, since the selections use private randomness and oracle answers are private, so conditioned on any preceding interaction the next group still satisfies the bound of Lemma E.2. Thus

$$\Pr[\text{accept}] \le (3/4)^R \le 2^{-\sigma}.$$

Perfect completeness is preserved.

Let Q be the proof-query bound of Theorem 4.1, taken to be at least its number of rounds, so that $Q = O(\log(S+n+m) \cdot \log\log(S+n+m))$. One simulation uses $O(Q)$ rounds. Query addresses have $O(\log(S+n+m))$ bits because the IOP has $O(\log(S+n+m))$ rounds and quasi-linear proof length. Forming the addresses costs $O(Q\log(S+n+m))$; copying the verifier-message prefix costs no more than the IOP verifier's online time, $\widetilde{O}(n+m+1/\varepsilon)$ per copy by Theorem 4.1, and the prover's work per copy is its IOP prover time, $\widetilde{O}((S+n+m)/\varepsilon)$. Multiplying the one-copy costs by $RQ = O(\sigma Q)$ gives the stated round complexity and running times; the verifier preprocessing is that of the IOP. For the communication complexity, the long proof strings are never sent, so the verifier and provers exchange only the IOP's short messages and the answers to queries. In the IOP of Theorem 4.1, these amount to $\widetilde{O}(1)$ field elements plus the $\widetilde{O}(1/\varepsilon)$ positions and replies of the decoding. So the communication complexity of each copy is $\widetilde{O}(1/\varepsilon)$ field elements, and over the RQ copies it is $\widetilde{O}(\sigma/\varepsilon)$ field elements. Only one batch is checked in each group, giving at most $Rt = O(\sigma \min\{\mathsf{q}_C, 1/\varepsilon\})$ verifier oracle queries. Each honest prover computes the transcript once, using q_C oracle calls, and reuses it in all copies.