Electronic Colloquium on Computational Complexity, Report No. 134 (2013)

Combinatorial PCPs with Short Proofs[*]

Or Meir[†]

September 24, 2013

Abstract

The PCP theorem (Arora et. al., J. ACM 45(1,3)) asserts the existence of proofs that can be verified by reading a very small part of the proof. Since the discovery of the theorem, there has been a considerable work on improving the theorem in terms of the length of the proofs, culminating in the construction of PCPs of quasi-linear length, by Ben-Sasson and Sudan (SICOMP 38(2)) and Dinur (J. ACM 54(3)).

One common theme in the aforementioned PCP constructions is that they all rely heavily on sophisticated algebraic machinery. The aforementioned work of Dinur (J. ACM 54(3)) suggested an alternative approach for constructing PCPs, which gives a simpler and arguably more intuitive proof of the PCP theorem using combinatorial techniques. However, this combinatorial construction only yields PCPs of polynomial length, and is therefore inferior to the algebraic constructions in this respect. This gives rise to the natural question of whether the proof length of the algebraic constructions can be matched using the combinatorial approach.

In this work, we provide a combinatorial construction of PCPs of length $n \cdot (\log n)^{O(\log\log n)}$, coming very close to the state of the art algebraic constructions (whose proof length is $n \cdot (\log n)^{O(1)}$). To this end, we develop a few generic PCP techniques which may be of independent interest.

It should be mentioned that our construction does use low degree polynomials at one point. However, our use of polynomials is confined to the construction of error correcting codes with a certain simple multiplication property, and it is conceivable that such codes could be constructed without the use of polynomials. In addition, we provide a variant of the main construction that does not use polynomials at all, and has proof length $n^4 \cdot (\log n)^{O(\log\log n)}$. This is already an improvement over aforementioned combinatorial construction of Dinur.

1 Introduction

1.1 Background and Our Results

The PCP theorem [AS98, ALM+98] is one of the major achievements of complexity theory. A PCP (Probabilistically Checkable Proof) is a proof system that allows checking the validity of a claim by reading only a constant number of bits of the proof. The PCP theorem asserts the existence of PCPs of polynomial length for any claim that can be stated as membership in an **NP** language. The theorem has found many applications, most notably in establishing lower bounds for approximation algorithms.

[*]A preliminary version of this work appeared in CCC 2011. This research was supported in part by the Israel Science Foundation (grants No. 460/05 and 1041/08) and by the Adams Fellowship Program of the Israel Academy of Sciences and Humanities.

[†]Institute for Advanced Study, Princeton, NJ. ormeir@ias.edu. Partially supported by NSF grant CCF-0832797.

ISSN 1433-8092

The original proof of the PCP theorem by Arora et al. [AS98, ALM$^+$98] was based on algebraic techniques: Given a claim to be verified, they construct a PCP for the claim by first "arithmetizing" the claim, i.e., reducing the claim to a related "algebraic" claim about polynomials over finite fields, and then constructing a PCP for this algebraic claim. The PCP for the algebraic claim, in turn, requires an arsenal of tools that employ the algebraic structure of polynomials. While those algebraic techniques are very important and useful, it seems somewhat odd that one has to go through algebra in order to prove the PCP theorem, since the theorem itself does not refer to algebra. Furthermore, those techniques seem to give little intuition as to why the theorem holds.

Given this state of affairs, it is an important goal to gain a better understanding of the PCP theorem and the reasons for which it holds. In her seminal paper, Dinur [Din07][1] has made a big step toward achieving this goal by giving an alternative proof of the PCP theorem using a combinatorial approach. Her proof is not only considerably simpler than the original proof, but also seems to shed more light on the intuitions that underlie the theorem.

However, Dinur's PCP construction is still inferior to the algebraic constructions in a few aspects. We believe that it is important to try to come up with combinatorial constructions of PCPs that match the algebraic constructions in those aspects, as this will hopefully advance our understanding of the PCP theorem. Two of those aspects, namely the running time of the verification procedure and the soundness error, have been dealt with in previous works on the subject [DM10, Mei09]. In this work, we deal with a third aspect that concerns the length of the proofs, to be discussed next.

Let L be a language in **NP**, and recall that there is a polynomial-time algorithm V that verifies the membership of a string x in L when given an additional **NP**-witness. Let $t : \mathbb{N} \to \mathbb{N}$ denote the running time of V. The original PCP theorem asserts that in order to verify that $x \in L$, the PCP verifier needs to use a proof of length $\operatorname{poly}(t(|x|))$ and $O(1)$ queries to the proof. However, using algebraic techniques, one can construct PCP verifiers that use a proof of length only $t \cdot \operatorname{poly}\log(t)$ and $O(1)$ queries [BS08, Din07][2]. It is not known whether one can construct such PCPs using a combinatorial approach such as Dinur's[3].

In addition to the proof length of the PCP verifier, we are also interested in its randomness complexity, that is, the number of random bits that are used by the verifier. The randomness complexity of PCP verifiers is usually logarithmic in their proof length, and tends to be more important than the proof length for applications of PCPs. The original PCP theorem, as well as Dinur's proof, yield PCP verifiers that have randomness complexity $O(\log t)$. However, the algebraic constructions of [BS08, Din07] achieve randomness complexity $\log t + O(\log \log t)$.

In this work, we present an (almost) combinatorial construction of PCPs that use proofs of length $t \cdot (\log t)^{O(\log \log t)}$ and have randomness complexity $\log t + O(\log^2 \log t)$, thus coming very close to the state of the art algebraic constructions. Formally, our main result is the following

Theorem 1.1 (Main theorem)**.** *For every time-constructible $t : \mathbb{N} \to \mathbb{N}$ and every language $L \in$ $\mathbf{NTIME}(t)$, there exists a PCP verifier for L with proof length $t \cdot (\log(t))^{O(\log \log t)}$, randomness complexity $\log t + O(\log^2 \log t)$, query complexity $O(1)$, and rejection probability $\Omega(1)$.*

In order to prove Theorem 1.1, we develop a few generic PCP techniques that may be of independent interest, and are discussed in Section 1.2.

[1]We mention that the works of [GS00, DR06] have addressed this goal prior to Dinur's work [Din07], but fell short of obtaining Dinur's result.

[2]In addition, the work of [BKK$^+$13] shows that if one is willing to use n^{ε} queries, then for the specific **NP** language CircuitSat it is possible to construct (non-uniformly) PCPs of length $O(n)$.

[3]We mention that the construction of PCPs that have proof length $t \cdot \operatorname{poly}\log(t)$ uses Dinur's combinatorial techniques in addition to the algebraic techniques. Still, the main part of this construction is algebraic.

Our use of polynomials. As mentioned above, our construction is only "almost" combinatorial. The only exception is that our construction does use low degree polynomials at one point. However, our use of polynomials is a very restricted one, and is confined to the construction of error correcting codes with a certain simple property. Specifically, we only use polynomials to construct a triplet of linear codes (C_A, C_B, C_M) that have the following "multiplication property":

- For every two codewords $c_A \in C_A$ and $c_B \in C_B$, it holds that $c_A \cdot c_B$ is a codeword of C_M, where $c_A \cdot c_B$ is obtained by coordinate-wise multiplication of c_A and c_B. The multiplication is done in the finite field over which the codes are linear.

Such a triplet (C_A, C_B, C_M) can easily be constructed using low-degree polynomials, e.g., by setting C_A and C_B to be Reed-Solomon codes of degree d and setting C_M to be a Reed-Solomon code of degree $2d$. Moreover, if the codes C_A, C_B, and C_M are allowed to have quadratic length, then they can also be constructed without using polynomials [Mei10]. However, in order to achieve the proof length 1.1 we need the codes C_A, C_B, and C_M to have quasi-linear length, and we do not know how to construct such codes without using polynomials. Still, a combinatorial construction of such codes is conceivable.

We also note that we can get purely combinatorial PCPs with proof length $t^4 \cdot (\log(t))^{O(\log\log t)}$ by plugging the latter multiplication codes of [Mei10] into our PCP construction. Moreover, we believe that it can be pushed down further to $t^2 \cdot (\log(t))^{O(\log\log t)}$, but have not verified it. While this result is far from the state-of-the-art PCPs, it is still a significant improvement over the previous combinatorial PCPs of [Din07], which have proof length $n^{O(1)}$, where the power is an unspecified constant that can be expected to be very large. For more details, see Section 7.2.

Extension of our result to PCPPs. We mention that as in previous works in this area, our construction of PCPs can be extended to yield the stronger notion of PCPs of Proximity (PCPPs, [BGH$^+$06, DR06]). For details, see Section 2.3.

1.2 Our techniques

Below, we sketch the main steps of our construction and the main techniques that we use.

Constructing PCPs from linear PCPPs. Our first step is reducing the construction of PCPs to the construction of a simpler object, called linear PCPPs [BHLM09]. Informally, a linear PCPP is a verifier that, when given a linear subspace $W \subseteq \mathbb{F}^n$ and oracle access to a vector $w \in \mathbb{F}^n$, verifies that $w \in W$ by making few queries to w and to an alleged proof. In other words, a linear PCPP is the restriction of the notion of a PCPP [BGH$^+$06, DR06] to the verification of membership in linear subspaces.

We show that *any* construction of a linear PCPP implies a construction of a full-fledged PCP, with a poly-logarithmic loss in the parameters. The construction of the full-fledged PCP is generic, and uses the linear PCPP as a black box. We believe that this construction is interesting in its own right, and may be useful for future works[4].

Our construction of PCPs from linear PCPPs is performed by combining the linear PCPP with the multiplication codes that were discussed in Section 1.1. Intuitively, the multiplication codes allow us to go from verifying linear claims to verifying non-linear claims.

[4]We note that the work of [BS08] has shown a stronger result, namely, that one can construct a full-feldged PCP from a linear PCPP that *can only verify membership in a Reed-Solomon code* (rather than a general linear subspace). However, their construction is signicantly more complicated than ours, and relies heavily on algebraic machinery.

Robustization via tensor product codes. Our next step is to note, following [BGH$^+$06, DR06, BS08, Din07], that it suffices to construct a linear PCPP that makes $\tilde{O}(\sqrt{n})$ queries to its oracle and has proof length $\tilde{O}(n)$. Such a linear PCPP can then be composed with itself for $O(\log\log n)$ times to yield[5] a linear PCPP with a constant number of queries and proof length $n \cdot (\log n)^{O(\log\log n)}$.

In order for us to be able to apply such composition [AS98, ALM$^+$98], the linear PCPP is required to have a property called robustness [BGH$^+$06, DR06]. The robustness property was achieved in several previous works by a technique called "robustization" [BGH$^+$06, DR06] (a.k.a. "alphabet reduction" or "parallelization"). The robustization technique allows one to transform every PCP with a "block-access" property into a robust PCP. Specifically, the latter property may be viewed as follows: It is required that the PCP proof can be partitioned to blocks, such that the PCP verifier always queries only a constant number of the blocks.

In our context, we do not know how to construct a PCP that satisfies the foregoing block-access property. We therefore generalize the robustization technique such that it can be applied to PCPs that satisfy a weaker requirement.

To this end, instead of partitioning the proof to blocks, we arrange the proof coordinates in a matrix. From this point of view, the foregoing block-access property may be viewed as restricting the PCP verifier to reading a constant number of rows of the matrix. We now generalize the robustization technique by allowing the PCP verifier to query *both rows and columns* of the aforementioned matrix, as long as it queries a *constant number* of rows and columns.

Both the standard robustization technique and our generalization use error correcting codes. The standard robustization technique transforms a PCP that has "block-access" into a robust one by encoding each of the blocks by an error correcting code. In our generalization, we transform the "row/column-access PCP" into a robust one by encoding the corresponding matrix via a robust tensor product code [BS06]. This means, roughly, that we first encode the rows of the matrix by an error correcting code, and then encode the columns of the new matrix by an error correcting code.

We stress that this generalization of the robustization method is generic, and may be useful for future constructions of PCPs.

Constructing linear PCPPs that make $\tilde{O}(\sqrt{n})$ queries. It remains to construct a linear PCPP that makes $\tilde{O}(\sqrt{n})$ queries to its oracle, has proof length $\tilde{O}(n)$, and satisfies the relaxed robustization requirement discussed above. The linear PCPP that we construct verifies that $w \in W$ in two stages. In the first stage, the linear subspace $W \subseteq \mathbb{F}^n$ and the vector $w \in \mathbb{F}^n$ are decomposed into subspaces $W_1, \ldots, W_{\tilde{O}(\sqrt{n})} \subseteq \mathbb{F}^{\tilde{O}(\sqrt{n})}$ and vectors $w_1, \ldots, w_{\tilde{O}(\sqrt{n})} \in \mathbb{F}^{\tilde{O}(\sqrt{n})}$, such that $w \in W$ if and only if $w_i \in W_i$ for every i. This is done using a decomposition technique of [Mei09].

In the second stage, the linear PCPP verifies that all the smaller assertions $w_i \in W_i$ hold simultaneously. To this end, we begin by considering the special case in which the subspaces $W_1, \ldots, W_{\tilde{O}(\sqrt{n})}$ are all identical, and show how to handle this simple case using error correcting codes. Then, we show how to decompose the general case to a constant number of instances of the foregoing special case, and handle those instances as before. The latter decomposition is performed via a novel application of routing networks and of the multiplication codes discussed in Section 1.1.

In the author's opinion, the second stage of this construction is the most interesting part of this work.

[5]We mention that after the composition one also needs to apply a query reduction technique and the gap amplification theorem of Dinur.

1.2.1 An alternative construction

In addition to the main construction described above, we also present a variant of this construction that may be more intuitive at the high level, although it is less modular. In this alternative construction, instead of constructing a linear PCPP and then transforming it to a PCP, we construct a PCP directly. In fact, we construct a stronger object, namely, a PCP of Proximity for CircuitEval [BGH+06]. The latter object is a verifier that, when given a boolean circuit $\varphi : \{0,1\}^m \to \{0,1\}$ and oracle access to $x \in \{0,1\}^m$, verifies that x satisfies φ by making few queries to x and to an alleged proof.

As in the main construction, constructing PCPP for CircuitEval with the required parameters boils down to constructing a such PCPP with proof length $\tilde{O}(n)$ and query complexity $\tilde{O}(\sqrt{n})$. As before, this is done in two stages: first, the circuit φ and the assignment x are decomposed to circuits $\varphi_1, \ldots, \varphi_{\tilde{O}(\sqrt{n})}$ of size $\tilde{O}(\sqrt{n})$ and tested assignments $x_1, \ldots, x_{\tilde{O}(\sqrt{n})}$. Then, we verify simultaneously that for every i, the assignment x_i satisfies ψ_i.

The crucial difference between the alternative construction and the main construction is in the way in which the latter simultaneous verification is done. In the main construction, the bulk of the work goes into reducing the general case to the simple case in which $W_1 = \ldots = W_{\tilde{O}(\sqrt{n})}$, while dealing with the simple case is easy. In the alternative construction, reducing the general case to the simple case in which $\varphi_1 = \ldots = \varphi_{\tilde{O}(\sqrt{n})}$ is almost trivial, and the bulk of the work goes into dealing with the latter simple case.

In order to reduce the general case to the simple case, we use the universal circuits. A universal circuit U takes as input a circuit ψ and an assignment y to ψ, and outputs $\psi(y)$. Now, observe that verifying that every assignment x_i satisfies the circuit φ_i is equivalent to verifying that the assignment (φ_i, x_i) satisfies the universal circuit U. Hence, we can reduce the general case to the case where $\varphi_1 = \ldots = \varphi_{\tilde{O}(\sqrt{n})} = U$. A similar idea was used in [Mei09] for different purposes.

It remains to deal with the case where $\varphi_1 = \ldots = \varphi_{\tilde{O}(\sqrt{n})}$. This is done by reducing this case to the linear case. That is, we transform $\varphi_1 = \ldots = \varphi_{\tilde{O}(\sqrt{n})}$ and $x_1, \ldots, x_{\tilde{O}(\sqrt{n})}$ into a linear subspace W and vectors $w_1, \ldots, w_{\tilde{O}(\sqrt{n})}$, such that it suffices to verify that $w_i \in W$ for every i. As noted above, dealing with latter linear case is easy. The reduction to the linear case itself is done using the same technique that is used to reduce PCPs to linear PCPPs in the main construction.

1.3 Organization of this paper

In Section 2, we recall the preliminaries that are required for this work, and state our main technical result (Theorem 2.5). In Section 3, we show how to construct general PCPs based on linear PCPPs. In Section 4, we show our generalization of the robustization technique. In Section 5, we present the construction of linear PCPPs with $\tilde{O}(\sqrt{n})$ queries. Finally, in Section 6, we show how to use the foregoing tools to construct the required PCPs and prove the main theorem (Theorem 1.1). We discuss the alternative construction and the purely combinatorial construction discussed above in Section 7.

2 Preliminaries

2.1 Notation

All logarithms in this paper are in base 2. For any $n \in \mathbb{N}$ we denote $[n] \stackrel{\text{def}}{=} \{1 \ldots, n\}$. For a string $x \in \{0,1\}^n$ and a sequence I of coordinates in $[n]$, we denote by $x_{|I}$ the projection of x to the coordinates in I.

For every collection of functions $g, f_1, \ldots, f_m : \mathbb{N} \to \mathbb{N}$, we denote by $g = \text{poly}(f_1, \ldots, f_m)$ the fact that g is upper bounded by some polynomial in $f_1, \ldots, f_m$. We use the notation

$$g = \text{poly}\log(f_1, \ldots, f_m)$$

as an abbreviation for

$$g = \text{poly}(\log f_1, \ldots, \log f_m).$$

We say that a function $t : \mathbb{N} \to \mathbb{N}$ is time-constructible if it can be computed in time $\text{poly}\log t$, and this definition extends naturally to functions of many variables.

For any two strings $x, y \in \{0,1\}^n$, the relative Hamming distance (or, simply, relative distance) between x and y is the fraction of coordinates on which x and y differ, and is denoted by $\text{dist}(x,y) \stackrel{\text{def}}{=} |\{x_i \neq y_i : i \in [n]\}|/n$. For every set $S \subseteq \{0,1\}^n$ and a string $x \in \{0,1\}^n$ we denote $\text{dist}(x,S) \stackrel{\text{def}}{=} \min_{y \in S}\{\text{dist}(x,y)\}$ (if S is empty, then we define $\text{dist}(x,S) = 1$). We say that x is ε-far from S (respectively, ε-close to S) if $\text{dist}(x,S) > \varepsilon$ (respectively, $\text{dist}(x,S) \leq \varepsilon$).

2.2 Boolean and linear circuits

In this work we consider two types of circuits: boolean circuits and linear circuits.

Boolean circuits. A boolean circuit is the standard type of circuit, whose wires carry boolean values and whose gates compute boolean operations. For convenience, we only allow NOT and AND gates. We will only consider boolean circuits that have a single output. We say that a boolean circuit $\varphi : \{0,1\}^m \to \{0,1\}$ accepts an input $x \in \{0,1\}^m$ if $\varphi(x) = 1$, and otherwise we say that φ rejects x.

Linear circuits. A linear circuit [Val77] is defined with respect to a finite field $\mathbb{F}$. The wires of such a circuit carry elements of $\mathbb{F}$, and a every gate of in the circuit computes a linear combination of its inputs, where the coefficients of the linear combination are elements in $\mathbb{F}$. We will usually consider linear circuits that have multiple outputs.

Note that every output of the circuit is a linear *non-affine* function[6] of the inputs, and that every linear function over $\mathbb{F}$ can be computed by such circuits. We note that in some previous works the definition of linear circuits is little different, and allows the circuits to compute affine functions. The reason we choose not to allow affine functions is that it allows us to state a stronger result (see Theorem 3.1 below).

We say that a linear circuit $\varphi : \mathbb{F}^m \to \mathbb{F}^p$ accepts an input $x \in \mathbb{F}^m$ if $\varphi(x) = \overline{0} \in \mathbb{F}^t$, and otherwise we say that φ rejects x. Observe that the set of inputs accepted by a linear circuit $\varphi : \mathbb{F}^m \to \mathbb{F}^p$ is a linear subspace of $\mathbb{F}^m$ of dimension at least $m - p$. We denote the latter subspace by $\mathbf{SAT}(\varphi)$, and say that φ accepts $\mathbf{SAT}(\varphi)$.

The size and fan-in/fan-out of circuits. For both types of circuits, the size of the circuit is defined to be the number of wires in the circuit. For convenience, we assume without loss of generality that all the circuits have fan-in and fan-out that are upper bounded by 2.

2.3 PCPs - definitions and techniques

In this section, we review the relevant background on PCPs, PCPs of Proximity (PCPPs), and linear PCPPs, as well as recall the techniques of composition, query reduction, and gap amplification.

[6]The function is non-affine because we did not allow constant gates.

2.3.1 PCPs

Recall that a PCP verifier for a language L is an algorithm that verifies a claim of the form $w \in L$ by querying few bits from an auxilary proof π. It is common to define a PCP verifier as an oracle machine that is given oracle access to the proof π and is allowed to make only few queries to this oracle. In this work, we will use a slightly different definition of PCPs from the PCP literature [ALM+98], which allows keeping track of some additional important parameters of the PCP.

More specifically, in the definition of PCPs we view the verifier as a machine that outputs its queries (as a list of coordinates), and a predicate (represented as a circuit) that should be applied to the answers given to those queries. We view the verifier as accepting if the predicate accepts the answers to the queries, and otherwise we view the verifier as rejecting. The advantage of this view is that it allows us to keep track of the complexity of the aforementioned predicate, which is called the decision complexity of the PCP. This view is also needed in order to apply the composition technique. Formally, we use the following definition of a PCP verifier.

Definition 2.1 (PCP verifier, following [ALM+98])**.** Let $L \subseteq \{0,1\}^*$ be a language, and let $r, q, \ell, d : \mathbb{N} \to \mathbb{N}$, $\rho : \mathbb{N} \to (0,1)$. A PCP verifier V for L with randomness complexity r, query complexity q, proof length ℓ, decision complexity d, and rejection ratio ρ, is a probabilistic polynomial time machine that satisfies the following requirements:

1. **Input:** The verifier V takes as input a string w.
2. **Output:** The verifier V outputs a tuple I of coordinates in $[\ell(n)]$ where $|I| \leq q(n)$, and a circuit $\psi : \{0,1\}^{|I|} \to \{0,1\}$ of size at most $d(n)$.
3. **Randomness complexity:** On every input w, and on every sequence of coin tosses, V tosses at most $r(n)$ coins.
4. **Completeness:** For every $w \in L$, there exists a string $\pi \in \{0,1\}^{\ell(n)}$ such that
$$\Pr\left[\psi\left(\pi_{|I}\right) = 1\right] = 1,$$
where ψ and I are generated by the verifier V on input w.
5. **Soundness:** For every string $w \notin L$ and every string $\pi \in \{0,1\}^{\ell(n)}$, it holds that
$$\Pr\left[\psi\left(\pi_{|I}\right) = 0\right] \geq \rho(n),$$
where ψ and I are generated by the verifier V on input w.

Dropping the proof length and query complexity. Definition 2.1 refers to many parameters, and it will be cumbersome to keep track of all of them. Fortunately, it turns out we can avoid keeping track of the proof length and the query complexity:

- **The query complexity:** Clearly, the decision complexity upper bounds the query complexity. Thus, in the rest of this work, we *do not keep track of the query complexity of our PCP verifiers.* The upper bound on the decision complexity of our PCPs will yield the required upper bound on their query complexity.
- **The proof length:** As is common in the PCP literature, we may assume without loss of generality that the proof length ℓ is upper bounded by $2^r \cdot q$, where r and q are the randomness complexity and query complexity of the PCP verifier. This assumption is justified by the fact

that $2^r \cdot q$ is the maximal number of different coordinates that the PCP verifier may query. Thus, in the rest of this work, we *do not keep track of the proof length of our PCP verifiers and only keep track of their randomness complexity and decision complexity.* The upper bounds on the randomness complexity and decision complexity of our PCPs will yield the required upper bound on their proof length.

2.3.2 PCPs of Proximity

A *PCP of Proximity (PCPP)* is a generalization of a PCP that was introduced independently by [BGH$^+$06] and [DR06][7], where the latter used the term "Assignment Testers". A PCPP verifier takes two inputs:

1. An *explicit input* that is given on the verifier input tape, and which the verifier is allowed to read entirely.
2. An *implicit input,* of which the verifier is only allowed to read a small number of bits.

In order to define the languages that such verifiers accept, we need to consider languages of pairs (w, x), where w is the explicit input and x is the implicit input. This motivates the following definition:

Definition 2.2. A pair-language is a relation $PL \subseteq \{0,1\}^* \times \{0,1\}^*$. For every $x \in \{0,1\}^*$, we denote $PL(w) \overset{\text{def}}{=} \{x : (w,x) \in PL\}$.

Definition 2.3. Let PL be a pair language, and let $t : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$. We say that PL is decidable in time t if there exists a Turing machine that on input $(w,x) \in \{0,1\}^* \times \{0,1\}^*$, runs for at most $t(|w|, |x|)$ steps, and accepts if and only if $(w,x) \in PL$.

Using Definition 2.2, we can describe the task of PCPP verifiers as follows: Given $w, x \in \{0,1\}^*$, verify that x is close to $PL(w)$ by reading all of w, and a small number of bits from x and from an additional proof. A PCP verifier can be thought of as a special case of a PCPP verifier in which the implicit input x is empty. Another interesting special case of a PCPP verifier is the case where the explicit input w is empty, in which case we can think of the PCPP verifier as verifying that the string x is close to being a correct claim (i.e., close to a language L) while reading only a constant number of bits of x and of an auxilary proof. The formal definition of a PCPP verifier that we use is the following.

Definition 2.4 (PCPP verifier, variant of [BGH$^+$05])**.** Let PL be a pair-language, let $r, \ell, d : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, $\rho : \mathbb{N} \times \mathbb{N} \to (0,1)$. A PCPP verifier V for PL with randomness complexity r, proof length ℓ, decision complexity d, and rejection ratio ρ, is a probabilistic polynomial time machine that satisfies the following requirements:

1. **Input:** The verifier V takes as a string w of length n and an integer m (represented in unary).
2. **Output:** The verifier V outputs a tuple I of coordinates in $[m + \ell(n,m)]$, and a circuit $\psi : \{0,1\}^{|I|} \to \{0,1\}$ of size at most $d(n,m)$.
3. **Randomness complexity:** On every input (w,m), and on every sequence of coin tosses, V tosses at most $r(n,m)$ coins.

[7]We mention that PCPs of Proximity are related to the previous notion holographic proofs of [BFLS91] and to the work of [Sze99], see [BGH$^+$06] for further discussion.

4. **Completeness:** For every input w and a string $x \in PL(w)$ of length m, there exists a string $\pi \in \{0,1\}^{\ell(n,m)}$ such that
$$\Pr\left[\psi\left((x \circ \pi)_{|I}\right) = 1\right] = 1,$$
where ψ and I are generated by the verifier V on input (w,m). We refer to π as a proof string that convinces V that $x \in PL(w)$.

5. **Soundness:** For every $x \in \{0,1\}^m$ and every string $\pi \in \{0,1\}^{\ell(n,m)}$, it holds that
$$\Pr\left[\psi\left((x \circ \pi)_{|I}\right) = 0\right] \geq \rho(n,m) \cdot \mathrm{dist}\left(x, PL(w)\right),$$
where ψ and I are generated by the verifier V on input (w,m). We refer to x as the tested assignment and to π as the proof string.

Our main PCPP theorem. In this work, we prove the following result, which implies our main theorem (Theorem 1.1) as a special case.

Theorem 2.5 (Main PCPP theorem)**.** *Let $t : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ be a time-constructible function and let PL be a pair-language that is decidable in time t. Then, there exists a PCPP verifier for PL with randomness complexity $\log t + O(\log^2 \log t)$, decision complexity $O(1)$, and rejection ratio $\Omega(1)$.*

Theorem 2.5 is proved in Section 6. We now show that Theorem 2.5 implies the main theorem, restated next.

Theorem (1.1, main theorem, restated)**.** *For every time-constructible $t : \mathbb{N} \to \mathbb{N}$ and every language $L \in \mathbf{NTIME}(t)$, there exists a PCP verifier for L with proof length $t(n) \cdot (\log(t(n)))^{O(\log\log t(n))}$, randomness complexity $\log t + O(\log^2 \log t)$, query complexity $O(1)$, and rejection probability $\Omega(1)$.*

Proof of Theorem 2.5 from Theorem 2.5. Let $t : \mathbb{N} \to \mathbb{N}$ be a time-constructible function and let $L \in \mathbf{NTIME}(t)$ be a language. By definition of $\mathbf{NTIME}(t)$, there exists a Turing machine M such that the following holds: a string $w \in \{0,1\}^*$ is in L if and only if there exists a string $x \in \{0,1\}^*$ such that M accepts the pair (w,x) in time $t(|w|)$. Now, define
$$PL \stackrel{\mathrm{def}}{=} \left\{(w,x) : \ M \text{accepts } (w,x) \text{in time } t(|w|)\right\},$$
let $t_{PL} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ be defined by $t_{PL}(n,m) = t(n)$, and note that PL is a pair language that is decidable in time t (using the machine M). By Theorem 2.5, there exists a PCPP verifier V_{PL} for PL that has randomness complexity $\log t_{PL} + O(\log\log t_{PL})$, decision complexity $O(1)$, and rejection ratio $\Omega(1)$.

We now construct a PCP verifier V for L as follows. For every $w \in L$, a proof π that convinces V that $w \in L$ consists of a string x such that $(w,x) \in PL$ and of a proof π_{PL} that convinces V_{PL} that $x \in PL(w)$. When invoked on input w, the verifier V simply emulates V_{PL}.

It is easy to see that V has the required proof length, randomness complexity and query complexity, and that V satisfies the completenss requirement. To see that V has rejection probability ρ, observe that if $w \notin L$, then every string x will satisfy $\mathrm{dist}(x, L(w)) = 1$ (by definition of relative distance from an empty set), and therefore V will reject with probability at least $\Omega(1) \cdot 1 = \Omega(1)$. ■

2.3.3 Linear PCPPs

We turn to discuss the notion of linear PCPPs [BHLM09]. Informally, a linear PCPP is a PCPP verifier that checks that a vector w satisfies a linear assertion by reading a small part of w, and of an alleged proof. A little more formally, a linear PCPP (over a finite field $\mathbb{F}$) is a verifier that takes as explicit input a linear subspace $W \subseteq \mathbb{F}^m$, read a small number of field elements from a vector $w \in \mathbb{F}^m$ and from an additional string $\pi \in \mathbb{F}^*$, accepts with probability 1 if w belongs to W, and rejects with significant probability if w is far from W. The subspace W is represented by a linear circuit,

The formal definition of a linear PCPP is similar to the definition of the standard PCPP (Definition 2.4), with the following differences:

1. The input to the linear PCPP consists of the field $\mathbb{F}$, and of the linear circuit φ (which represents the linear assertion being tested). The field $\mathbb{F}$ is represented using some representation that allows computing field operations in time $\operatorname{poly}\log|\mathbb{F}|$; for example, if $|\mathbb{F}|$ is of size p^d for prime p, then $\mathbb{F}$ can be represented by p and an irreducible polynomial of degree d over $\mathbf{GF}(p)$.

2. The implicit input x and proof π are vectors over $\mathbb{F}$ rather than boolean strings.

3. The output of the linear PCPP should be a linear circuit over $\mathbb{F}$ rather than a boolean circuit.

We turn to present the formal definition of a linear PCPP.

Definition 2.6 (Variant of [BHLM09]). Let $r, \ell, d : \mathbb{N} \to \mathbb{N}$, $\rho : \mathbb{N} \to (0, 1)$. A linear PCPP verifier V with randomness complexity r, proof length ℓ, decision complexity d, and rejection ratio ρ, is a probabilistic polynomial time machine that satisfies the following requirements:

1. **Input:** The verifier V takes as input a finite field $\mathbb{F}$ and a linear circuit $\varphi : \mathbb{F}^m \to \mathbb{F}^p$ of size n.

2. **Output:** The verifier V outputs a tuple I of coordinates in $[m + \ell(n)]$, and a linear circuit $\psi : \mathbb{F}^{|I|} \to \mathbb{F}^{p'}$ of size at most $d(n)$.

3. **Randomness complexity:** On every finite field $\mathbb{F}$ and input circuit φ, and on every sequence of coin tosses, V tosses at most $r(n)$ coins.

4. **Completeness:** For every $x \in \mathbf{SAT}(\varphi)$, there exists a string $\pi \in \mathbb{F}^{\ell(n)}$ such that
$$\Pr\left[\psi\left((x \circ \pi)_{|I}\right) \text{ accepts}\right] = 1,$$
where ψ and I are generated by the verifier V on input $(\mathbb{F}, \varphi)$. We refer to π as the proof of x, or as the proof that convinces V that φ accepts x.

5. **Soundness:** For every $x \in \mathbb{F}^m$ and every string $\pi \in \mathbb{F}^{\ell(n)}$, it holds that
$$\Pr\left[\psi\left((x \circ \pi)_{|I}\right) \text{ rejects}\right] \geq \rho(n) \cdot \operatorname{dist}\left(x, \mathbf{SAT}(\varphi)\right),$$
where ψ and I are generated by the verifier V on input $(\mathbb{F}, \varphi)$. We refer to x as the tested assignment and to π as the proof string.

Remark 2.7. Note that a linear PCPP would have been a special case of a full-fledged PCPP (Definition 2.4) if it were not for the requirement that it outputs a linear circuit instead of a boolean circuit. Linear PCPPs were defined by [BHLM09], and are also related to the notion of linear inner verifier of [GS06].

The minimal field size. The above definition of linear PCPPs requires a single verifier to be able to work with every finite field, rather than requiring the verifier to work only with a fixed finite field. This is a rather strong definition, and in fact, the actual linear PCPPs that construct are weaker, and can only work with large finite fields. This motivates the definition of the following additional parameter of a linear PCPP.

Definition 2.8. Let $F : \mathbb{N} \to \mathbb{N}$. We say that a linear PCPP has a minimal field size F if, whenever it is invoked on a linear circuit φ of size n over a field $\mathbb{F}$, it is required that $|\mathbb{F}| \geq F(n)$.

2.3.4 Composition of linear PCPPs

In this section we review the composition technique [AS98, ALM+98]. While this technique is usually applied to general PCPs and PCPPs, in this work we apply it only to linear PCPPs. Thus, in this section we describe the specialization of this technique to linear PCPPs rather than its common instantiation.

The composition technique allows reducing the query complexity and decision complexity of a linear PCPP verifier V_{out} (which is referred to as the outer verifier) by "composing" it with another linear PCPP verifier V_{in} (which is referred to as the inner verifier). The result of the composition of V_{out} and V_{in} is a verifier V_{comp} that behaves roughly as follows: when invoked on input circuit φ, the verifier V_{comp} first invokes V_{out} on φ, thus obtaining a circuit ψ, and then invokes V_{in} on ψ, thus obtaining a circuit ξ. The verifier V_{comp} outputs the circuit ξ as its output circuit. It is easy to see that if V_{out} and V_{in} have decision complexities $d_{\text{out}}(n)$ and $d_{\text{in}}(n)$ respectively, then V_{comp} has decision complexity $d_{\text{in}}(d_{\text{out}}(n))$, which will typically be much smaller than both $d_{\text{out}}(n)$ and $d_{\text{in}}(n)$. The cost of applying the composition technique is that the randomness complexity and rejection ratio of V_{comp} are worse than those of V_{out} and V_{in} .

In order to lower bound rejection ratio of V_{comp}, we need the outer verifier V_{out} to be robust. This means, roughly, that if V_{out} is invoked on an assignment x that is far from satisfying φ, then $(x \circ \pi)_{|I}$ will be far on average from satisfying ψ, where I and ψ are the queries and predicate that V_{comp} outputs. More formally, the robustness of a linear PCPP verifier is defined as follows.

Definition 2.9. Let $\rho : \mathbb{N} \to (0, 1)$. A linear PCPP verifier is said to have (expected) robustness $\rho(n)$ if for every input circuit φ of size n, a tested assignment x, and a proof string π, it holds that

$$\mathbb{E}\left[\text{dist}\left((x \circ \pi)_{|I}, \mathbf{SAT}(\psi)\right)\right] \geq \rho(n) \cdot \text{dist}\left(x, \mathbf{SAT}(\varphi)\right)$$

Remark 2.10. Observe that expected robustness is a strengthening of the rejection ratio parameter. That is, if an assignment tester has (expected) robustness ρ, then it must also have rejection ratio at least ρ. Therefore, whenever we state the robustness of an assignment tester, we avoid stating its rejection ratio.

We now turn to state the composition theorem. We will use the following composition theorem, which is a specialization of the composition theorem of [BGH+06, DR06] to linear PCPPs.

Theorem 2.11 (Composition theorem, [BGH+06, DR06]). *Let V_{out} be a linear PCPP verifier with randomness complexity $r_{\text{out}}(n)$, decision complexity $d_{\text{out}}(n)$, minimal field size $F_{\text{out}}(n)$, and* robustness *$\rho_{\text{out}}(n)$, and let V_{in} be a linear PCPP verifier with randomness complexity $r_{\text{in}}(n)$, decision complexity $d_{\text{in}}(n)$, minimal field size $F_{in}(n)$, and rejection ratio $\rho_{\text{in}}(n)$. Then, there exists a linear PCPP verifier V_{comp} that has randomness complexity $r_{\text{out}}(n) + r_{\text{in}}(d_{\text{out}}(n))$, decision complexity $d_{\text{in}}(d_{\text{out}}(n))$, minimal field size* $\max\{F_{\text{out}}(n), F_{in}(n), \}$, *and rejection ratio $\rho_{\text{out}}(n) \cdot \rho_{\text{in}}(d_{\text{out}}(n))$. Furthermore, if V_{in} has* robustness *$\rho_{\text{in}}(n)$, then V_{comp} has* robustness *$\rho_{\text{out}}(n) \cdot \rho_{\text{in}}(d_{\text{out}}(n))$.*

In addition, the computation of V_{comp} (i.e., the sampling of I and ψ) can be performed by a probabilistic polynomial time universal algorithm with black-box access to V_{out} and V_{in} (where the algorithm itself is independent of V_{out} and V_{in}), and that invokes V_{out} once on an input circuit of size n and invokes V_{in} once on an input circuit of size $d_{\mathrm{out}}(n)$.

The proof of Theorem 2.11 is a straightforward adaptation of the proof of the composition theorem in [BGH$^+$06], and we do not include it here.

2.3.5 Decision complexity reduction and rejection ratio amplification of PCPPs

In this section, we review two useful theorems for improving the query complexity, decision complexity, and rejection ratio of PCPPs. The first result, stated next, allows reducing the decision complexity of a PCPP verifier (and hence, its query complexity) to a constant at the expense of increasing its randomness complexity and reducing its rejection ratio.

Theorem 2.12 (Decision complexity reduction, folklore)**.** *Let PL be a language, and let V be a PCPP verifier for PL with randomness complexity $r(n,m)$, decision complexity $d(n,m)$, and rejection ratio $\rho(n,m)$. Then, there exists a PCPP verifier V' with decision complexity $O(1)$, randomness complexity $r(n,m) + O(\log d(n,m))$, and rejection ratio $\rho(n,m)/\operatorname{poly}(d(n,m))$.*

Proof sketch. When invoked on input circuit φ, the verifier V' acts as follows: the verifier V' begins with invoking the verifier V on (w,m), this obtaining an output circuit ψ. Then, the verifier V' transforms ψ into a 3-CNF formula F, using the same technique as that is used in the Karp reduction of CircuitValue to 3Sat. Finally, V' outputs a random clause of F as its output circuit.

We mention that the transformation of ψ to F requires adding auxilary variables to F. Those auxilary variables are added to the proof string of V' as additional proof coordinates. ■

The next result that we review is a corollary of the gap amplification theorem of [Din07] for assignment testers. This theorem allows increasing the rejection ratio of a PCPP verifier to a constant at the expense of increasing its randomness complexity. The original theorem of [Din07] can only be applied to PCPPs with constant decision complexity. By combining this theorem with the above Theorem 2.12, we obtain the following result.

Theorem 2.13 (Rejection ratio amplification, corollary of [Din07, Thm 9.1])**.** *Let PL be a language, and let V be a PCPP verifier for PL with randomness complexity $r(n,m)$, decision complexity $d(n,m)$, and rejection ratio $\rho(n,m)$. Then, there exists a PCPP verifier V' with rejection ratio $\Omega(1)$, decision complexity $O(1)$, and randomness complexity $r(n,m) + O\left(\log \frac{d(n,m)}{\rho(n,m)}\right)$.*

Remark 2.14. The original theorem stated in [Din07] only doubles the rejection ratio of a PCPP verifier while increasing its randomness complexity by only a constant term. The above Theorem 2.13 is obtained in two stages: First, we apply Theorem 2.12 to the PCPP verifier, thus obtaining a PCPP verifier with rejection ratio $\rho(n,m)/\operatorname{poly}(d(n,m))$. Then we apply the original theorem of [Din07] to the latter PCPP verifier for $O\left(\log \frac{d(n,m)}{\rho(n,m)}\right)$ times, thus obtaining a PCPP verifier with rejection ratio $\Omega(1)$ and randomness complexity $r(n,m) + O\left(\log \frac{d(n,m)}{\rho(n,m)}\right)$.

2.4 Error correcting codes

In this section we review the basics of error correcting codes. We then describe a particular family of codes to which we refer as multiplication codes, and which will be used in Sections 3 and 5. We restrict ourselves to linear error correctings codes.

A (linear) code C over a field $\mathbb{F}$ with message length k and block length l is an injective linear function from $\mathbb{F}^k$ to $\mathbb{F}^l$ (where linearity is over $\mathbb{F}$). The rate R_C of the code C is the ratio k/l.

We will sometimes identify C with its image $C(\mathbb{F}^k)$. Specifically, we will write $c \in C$ to indicate the fact that there exists $x \in \mathbb{F}^k$ such that $c = C(x)$. In such case, we also say that c is a codeword of C. The relative distance of a code C is defined to be $\delta_C \overset{\text{def}}{=} \min_{c_1 \neq c_2 \in C} \{\text{dist}(c_1, c_2)\}$. We will also use the notation $\text{dist}(w, C)$ to denote the relative distance of a string $w \in \{0,1\}^l$ from C, and say that w is ε-close (respectively, ε-far) from C if $\text{dist}(w, C) \leq \varepsilon$ (respectively, if $\text{dist}(w, C) > \varepsilon$).

A code $C : \mathbb{F}^k \to \mathbb{F}^l$ is said to be systematic if for every $x \in \mathbb{F}^k$ it holds that $C(x)_{|[k]} = x$. By Gaussian elimination, every linear code may be assumed to be systematic without loss of generality. All the codes in this paper are assumed to be systematic.

We use the the following fact, which asserts that the existence of asymptotically good codes, i.e., codes whose rate and relative distance are constants that are independent of the message length. Furthermore, we can choose those codes such that the codewords can be identified by a linear circuit whose size is almost linear in the message length. There are many constructions of codes that satisfy those properties, and in particular, there are such constructions that do not use algebraic techniques (e.g., the expander codes of [SS96]).

Fact 2.15. *There exist universal constants $R_C > 0$ and $\delta_C > 0$ such that the following holds: for every message length $k \in \mathbb{N}$ and every field $\mathbb{F}$, there exists a systematic code C over $\mathbb{F}$ with message length k, rate at least R_C, and relative distance at least δ_C.*

Furthermore, if we denote the block length of C by l, then there exists a linear circuit φ_C over $\mathbb{F}$ of size $\tilde{O}(k)$ that takes as input vectors in $\mathbb{F}^l$ and accepts a vector if and only if it is a codeword of C (in other words, $\mathbf{SAT}(\varphi_C) = C(\mathbb{F}^k)$). Finally, there exists an algorithm that on input k and $\mathbb{F}$, runs in time polynomial in k and $\log|\mathbb{F}|$, and outputs the linear circuit φ_C of the corresponding code C.

2.4.1 Multiplication codes

As mentioned in the introduction, in this work we use error correcting codes with a special multiplication property, and we construct those codes by using low-degree polynomials. Those multiplication codes are used in the construction of full-fledged PCPs from linear PCPPs in Section 3, and in the construction of linear PCPPs in Section 5. In order to avoid introducing additional notation, we choose the rate and relative distance of the multiplication codes to be the same as the rate and relative distance of the codes of Fact 2.15, although it is not necessary.

Fact 2.16. *Let R_C and δ_C be the universal constants of Fact 2.15. Then, for every message length $k \in \mathbb{N}$ and every field $\mathbb{F}$ such that $|\mathbb{F}| \geq k/R_C$, there exists a triplet (C_A, C_B, C_M) of systematic codes over $\mathbb{F}$ that have the following properties:*

1. *C_A and C_B have message length k (the message length of C_M may be larger).*
2. *C_A, C_B, and C_M all have the same block length l, and have rate at least R_C and relative distance at least δ_C.*
3. ***Multiplication:** For every $c_A \in C_A$ and $c_B \in C_B$ it holds that $c_A \cdot c_B \in C_M$.*

Furthermore, there exists a triplet of linear circuits $(\varphi_A, \varphi_B, \varphi_M)$ over $\mathbb{F}$, where each of the circuits $\varphi_A, \varphi_B, \varphi_M$ is of size at most $\tilde{O}(k)$, takes as input vectors in $\mathbb{F}^l$ and accepts a vector if and only if it is a codeword of C_A, C_B, or C_M respectively. Finally, there exists an algorithm that when given as input k and $\mathbb{F}$, runs in time that is polynomial in k and $\log|\mathbb{F}|$, and outputs the corresponding triplet $(\varphi_A, \varphi_B, \varphi_M)$.

Proof sketch. Choose R_C to be any constant smaller than $1/2$. Take C_A and C_B to be the Reed-Solomon codes of degree $k-1$ and block length $l \stackrel{\text{def}}{=} k/R_C$, and let C_M be the Reed-Solomon code of degree $2k-2$ and block length l. It is easy to show that C_A, C_B, and C_M have the required properties. ■

We refer to (C_A, C_B, C_M) as a triplet of multiplication codes.

Remark 2.17. The construction of the foregoing multiplication codes is the only place in our work in which we use low-degree polynomials. We note that is plausible that such codes can be constructed without the use of algebra. In fact, as we have shown in a previous work [Mei10], such a triplet can be constructed combinatorially if the block length is allowed to be $O(k^2)$. However, in this work we can not afford such a large block length.

Remark 2.18. The notion of multiplication codes can be seen as a variant of "error-correcting pairs", which were introduced by [Köt92, Pel92] in order to generalize the Berlekamp–Welch algorithm. See also Lecture 11 in [Sud01].

2.5 Routing networks

In our construction of PCPs, we use a special kind of graphs called permutation routing networks (see, e.g., [Lei92]). In order to motivate this notion, let us think of the vertices of the graph as computers in a network, such that two computers can communicate if and only if they are connected by an edge. Suppose that there is some set S of computers in the network such that each computer in S needs to send a message to some other computer in S, and furthermore that each computer in S needs to receive a message from exactly one computer in S (in other words, the mapping from source computers to target computers is a permutation). Then, the property of the routing network says that we can route the messages in the network such that each computer in the network forwards exactly one message. Formally, we use the following definition of routing networks.

Definition 2.19. A routing network of order n is a graph $G = (V, E)$ along with a special set of vertices $T \subseteq V$ of size n, such that the following requirement holds: For every permuation σ on T, there exists a set $\mathcal{P}$ of vertex-disjoint paths in G that connect each $v \in T$ to $\sigma(v) \in T$.

Routing networks were studied extensively in the literature of distributed computing, and several constructions of efficient routing networks are known. In particular, we use the following fact on routing networks, which can be proved by several constructions.

Fact 2.20 (see, e.g, [Lei92])**.** *There exists an infinite family of routing networks $\{G_n\}_{n=1}^{\infty}$, the network G_n being of order n, such that the following properties hold.*

1. *G_n has $\tilde{O}(n)$ vertices.*
2. *The edges of G_n can be colored using 4 colors such that no two edges of the same color share a vertex.*
3. *There exists an algorithm that on input n, runs in time $\text{poly}(n)$ and outputs G_n.*

4. *There exists a polynomial time algorithm that when given as input G_n and a permutation $\sigma : T \to T$ outputs a set $\mathcal{P}$ of vertex-disjoint paths that connect each $v \in T$ to $\sigma(v)$.*

3 PCPs and Linear PCPPs.

In this section, we show how to construct a full-fledged PCPP using a linear PCPP as a building block. Specifically, we prove the following result.

Theorem 3.1. *Suppose that there exists a linear PCPP verifier V with randomness complexity $r(n)$, decision complexity $d(n)$, rejection ratio $\rho(n)$, and minimal field size $F(n)$. Then, for every time-constructible $t : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ and every pair-language PL that is decidable in time t, there exists a PCPP verifier V' for PL with randomness complexity $\max\left\{\log t, r\left(\tilde{O}(t)\right)\right\} + O(1)$, decision complexity $O\left(d\left(\tilde{O}(t)\right) \cdot \operatorname{poly}\log\left(F(t), t\right)\right)$, and rejection ratio $\Omega\left(\rho\left(\tilde{O}(t)\right)\right)$.*

This section is divided to three sections: In Section 3.1, we provide a high-level overview of the proof of Theorem 3.1. Then, in Section 3.2, we show how to construct a PCPP verifier for the **NP**-complete language QuadraticEquations by using a linear PCPP. Finally, in Section 3.3, we complete the proof of Theorem 3.1 by showing how to construct a PCPP verifier for any pair-language using a PCPP verifier for the language QuadraticEquations.

The language QuadraticEquations. We recall the definition of QuadraticEquations, which is the pair-language of pairs (E, x) where E is a system of quadratic equations and x is a satisfying assignment of E. Formally, the pair-language QuadraticEquations consists of pairs (E, x) such that:

1. E is a system of quadratic equations over boolean variables of the form
$$\begin{aligned} \sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_{1,i,j} \cdot X_i \cdot X_j &= \beta_1 \\ &\vdots \\ \sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_{p,i,j} \cdot X_i \cdot X_j &= \beta_p \end{aligned}$$
where $X_1, \ldots, X_m$ are boolean variables, and $\alpha_{1,1,1}, \ldots, \alpha_{p,m,m}$ and $\beta_1, \ldots, \beta_p$ are boolean constants.

2. $x \in \{0,1\}^m$ is an assignment to $X_1, \ldots, X_m$ that satisfies the system E.

The system E is represented by the list of the indices of non-zero coefficients $\alpha_{k,i,j}$ and by the vector $\overline{\beta} \stackrel{\text{def}}{=} (\beta_1, \ldots, \beta_p)$. We define the size of E to be the number of non-zero cofficients $\alpha_{k,i,j}$. Note that the size of E is always at least p, or otherwise E would have trivial equations.

Remark 3.2. One may argue that using the pair-language QuadraticEquations is an "algebraic step". However, it is not hard to adjust the proof to work with the **NP**-complete language CircuitSat instead of QuadraticEquations. We chose QuadraticEquations only for technical convenience.

3.1 Proof overview

Intuitively, the idea that underlies the proof of Theorem 3.1 is that the multiplication codes of Fact 2.16 allow us to go from verifying linear claims to verifying non-linear claims. Little more specifically, first note that it suffices to construct a PCPP for QUADRATICEQUATIONS, since it is an **NP**-complete language. Given a system of quadratic equations E and an assignment x to E, the PCP proof is expected to contain a string y that should contain the value that x assigns to every quadratic term that appears in E with a non-zero coefficient $\alpha_{k,i,j}$. Now, E may be viewed as a system of *linear* equations over y, so we can use the linear PCPP verifier V on y to check that y satisfies E. It remains to verify that y is indeed consistent with x, and the point is that this consistency can be verified by using the multiplication codes together with V.

To be more concrete, the verifier V' acts as follows. Let C_A, C_B, and C_M be the triplet of multiplication codes of Fact 2.16. The verifier V' constructs two projections of the string x, denoted a and b, such that $y = a \cdot b$ (where the multiplication is coordinate-wise). The verifier V' expects the proof to contain the encodings $c^a \stackrel{\text{def}}{=} C_A(a)$ and $c^b \stackrel{\text{def}}{=} C_B(b)$, as well as the vector $c' \stackrel{\text{def}}{=} C_A(a) \cdot C_B(b) \in C_M$. Note that the string y is a substring of c', since C_A, C_B, and C_M are systematic. Now, V' invokes V to verify that y and β satisfy the system E, where E is viewed as a linear system over y and over the vector $\overline{\beta} \stackrel{\text{def}}{=} (\beta_1, \ldots, \beta_p)$, and where y is retrieved from c' and $\overline{\beta}$ is taken from the explicit input. V also verifies that c^a, c^b and c' are indeed legal codewords of C_A, C_B and C_M.

It remains for V to verify that the vectors c^a, c^b, and c' in the proof string are constructed as expected. To this end, observe that $C_A(a)$ and $C_B(b)$ are obtained from x via a linear transformation, and hence V' verifies the consistency of c^a and c^b with x simply by invoking V. In order to verify the consistency of c' with c^a and c^b, the verifier V' checks that the vectors c' and $c^a \cdot c^b$ agree on a random coordinate. The soundness of the latter check is proved using the relative distance of C_M.

3.2 PCPPs for quadratic equations from linear PCPPs

In this section, we show how to construct a PCPP verifier for QUADRATICEQUATIONS using a linear PCPP verifier. This construction is the main technical step in the proof of Theorem 3.1. Let V be a linear PCPP verifier with randomness complexity $r(n)$, decision complexity $d(n)$, rejection ratio $\rho(n)$, and minimal field size $F(n)$. We construct a PCPP verifier V' for QUADRATICEQUATIONS with randomness complexity $\max\left\{\log O(n), r\left(\tilde{O}(n)\right)\right\}$, decision complexity $d\left(\tilde{O}(n)\right) \cdot \operatorname{poly}\log\left(F(n), n\right)$, and rejection ratio $\Omega\left(\rho\left(\tilde{O}(n)\right)\right)$.

Fix a system of quadratic equations E of size n over m variables (i.e., E has n non-zero coefficients), and an assignment x to E. Let R_C and δ_C be the universal constants from Fact 2.15, and let $\mathbb{F}$ be a finite field of characteristic 2 and of size at least $\max\{F(n), n/R_C\}$. By Fact 2.16, there exists a a triplet of multiplication codes (C_A, C_B, C_M) of message length n over $\mathbb{F}$ with rate R_C, relative distance δ_C, and block length $l_M = n/R_C$.

In addition, by Fact 2.15, there exists a code C of message length p over $\mathbb{F}$ with rate R_C, relative distance R_C, and block length $l_C = p/R_C$. Let us denote by $c^\beta = C(\beta)$ the encoding of the vector of free coefficients $\beta \stackrel{\text{def}}{=} (\beta_1, \ldots, \beta_p)$ by C.

The proof strings of V'. Suppose that x satisfies E. We describe how to construct the proof string that convinces V' that x satisfies E, or more formally, that $x \in$ QUADRATICEQUATIONS(E).

We begin by arranging the quadratic terms that appear in E with a non-zero coefficient in an arbitrary fixed order, and define $y \in \{0,1\}^n$ to be the string whose i-th bit is the value that

x assigns to the i-th quadratic term $X_u \cdot X_v$. Observe that E induces a system of homogenous linear equations over y and β. In addition, we define two strings $a, b \in \{0,1\}^n$ as follows: The i-th bit of a is the value that x assigns to the first factor of the i-th quadratic term $X_u \cdot X_v$ in E (according to some arbitrary order of the factors), and b is defined similarly for the second factors. Observe that $y = a \cdot b$ (with the multiplication being coordinate-wise).

Now, let us view x, y, a, and b, as vectors over $\mathbb{F}$ by embedding $\{0,1\}$ in $\mathbb{F}$. We define c^a and c^b to be the encodings of a and b via C_A and C_B respectively, and define $c' = c^a \cdot c^b$. Note that $c' \in C_M$, and that since C_A, C_B, and C_M are systematic, it holds that $(c')_{|[n]} = y$.

Finally, we define the proof string that convinces V' that x satisfies E as the concatenation of c^a, c^b, c', and an additional string π to be described next. The string π is a proof string of the linear PCPP verifier V that convinces V that the vector

$$\underbrace{x \circ \cdots \circ x}_{\lfloor l_M/m \rfloor} \circ c^a \circ c^b \circ c' \circ \underbrace{c^\beta \circ \cdots \circ c^\beta}_{\lfloor l_M/l_C \rfloor}$$

satisfies the following linear assertions:

1. c^a, c^b, c', and c^β are legal codewords of C_A, C_B, C_M, and C respectively.

2. The $\lfloor l_M/m \rfloor$ copies of x are indeed equal to each other, and so are the $\lfloor l_M/l_C \rfloor$ copies of c^β.

3. y and β satisfies the system of linear equations induced by E, where y is retrieved from c' and β is retrieved from c^β.

4. The strings a and b are consistent with the first copy of x in π', where a, and b are retrieved from c^a, and c^b respectively. Here, consistency means that the occurences of the value of each x_i in a and b are indeed consistent with x_i.

Note that the way we defined the proof string π', it is a vector over $\mathbb{F}$ rather than a binary string. However, π' can be converted to a binary string by representing every element of $\mathbb{F}$ by a binary string of length $\lceil \log |\mathbb{F}| \rceil$.

The action of V'. We turn to describe the action of V' on input E when given access to an assignment x to E, and to purported proof π' of the form

$$c^a \circ c^b \circ c' \circ \pi$$

The verifier V' performs the following checks, while recycling randomness:

1. V' invokes V to verify that the vector

$$u \overset{\text{def}}{=} \underbrace{x \circ \cdots \circ x}_{\lceil l_M/m \rceil} \circ c^a \circ c^b \circ c' \circ \underbrace{c^\beta \circ \cdots \circ c^\beta}_{\lceil l_M/l_C \rceil}$$

   satisfies the linear assertions listed above. To this end, V uses π as its proof string.

2. V' chooses uniformly at random $k \in [l_M]$ and checks that $(c^a)_k \cdot \left(c^b\right)_k = (c')_k$.

The implementation of the foregoing checks is mostly straightforward. However, few comments are in place:

1. In order to invoke V, the verifier V' needs to construct a linear circuit φ that will serve as the input to V. The circuit φ is a linear circuit that takes as input the vector u defined above and accepts if and only if u satisfies the linear assertions listed above. Moreover, we would like φ to be of size $\tilde{O}(n)$. The only issue that is not straightforward in the construction of such a circuit φ is verifying the first assertion, namely, that c^a, c^b, c', and c^β are legal codewords of C_A, C_B, C_M, and C respectively.
   In order to verify this assertion, V' generates linear circuits φ_A, φ_B, φ_M, φ_C of size $\tilde{O}(n)$ that check membership in the codes C_A, C_B, C_M, and C. Recall that by Facts 2.15 and 2.16, such linear circuits can be computed in polynomial time. The verifier V' then combines the circuits φ_A, φ_B, φ_M, φ_C into the construction of φ in the natural way.

2. The verifier V makes queries to the vector u defined above. However, this vector can not be constructed explicitly by V'. Thus, V' needs to emulate V in a way that will simulate access to u without constructing it explicitly. To this end, V' takes the tuple of queries I that V outputs, and modifies the queries in this tuple in the straightforward way. For example, every query in I that is directed to a coordinate of the $\lceil l_M/m \rceil$ copies of x in u is modified to a coordinate of x in the oracle of V'.
   One exception is the queries to the copies of c^β. The vector c^β is not found in the oracle of V', but is rather constructed explicitly by V'. Thus, in order to emulate the queries to c^β, the verifier V' removes them from the tuple I completely. Then, V' takes the circuit ψ that V outputs, and hard-wires the answers to the latter queries to the corresponding inputs of ψ.

3. The verifier V' is required to output a boolean circuit ψ', while the verifier V outputs a linear circuit ψ over $\mathbb{F}$. To deal with this issue, ψ' emulates ψ by replacing the gates of ψ with boolean circuits that compute the corresponding field operations over $\mathbb{F}$. By assumption, there are such boolean circuits of size $\text{poly}\log|\mathbb{F}|$. In particular, ψ' replaces each input gate of ψ, which takes as input an element of $\mathbb{F}$, with $\log|\mathbb{F}|$ boolean input gates that are fed with the boolean representation of the the latter element of $\mathbb{F}$.

The randomness complexity and decision complexity of V'. We turn to discuss the randomness complexity and decision complexity of V'. To this end, we first upper bound the size of the linear circuit φ. We begin by noting, as discussed above, that the linear circuit φ that is given as input to V is of size $\tilde{O}(n)$ if constructed approporiately.

Next, in order to analyze the randomness complexity of V', observe that V' uses at most $r(|\varphi|)$ random bits in order to emulate V, and uses at most $\log O(n)$ random bits to choose the coordinate k. Since V' uses the same random bits for those two operations, the total randomness complexity of V' is indeed $\max\left\{\log O(n), r\left(\tilde{O}(n)\right)\right\}$.

In order to analyze the decision complexity of V', observe that the circuit ψ' performs $O(|\psi|) = O\left(d(|\varphi|)\right)$ field operations in order to emulate ψ, and an additional one field operation in order to check that $c^a_k \cdot c^b_k = (c')_k$. Since every field opeation can be computed by a circuit of size $\text{poly}\log|\mathbb{F}|$, it follows that the decision complexity of V' is $d\left(|\varphi|\right) \cdot \text{poly}\log|\mathbb{F}| = d\left(\tilde{O}(n)\right) \cdot \text{poly}\log\left(F(n), n\right)$.

The rejection ratio of V'. It remains to analyze the rejection ratio of V'. Fix an assignment x to E and a proof string π' of the form

$$c^a \circ c^b \circ c' \circ \pi.$$

Let ε be the relative distance of x to the nearest satisfying assignment of E. We show that the boolean circuit ψ' rejects $(x \circ \pi')_{|I'}$ with probability at least $\Omega\left(\rho\left(\tilde{O}(n)\right)\right) \cdot \varepsilon$. First, observe that if the vector

$$u \overset{\text{def}}{=} \underbrace{x \circ \cdots \circ x}_{\lfloor l_M/|x| \rfloor} \circ c^a \circ c^b \circ c' \circ \underbrace{c^\beta \circ \cdots \circ c^\beta}_{\lfloor l_M/l_C \rfloor}$$

is $\min\left\{\frac{\delta_C}{40}, \frac{\varepsilon}{10}\right\}$-far from satisfying the linear circuit φ, then by the soundness of V it holds with probability at least

$$\Omega\left(\rho\left(\tilde{O}(n)\right)\right) \cdot \min\left\{\frac{\delta_C}{32}, \frac{\varepsilon}{10}\right\} = \Omega\left(\rho\left(\tilde{O}(n)\right)\right) \cdot \varepsilon$$

that the linear circuit ψ rejects $(u \circ \pi)_{|I}$, in which case ψ' rejects $(x \circ \pi')_{|I'}$ as required. Thus, we may assume that the vector u is $\min\left\{\frac{\delta_C}{40}, \frac{\varepsilon}{10}\right\}$-close to a vector $\overline{u}$ that satisfies φ. The vector $\overline{u}$ must be of the form

$$\overline{u} \overset{\text{def}}{=} \underbrace{\overline{x} \circ \cdots \circ \overline{x}}_{\lfloor l_M/|x| \rfloor} \circ \overline{c}^a \circ \overline{c}^b \circ \overline{c}' \circ \underbrace{\overline{c}^\beta \circ \cdots \circ \overline{c}^\beta}_{\lfloor l_M/l_C \rfloor}$$

where $\overline{x} \in \mathbb{F}^m$, and where $\overline{c}^a$, $\overline{c}^b$, $\overline{c}'$, and $\overline{c}^\beta$ are codewords of C_A, C_B, C_M, and C respectively. Furthermore, the vectors $\overline{a} \overset{\text{def}}{=} \overline{c}^a_{|[q]}$ and $\overline{b} \overset{\text{def}}{=} \overline{c}^b_{|[q]}$ are consistent with $\overline{x}$ in the same sense in which the vectors a and b are supposed to be consistent with x, and the vectors $\overline{y} \overset{\text{def}}{=} (c')_{|[q]}$ and $\overline{\beta} = \overline{c}^\beta_{|[p]}$ satisfy E, where E is viewed as a linear system.

Next, we claim that $\text{dist}(x, \overline{x}) \leq \varepsilon/2$. Otherwise, we would have had that $\text{dist}(u, \overline{u}) > \varepsilon/10$, since the strings $x \circ \cdots \circ x$ and $\overline{x} \circ \cdots \circ \overline{x}$ form 1/5 fraction of u and $\overline{u}$ respectively. It follows that $\overline{x}$ is not a satisfying assignment of E (since x is ε-far from every satisfying assignment). Similarly, it must hold that $\text{dist}(c^\beta, \overline{c}^\beta) \leq \delta_C/2$, since otherwise we would have had that $\text{dist}(u, \overline{u}) > \delta_C/10$. This implies that $c^\beta = \overline{c}^\beta$, since c^β and $\overline{c}^\beta$ are legal codewords of C.

Now, we claim that $\overline{c}^a \cdot \overline{c}^b \neq \overline{c}'$. Otherwise, the string $\overline{y}$ would have contained the correct value of each quadratic term of E under the assignment $\overline{x}$, and therefore in such case the vector $\overline{y}$ and the vector β could not satisfy E as a linear system. It thus holds that $\overline{c}^a \cdot \overline{c}^b \neq \overline{c}'$, and since both $\overline{c}^a \cdot \overline{c}^b$ and $\overline{c}'$ are codewords of C_M, it follows that $\text{dist}\left(\overline{c}^a \cdot \overline{c}^b, \overline{c}'\right) \geq \delta_C$.

Finally, it must hold that each of c^a,c^b,c' are $\frac{\delta_C}{8}$-close to c^a,c^b,c' respectively, or otherwise we would have had $\text{dist}(u, \overline{u}) > \delta_C/40$, since c^a,c^b,c' and $\overline{c}^a$,$\overline{c}^b$,$\overline{c}'$ form 1/5 fraction of u and $\overline{u}$ respectively. This, together with the fact $\text{dist}\left(\overline{c}^a \cdot \overline{c}^b, \overline{c}'\right) \geq \delta_C$, implies that $\text{dist}(c^a \cdot c^b, c') \geq \delta_C/2$. This implies that with probability at least $\delta_C/2 = \Omega\left(\rho\left(\tilde{O}(n)\right)\right)$, it holds that $c^a_k \cdot c^b_k \neq (c')_k$, in which case the boolean circuit ψ' rejects as required. This concludes the analysis of the rejection ratio of V'.

3.3 General PCPPs from PCPPs for quadratic equations

We now complete the proof of Theorem 3.1 by showing how to construct a PCPP verifier for every pair language using a PCPP verifier for QuadraticEquations. The construction is straightforward, and is based on the **NP**-completeness of QuadraticEquations.

Let V_{QE} be the PCPP verifier for QuadraticEquations that was constructed above. Let $t : \mathbb{N} \times \mathbb{N} \times \mathbb{N}$ be a time-constructible function, and let PL be a pair-language that is decidable in time t. We show how to construct a PCPP verifier V_{PL} for PL with randomness complexity $\max\left\{\log t, r\left(\tilde{O}(t)\right)\right\} + O(1)$, decision complexity $O\left(d\left(\tilde{O}(t)\right) \cdot \text{poly}\log\left(F(t), t\right)\right)$, and rejection ratio $\Omega\left(\rho\left(\tilde{O}(t)\right)\right)$. We begin with the following fact, which can be obtained by combining the efficient reduction of Turing machines to circuits of [PF79] with the Karp reduction of CircuitSat to QuadraticEquations.

Fact 3.3. *There exists an algorithm that when given as input a pair (w, m) for $w \in \{0,1\}^*$ and $m \in \mathbb{N}$, runs for $\tilde{O}(t(|w|, m))$ steps, and outputs a system of quadratic equations E_w of size $\tilde{O}(t(|w|, m))$ over boolean variables $X_1, \ldots, X_m$ and $Z_1, \ldots, Z_s$ (for $s = \tilde{O}(t(|w|, m))$), such that E_w satisfies the following property.*

- *A string $x \in \{0,1\}^m$ belongs to $PL(w)$ if and only if there exists a string $z \in \{0,1\}^s$ such that $x \circ z$ is a satisfying assignment of E_w.*

A natural way to construct V_{PL} now would be the following: For every $(w, x) \in PL$, the proof string that convinces V_{PL} that $x \in PL(w)$ would consist of an assignment z for which $x \circ z$ is a satisfying assignment of E_w, and of a proof string that convinces V_{QE} that $x \circ z$ satisfies E_w. The verifier V_{PL} would work by emulating V_{QE} to verify that $x \circ z$ is a satisfying assignment of E_w.

The above construction almost works, but there is still one issue that needs to be resolved: if x is very short compared to z, it could be the case that x is far from $L(w)$ yet $x \circ z$ is close to a satisfying E_w. In order to resolve this issue, we introduce copies $\lceil s/m \rceil$ of x into E_w, such that the "x part" and "z part" in the assignment of E_w are of comparable lengths. More formally, we use the following easy corollary of Fact 3.3.

Corollary 3.4. *There exists an algorithm that when given as input a pair (w, m) for $w \in \{0,1\}^*$ and $m \in \mathbb{N}$, runs for $\tilde{O}(t(|w|, m))$ steps, and outputs a system of quadratic equations E'_w of size $\tilde{O}(t(|w|, m))$ over boolean variables*

$$X_1^1, \ldots, X_m^1, X_1^2, \ldots, X_m^2, \ldots, X_1^{\lceil s/m \rceil}, \ldots, X_m^{\lceil s/m \rceil}, Z_1, \ldots, Z_s,$$

(for $s = \tilde{O}(t(|w|, m))$), such that E_w satisfies the following two properties:

1. *A string $x \in \{0,1\}^m$ belongs to $PL(w)$ if and only if there exists a string $z \in \{0,1\}^s$ such that*

$$\underbrace{x \circ \cdots \circ x}_{\lceil s/m \rceil} \circ z$$

   is a satisfying assignment of E'_w.

2. *An assignment $x^1 \circ \cdots \circ x^{\lceil s/m \rceil} \circ z$ satisfies E'_w, only if that $x^1 = \cdots = x^{\lceil s/m \rceil}$.*

Proof sketch. The system E'_w is constructed from the system E_w of Fact 3.3 by replacing the variables $X_1, \ldots, X_m$ with the variables $X_1^1, \ldots, X_m^1, \ldots, X_1^{\lceil s/m \rceil}, \ldots, X_m^{\lceil s/m \rceil}$, and adding to E_w the equations $X_i^j = X_i^{j+1}$ for every $i \in [m]$ and $j \in [\lceil s/m \rceil - 1]$. ■

We turn to describe the PCPP verifier V_{PL}. For every $(w, x) \in PL$, the proof string π_{PL} that convinces V_{PL} that $x \in PL(w)$ consists of an assignment z for which $\underbrace{x \circ \cdots \circ x}_{\lceil s/m \rceil} \circ z$ is a satisfying assignment of E'_w, and of a proof string π_{QE} that convinces V_{QE} that $\underbrace{x \circ \cdots \circ x}_{\lceil s/m \rceil} \circ z$ satisfies E'_w. Now, when V_{PL} is invoked on input (w, m), tested assignment x, and proof string $\pi_{PL} = z \circ \pi_{\mathrm{QE}}$, the verifier V_{PL} emulates the action of V_{QE} on input E'_w, tested assignment $\underbrace{x \circ \cdots \circ x}_{\lceil s/m \rceil} \circ z$, and proof string π_{QE}, by redirecting the queries of V_{QE} accordingly.

It is easy to verify that V_{PL} has the required randomness complexity and decision complexity. It remains to analyze the rejection ratio of V_{PL}. Let $w \in \{0,1\}^*$ and let $x \in \{0,1\}^m$ be a string that is ε-far from $L(w) \cap \{0,1\}^m$. Then, it holds that $x \circ \cdots \circ x \circ z$ is $\left(\frac{1}{2} \cdot \varepsilon\right)$-far from satisfying E'_w, since the copies of x form at least half of the vector $\underbrace{x \circ \cdots \circ x}_{\lceil s/m \rceil} \circ z$. Thus, the verifier V_{QE} rejects E'_w and $x \circ \cdots \circ x \circ z$ with probability at least $\Omega\left(\rho\left(\tilde{O}\left(t(|w|, m)\right)\right)\right) \cdot \frac{1}{2} \cdot \varepsilon$. This, in turn, implies that V_{PL} rejects w and x with probability at least $\Omega\left(\rho\left(\tilde{O}\left(t(|w|, m)\right)\right)\right) \cdot \varepsilon$, as required. This concludes the proof of Theorem 3.1.

4 A Generalization of the Robustization Technique

Our construction of PCPs uses the composition technique in order to reduce the decision complexity of our linear PCPPs (see details in Section 6). As explained in Section 2.3.4, in order to apply composition, we need our linear PCPPs to be robust. This property was achieved in previous works by a technique called "robustization" [BGH+06, DR06], which can not be applied to our linear PCPPs. In order tor resolve this issue, we generalize the robustization technique such that it can be applied to our linear PCPPs.

This section is organized as follows. In Section 4.1 below, we review the standard robustization method. Then, in Section 4.2, we describe our generalization of the robustization method, and sketch the proof of our generalized robustization theorem. Before providing the full proof, we recall, in Section 4.3, the notion of tensor product codes, which are the main tool that we use in the proof. Finally, in Section 4.4, we provide the full proof of our generalized robustization theorem.

Remark 4.1. Although all the results in this section are stated for linear PCPPs, their analogues for general PCPPs can be proved using roughly the same proofs.

4.1 Background on the robustization technique

The robustization technique [BGH+06, DR06] (a.k.a. "alphabet reduction"), allows transforming every linear PCPP with a certain query structure into a robust linear PCPP. Basically, the linear PCPP should have the property that the tested assignment and proof strings can be partitioned into blocks, such that the verifier always queries only a constant number of blocks. Formally, this property is defined as follows.

Definition 4.2. Let V be a linear PCPP verifier with proof length $\ell(n)$, and let $b \in \mathbb{N}$. We say that V has b-block access if for every circuit φ, the following holds. Let n and m denote the size and input length of φ respectively. Then, there exists a partition $B_1 \uplus \ldots \uplus B_p = [m + \ell(n)]$ such that when V is invoked on φ and outputs a queries tuple I, the queries in I always consists of whole sets $B_{j_1}, \ldots, B_{j_{b'}}$ where $b' \leq b$ (the sets $B_{j_1}, \ldots, B_{j_{b'}}$ may not be distinct). Moreover, we require that each B_j either contains only assignment coordinates (i.e., from $[m]$) or contains only proof coordinates (i.e., from $[m + \ell(n)] - [m]$).

The sets $B_1, \ldots, B_p$ are referred to as the blocks of V with respect to φ, or simply as blocks. We refer to the blocks that contain assignment coordinates as the assignment blocks, and to the blocks that contain proof coordinates to as the proof blocks.

The robustization technique yields the following result.

Theorem 4.3 (Robustization, variant of [Mei09]). *Suppose that there exists a linear PCPP verifier V that has has b-block access, and also has randomness complexity $r(n)$, decision complexity $d(n)$,* *rejection ratio $\rho(n)$, and minimal field size $F(n)$. Then, there exists a linear PCPP verifier V' that has* robustness $\Omega(\rho(n)/b)$*, and also has randomness complexity $O(r(n))$, decision complexity $O(b \cdot d(n))$, and minimal field size $F(n)$.*

The verifier V' is constructed roughly as follows. The proof is expected to contain the encoding of each of the blocks via an error correcting code. The verifier V' emulates the verifier V, but whenever V queries a block, the verifier V' also queries the purported encoding of the block, and verifies that it is indeed the legal encoding of the block.

To see that V' is robust, observe that whenever V rejects, at least one of the blocks that V queries must be modified in order to make V accept. Hence, in order to make V' accept, both the aforementioned block *and its encoding* must be modified. However, modifying the latter encoding to another legal encoding requires flipping many coordinates, and therefore the view of V' is far from any accepting view.

Theorem 4.3 versus the robustization of [BGH$^+$06, DR06, Mei09]. While we did not provide a proof for Theorem 4.3, we note that Theorem 4.3 could be proved for general PCPPs rather than linear PCPPs. Such a theorem would be an extension of [BGH$^+$06, Lemma 2.13] and [DR06, Lemma 3.6], since [BGH$^+$06, Lemma 2.13] only deals with PCPs (rather than PCPPs), [DR06, Lemma 3.6] only deals with the case in which the assignment blocks are singletons, and both results require all proof blocks to be of the same size. Theorem 5.23 of [Mei09] is very similar to the above Theorem 4.3, but makes additional requirements from the original verifier V in order to support a better running time of the resulting verifier V'.

4.2 Robustization of PCPPs with row/column access

Unfortunately, our linear PCPPs do not have b-block access. In order to resolve this issue, we define a relaxation of the block access property, which we call row/column access, and prove a more general robustization theorem that applies to PCPs with the latter property. Informally, the property of row/column access means that the tested assignment and the proof string can be arranged in matrices, such that the verifer always queries at most a constant number of rows and columns of those matrices. The formal definition is as follows.

Definition 4.4. Let V be a linear PCPP verifier with proof length $\ell(n)$. We say that V has b-row/column access if for every circuit φ, there exist matrices M_x and M_π such that the following holds. The entries of M_x and M_π are in one-to-one correspondence with the coordinates in $[m]$ and $[m + \ell(n)] \setminus [m]$ respectively; and the queries tuple I always consists of some whole rows and columns of M_x and M_π, and at most b such whole rows and columns (not necessarily distinct). Moreover, we require that every row and column of M_π is in I with non-zero probability.

If a row (respectively, column) of M_x or M_π is wholly contained in I, we say that V queries this row (respectively, column). We refer to the matrix M_x and M_π as the assignment matrix and proof matrix of V with respect to φ, or simply as the assignment matrix and proof matrix.

Remark 4.5. Note that if a verifier V has b-block access with all the blocks being of the same size, then V may also be viewed as having b-row/column access while querying only rows of the matrices M_x and M_π. In this sense, the property of row/column access is a generalization of the block access property, since it allows the verifier to query both columns and the rows.

Remark 4.6. Note that in some cases it may not be possible to arrange the assignment coordinates in a matrix in a non-trivial way. For example, if m is a prime, then the only way to arrange the assignment in a matrix is to choose a matrix with only one row or column. This issue can be solved by allowing the assignment matrix to be padded with few dummy coordinates. In order to streamline the presentation, we ignore this issue in this work, and refer the reader to [Mei09, Thm 5.23] for a proof that deals with this issue.

Our main result in this section is the following analogue of the known robustization theorem (Theorem 4.3) for linear PCPPs that have row/column access. Unfortunately, this theorem suffers from signifcantly worse parameters than Theorem 4.3. In particular, it increases the decision complexity by an additive term $2^{r(n)}$. Fortunately, this theorem is sufficient for our needs, since in our target application the randomness complexity $r(n)$ will be logarithmic in the decision complexity.

Theorem 4.7 (Robustization for PCPs with row/column access)**.** *Suppose that there exists a linear PCPP verifier V that has b-row/column access and also has randomness complexity $r(n)$, decision complexity $d(n)$, rejection ratio $\rho(n)$, and minimal field size $F(n)$. Then, there exists a linear PCPP verifier V' that has* robustness $\Omega(\rho(n)/b^3)$ *and also has randomness complexity $r(n)+O(1)$, decision complexity $O(b \cdot d(n) + 2^{r(n)})$, and minimal field size $F(n)$.*

We turn to sketch the proof of Theorem 4.7, and provide the full proof in Section 4.4. We first note that we can assume without loss of generality that V does not query the columns of M_x, but rather queries only the rows of M_x (this assumption is justified by Lemma 4.14). Hence, V may be thought of as having block access with respect to M_x, amd therefore queries to M_x can be robustized using the standard robustization technique of Theorem 4.3. In the following sketch, we ignore the queries to M_x and focus on the queries of V to M_π.

It is tempting to try to deal with the queries to M_π by adapting the proof of Theorem 4.3. Such a proof would require the proof string of V' to contain the encoding of every row and column of M_π via an error correcting code. Then, the verifier V' would read the encoding of every row and column that are queried by V and check that it is a legal encoding.

To analyze the robustness of this verifier V', one would argue that that if most of the rows and columns in the proof string V' are close to an accepting view of V', then one could decode them to the corresponding nearest legal codewords, and thus obtain a proof string that convinces V to accept x. Since such a proof string can not exist only if x is far from a satisfying assignment, this would lead to a contradiction and thus establish the soundness of V'.

However, the foregoing argument fails. The reason is that the purported encodings of the rows and columns of M_π may be *inconsistent*. That is, the proof string might contain encodings whose encoded messages do not agree on the intersections of the rows and columns. Such an inconsistency will fail the above argument, because in such a case there would not be a decode the proof string in a way that is consistent with both the rows and the columns.

In order to resolve this issue, we use tensor prodcut codes, which are reviewed below in Section 4.3. In short, given a code C with block length l, the tensor code $C \otimes C$ is the code whose codewords are exactly the $l \times l$ matrices N such that every row and every column of N is a codeword of C. The key property of tensor codes that is important for us the following: Let N' be a matrix that is close to a codeword N of $C \otimes C$. Then, for most of the rows and columns of N', the closest codeword of C is the corresponding row or column of N. In particular, this means that the messages encoded by most rows and columns of N' are consistent with each other, which solves the consistency problem we had before.

We now construct the verifier V' as follows. The proof string of V' is expected to contain the encoding N_π of M_π via a tensor code $C \otimes C$. It can be shown that N_π contains the encoding via C of every row and column of M_π. The verifier V' emulates the original verifier V, and whenever the original verifier V queries a row or a column of M_π, the verifier V' also queries the encoding of this row or column in N_π and verifies that this encoding is correct.

In order to show that V' is robust, fix a purported encoding N'_π of M_π that are given in the proof string of V'. We first note that if N'_π is close to a legal codeword of $C \otimes C$, then we can use the foregoing property of tensor codes to argue that most of the rows and columns of N'_π are consistent, and the analysis discussed above would go through.

It remains to verify that N'_π is close to be a legal codeword of $C \otimes C$. To this end, we require $C \otimes C$ to have a *robust tester*. This means, roughly, that there exists a *robust* linear PCPP verifier that is only capable of verifying the assertion that a matrix is close to a legal codeword of $C \otimes C$ (rather than verifying any linear assertion). Tensor codes satisfying this requirement can be constructed combinatorially using a few methods (see [BS06, DSW06, BV09b, BV09a, Vid11]). The verifier V' will invoke the robust tester of $C \otimes C$ to verify that N'_π is close to $C \otimes C$, and then proceed as before.

Remark 4.8. The foregoing description oversimplifies things a little. In particular, note that the foregoing property of the tensor codes only guarantees that *most of* the rows and columns are consistent, and not all of them. Thus, in general it could be the case that V always queries the inconsistent rows and columns. In order to handle this issue, we begin the construction by modifying V such that queries rows and columns according to the uniform distribution, and then proceed as before. The latter transformation is implemented by a vairant of the expander-replacement technique of [PY91].

4.3 Tensor product codes

In this section, we review the notion of tensor product of codes, and present some of its properties. See [MS88] and [Sud01, Lect. 6 (2.4)] for the basics of this subject. While in the overview of Section 4.2 we focused on a tensor code $C \otimes C$, which is obtained by taking the tensor product of a code C with itself, in the following discussion we consider a more general definition that allows taking the tensor product of two different codes.

Definition 4.9 (Tensor codes)**.** Let $C_{\text{row}} : \mathbb{F}^{k_{\text{row}}} \to \mathbb{F}^{\ell_{\text{row}}}$, $C_{\text{col}} : \mathbb{F}^{l_{\text{col}}} \to \mathbb{F}^{l_{\text{col}}}$ be codes. The tensor product code $C_{\text{row}} \otimes C_{\text{col}}$ is a code of message length $k_{\text{row}} \cdot k_{\text{col}}$ and block length $l_{\text{row}} \cdot l_{\text{col}}$ that encodes a message $x \in \mathbb{F}^{k_{\text{row}} \cdot k_{\text{col}}}$ as follows: In order to encode x, we first view x as a $k_{\text{col}} \times k_{\text{row}}$ matrix, and encode each of its rows via the code C_{row}, resulting in a $k_{\text{col}} \times l_{\text{row}}$ matrix x'. Then, we encode each column of x' via the code C_{col}. The resulting $l_{\text{col}} \times l_{\text{row}}$ matrix is defined to be the encoding of x via $C_{\text{row}} \otimes C_{\text{col}}$.

The following fact lists some of the basic and standard properties of the tensor product operation.

Fact 4.10. *Let $C_{\text{row}} : \mathbb{F}^{k_{\text{row}}} \to \mathbb{F}^{l_{\text{row}}}$, $C_{\text{col}} : \mathbb{F}^{k_{\text{col},}} \to \mathbb{F}^{l_{\text{col}}}$ be codes with rates $R_{\text{row}}, R_{\text{col}}$ and relative distances $\delta_{\text{row}}, \delta_{\text{col}}$ respectively. The following properties hold:*

1. *The code $C_{\text{row}} \otimes C_{\text{col}}$ has rate $R_{\text{row}} \cdot R_{\text{col}}$ and relative distance $\delta_{\text{row}} \cdot \delta_{\text{col}}$.*

2. *An $l_{\text{col}} \times l_{\text{row}}$ matrix N over $\mathbb{F}$ is a codeword of $C_{\text{row}} \otimes C_{\text{col}}$ if and only if all the rows of N are codewords of C_{col} and all the columns of N are codewords of C_{col}.*

3. *If C_{row} and C_{col} are systematic, then the message encoded by a codeword N of $C_{\text{row}} \otimes C_{\text{col}}$ is the top-left $k_{\text{col}} \times k_{\text{row}}$ submatrix of N.*

The next claim formalizes the property of tensor product codes that was mentioned in Section 4.2, and is used to ensure the consistency of the rows and the columns in the analysis of the robustness of V'.

Claim 4.11. *Let $C_{\text{row}} : \mathbb{F}^{k_{\text{row}}} \to \mathbb{F}^{\ell_{\text{row}}}$, $C_{\text{col}} : \mathbb{F}^{l_{\text{col}}} \to \mathbb{F}^{l_{\text{col}}}$ be codes with relative distances $\delta_{\text{row}}, \delta_{\text{col}}$ respectively, and let N' be an $l_{\text{col}} \times l_{\text{row}}$ matrix over $\mathbb{F}$ that is ε-close to a codeword N of $C_{\text{row}} \otimes C_{\text{col}}$. For every $i \in [l_{\text{col}}]$, we say that the i-th row of N'* **decodes well** *if the codeword of C_{row} that is closest to the i-th row of N' is the i-th row of N. The definition of a column of N' that decodes well is similar. Then, it holds that at least $(1 - 2\varepsilon/\delta_{\text{row}})$ fraction of the rows of N' and at least $(1 - 2\varepsilon/\delta_{\text{col}})$ fraction of the columns of N' decode well.*

Proof. We prove the claim only for the rows of N', and the proof for the columns is similar. Suppose, for the sake of contradiction, that more than $2\varepsilon/\delta$ fraction of the rows of N' do not decode well. Observe that each row of N' that does not decode well must be $\delta_{\text{row}}/2$-far from the corresponding row of N (or otherwise the corresponding row of N would have been the closest codeword of C_{row}). This implies that the relative distance of N' from N must be more than $(2\varepsilon/\delta_{\text{row}}) \cdot (\delta_{\text{row}}/2) > \varepsilon$, thus contradicting the assumption that N' is ε-close to N. It follows that at most $2\varepsilon/\delta_{\text{row}}$ fraction of the rows of N' do not decode well, as required. ■

As explained in Section 4.2, the proof of Theorem 4.7 uses tensor codes that have a robust tester. Those tensor codes also have the good properties of the good codes of Fact 2.15. The properties of those tensor codes are summarized by the following fact, which can be established using the works of [BS06, DSW06, BV09b, BV09a, Vid11].

Fact 4.12. *Let $R_C, \delta_C > 0$ be the universal constants of Fact 2.15, and let $\rho_C > 0$ be a universal constant. For every fiinit field $\mathbb{F}$, there exists an infinite sequence of systematic linear codes $\{C_k\}_{k=1}^{\infty}$ over $\mathbb{F}$ that satisfies the following properties:*

1. *For every $k \in \mathbb{N}$, the code C_k has message length k, rate R_C and relative distance δ_C.*
2. *For every $k \in \mathbb{N}$, there exists a linear circuit φ_k over $\mathbb{F}$ of size $O(k)$ that accepts a vector if and only if it is a codeword of C_k (in other words, $\mathbf{SAT}(\varphi_k) = C_k(\mathbb{F}^k)$). Furthermore, there exists a algorithm that on input k and $\mathbb{F}$, runs in time that is polynomial in k and $\log|\mathbb{F}|$, and outputs the linear circuit φ_k of the corresponding code C_k over $\mathbb{F}$.*
3. ***Robust testing of $C_{k_1} \otimes C_{k_2}$:*** *There exists a probabilistic algorithm V_T that behaves as follows. Let $\mathbb{F}$ be a finite field, let $k_1, k_2 \in \mathbb{N}$ and let l_1 and l_2 be the block lengths of C_{k_1} and C_{k_2}. When invoked on input $k_1, k_2, \mathbb{F}$, the algorithm runs in time $\text{poly}(k_1, k_2, \log|\mathbb{F}|)$, tosses at most $\log(\max\{k_1, k_2\}) + O(1)$ coins, and outputs a queries tuple I of coordinates in $[l_1 \cdot l_2]$ and a linear circuit ψ that satisfy the following requirements:*
   (a) ***Completeness:*** *for every codeword $N \in C_{k_1} \otimes C_{k_2}$, it holds that ψ accepts $N_{|I}$ with probability 1.*
   (b) ***Robust soundness:*** *for every vector $N \in \mathbb{F}^{l_1 \cdot l_2}$, it holds that*
   $$\mathbb{E}\left[\text{dist}\left(N_{|I}, \mathbf{SAT}(\psi)\right)\right] \geq \rho_T \cdot \text{dist}\left(N, C_{k_1} \otimes C_{k_2}\right).$$
   (c) ***Decision complexity:*** *The linear circuit ψ is always of size at most $O(\max\{k_1, k_2\})$.*

Remark 4.13. Note that circuit φ_k that recognizes C_k is of linear size, which is better than the corresponding circuit of Fact 2.15, which has only almost-linear size. In fact, for our purposes, we could have also assumed that φ_k has only almost-linear size as well. We chose to assume that φ_T^k has linear size because this choice improves the parameters of Theorem 4.7. Since this theorem may be useful for other works, we preferred to use the best parameters possible. We note that such a circuit φ_k of linear size can be obtained by choosing the sequence $\{C_k\}_{k=1}^{\infty}$ to be the expander codes of [SS96].

4.4 Proof of Theorem 4.7

In this section we provide the full proof of Theorem 4.7. As explained in Remark 4.8, our first step is to modify V such that it queries rows and columns according to the uniform distribution. More formally, we have the following lemma, which is proved by a variant of the expander-replacement technique of [PY91]. Since this is not the focus of this paper, we defer the proof of the lemma to Appendix A.

Lemma 4.14. *Suppose that there exists a linear PCPP verifier V that has randomness complexity $r(n)$, decision complexity $d(n)$, rejection ratio $\rho(n)$, and minimal field size $F(n)$, and that has b-row/column access. Then, there exists a linear PCPP verifier V_U with randomness complexity $r_U(n) = r(n) + O(1)$, decision complexity $d_U(n) = O(b \cdot d(n) + 2^{r(n)})$, rejection ratio $\rho_U(n) = \Omega(\rho(n)/b^2)$, minimal field size $F(n)$, and $O(b)$-row/column access, and which satisfies the following properties:*

- *V_U queries every row of the assignment matrix with equal probability, and always queries at least one such row.*
- *V_U does not query the columns of the assignment matrix.*
- *V_U queries every row of the proof matrix with equal probability, and same goes for the columns.*

We turn to the prove Theorem 4.7. Fix a linear PCPP verifier V that has randomness complexity $r(n)$, decision complexity $d(n)$, rejection ratio $\rho(n)$, and minimal field size $F(n)$, and that has b-row/column access. Let V_U denote the linear PCPP verifier that is obtained by applying Lemma 4.14 to V.

In the rest of this section, we use V_U to construct a linear PCPP verifier V' that has *robustness* $\Omega(\rho(n)/b^3)$ and also has randomness complexity $r(n) + O(1)$, decision complexity $O(b \cdot d(n) + 2^{r(n)})$, and minimal field size $F(n)$. Let $\mathbb{F}$ be a finite field of size at least $F(n)$, and fix a linear circuit $\varphi : \mathbb{F}^m \to \mathbb{F}^t$ of size n over $\mathbb{F}$. Let M_x and M_π be the assignment matrix and proof matrix of V_U with respect to φ.

We denote by $k_{x,\text{row}}$, $k_{x,\text{col}}$, $k_{\pi,\text{row}}$, $k_{\pi,\text{col}}$ the dimensions of M_x and M_π, i.e., M_x is a $k_{x,\text{col}} \times k_{x,\text{row}}$ matrix and M_π is a $k_{\pi,\text{col}} \times k_{\pi,\text{row}}$ matrix. Let $C_x \overset{\text{def}}{=} C_{k_{x,\text{row}}}$, $C_{\pi,\text{row}} \overset{\text{def}}{=} C_{k_{\pi,\text{row}}}$, and $C_{\pi,\text{col}} \overset{\text{def}}{=} C_{k_{\pi,\text{col}}}$ be the codes from Fact 4.12 with message lengths $k_{x,\text{row}}$, $k_{\pi,\text{row}}$, and $k_{\pi,\text{col}}$ respectively. Note that we did not define a code for the columns of M_x: since V_U only queries the rows M_x, we do not need to encode its columns.

The proof strings of V'. We begin the description of V' by describe its proof strings. Fix an assignment $x \in \mathbb{F}^m$ that satisfies φ. We define the proof string π' that convinces V' that φ accepts x as follows: Let N_x be the matrix obtained by encoding every row of M_x by C_x, and let N_π be the matrix that is obtained by encoding the proof matrix M_π of V_U via $C_{\pi,\text{row}} \otimes C_{\pi,\text{col}}$. Note that since

C_x, $C_{\pi,\text{row}}$, and $C_{\pi,\text{col}}$ are systematic, it holds that M_x is the $k_{x,\text{col}} \times k_{x,\text{row}}$ leftmost submatrix of N_x, and that similarly, M_π is the $k_{\pi,\text{col}} \times k_{\pi,\text{row}}$ top-left submatrix of N_π.

We now choose the proof string π' such that the concatenated vector $x \circ \pi'$ consists of the matrices N_x and N_π. More specifically, the proof string π' consists of the matrix N_π, and of the matrix N_x except for its leftmost $k_{x,\text{col}} \times k_{x,\text{row}}$ submatrix, where the latter submatrix is excluded since it is identical to x.

The action of V'. We proceed to describe the action of V' on input φ. For convenience, we think of V' as testing a fixed assignment x given a fixed proof string π'. Suppose that the vector $x \circ \pi'$ consists of two matrices N_x and N_π, and let M_x and M_π denote corresponding submatrices of N_x and N_π respectively. The verifier V' performs the following checks, while recycling randomness:

1. V' invokes the robust tester V_T (from Fact 4.12) to test that N_π is a legal codewords of $C_{\pi,\text{row}} \otimes C_{\pi,\text{col}}$ respectively.
2. V' emulates the action of V_U on M_x and M_π. Recall that V_U queries at most b rows and columns of M_x and M_π. The verifier V' queries the corresponding rows and columns of N_x and N_π, and extracts from them the corresponding rows and columns of M_x and M_π. The verifier V' then checks that V_U would have accepted those rows and columns of M_x and M_π, and that those rows and columns of N_x and N_π are legal codewords of the corresponding codes.

For convenience, let us denote by ψ' and I' the linear circuit and queries tuple that V' outputs. Let us denote by ψ_U and I_U the linear circuit and queries tuple that V_U outputs when it is invoked by V'. Let us denote by ψ_T and I_T the predicates and queries tuples that V_T outputs when it is invoked on N_π.

We need to make one more modification in the foregoing construction: The queries tuple I' of V' consists of I_T (which is used in the first check above), and of a collection of rows and columns of N_x and N_π (which are used in the second check above). We would like all of those parts to have roughly the same "weight" in I', that is, we would like each of them to constitute more or less the same fraction of I'. Moreover, we would like each of the rows and columns of N_x and N_π to constitute the same fraction of I',

To this end, we modify the queries tuple I' such that it contains multiple copies of I_T such that the total number of coordinates in those copies constitutes $1/8$ fraction of the coordinates in I'. We also add multiple copies of each row and column of N_x and N_π, such that each of those sets of copies constitute at least $1/8b$ fraction of the coordinates in I'. We also modify the output predicate ψ' of V' such that it checks that queries that query the same coordinate are given the same answer, and in particular, that all the copies of the same string are equal to each other.

The randomness complexity and decision complexity of V'. We turn to analyze the randomness complexity and decision complexity of V'. Let

$$k = \max\{k_{\pi,\text{row}}, k_{\pi,\text{col}}\}$$

It is not hard to see that the randomness complexity $r'(n)$ and decision complexity $d'(n)$ of V' satisfy

$$\begin{aligned} r'(n) &\leq \max\{r_U(n), \log k\} + O(1) \\ d'(n) &\leq d_U(n) + O(k), \end{aligned}$$

where $r_U(n) = r(n) + O(1)$ and $d_U(n) = O(b \cdot d(n) + 2^{r(n)})$ are the randomness complexity and decision complexity of V_U respectively, and the dependence on k comes from the invocation of V_T. In order to obtain the desired upper bounds on r' and d', we show that $r_U(n) \geq \log(k)$ and that $d_U(n) \geq O(k)$. This will imply that $r'(n) = r(n) + O(1)$ and that $d'(n) = O(b \cdot d(n) + 2^{r(n)})$, as required.

Recall that we assume that the verifier V queries both rows and columns of M_π, and that it queries *whole* rows and columns. This immediately implies that $d(n) \geq k$ and hence $d_U(n) = O(d(n)) \geq O(k)$. As for the randomness complexity, recall that the definition of row/column access requires that V queries every row and column of the proof matrix with non-zero probability, and this implies in particular that its randomness complexity must be at least $\log k$.

4.4.1 The robustness of V'

It remains to show that V' has robustness $\Omega(\rho(n)/b^3)$. Fix an assignment x to φ that is ε-far from $\mathbf{SAT}(\varphi)$, and let π' be a proof string. We show that

$$\mathbb{E}\left[\text{dist}\left((x \circ \pi)_{|I'}, \mathbf{SAT}(\psi')\right)\right] \geq \Omega(\rho(n)/b^3) \cdot \varepsilon. \tag{1}$$

Let N_x and N_π be the matrices contained in $x \circ \pi'$, and let M_x and M_π be their corresponding leftmost $k_{x,\text{col}} \times k_{x,\text{row}}$ and top-leftmost $k_{\pi,\text{col}} \times k_{\pi,\text{row}}$ submatrices respectively. Let N_x^{dec} be the matrix obtained by decoding each row of N_x to the closest codeword of C_x. Let N_π^{dec} be the legal codeword $C_{\pi,\text{row}} \otimes C_{\pi,\text{col}}$ that is closest to N_π. Let M_x^{dec} and M_π^{dec} be the corresponding $k_{x,\text{col}} \times k_{x,\text{row}}$ and $k_{\pi,\text{col}} \times k_{\pi,\text{row}}$ submatrices of N_x^{dec} and N_π^{dec} respectively.

Let τ be a threshold to be defined next. We divide our analysis to two cases: the case in which N_x or N_π are τ-far from N_x^{dec} or N_π^{dec} respectively, and the case in which both N_x and N_π are τ-close to N_x^{dec} or N_π^{dec} respectively. Let $\rho_U = \Omega(\rho(n)/b^2)$ denote the rejection ratio of V_U. Then, we define τ by

$$\tau \overset{\text{def}}{=} R_C \cdot \delta_C \cdot \rho_U \cdot \varepsilon/4 \cdot b = \Omega(\rho(n)/b^3) \cdot \varepsilon.$$

We turn to analyze each of the latter cases separately.

N_x or N_π are τ-far from N_x^{dec} or N_π^{dec}. We deal separately with N_x and N_π. If N_π is τ-far from N_π^{dec}, we obtain Inequality 1 above immediately from the robustness of the tester V_T. To see it, observe that the robustness of V_T (Fact 4.12) implies that

$$\mathbb{E}\left[\text{dist}\left(N_{\pi|I_T}, \mathbf{SAT}(\psi_T)\right)\right] \geq \rho_T \cdot \tau$$

where I_T and ψ_T denote the outputs of V_T when invoked on N_π. By our construction of V', the copies of I_T form $1/8$ fraction of the coordinates of I', and therefore it follows that

$$\mathbb{E}\left[\text{dist}\left((x \circ \pi)_{|I'}, \mathbf{SAT}(\psi')\right)\right] \geq \frac{1}{8} \cdot \rho_T \cdot \tau \geq \Omega(\rho(n)/b^3) \cdot \varepsilon,$$

as required.

Suppose now that N_x is τ-far from N_x^{dec}. Recall that $k_{x,\text{col}}$ is the number of rows of N_x. For every $j \in [k_{x.\text{col}}]$, let δ_j denote the relative distance of the j-th row of N_x from C_x. Since N_x is τ-far from N_x^{dec}, it holds that

$$\frac{1}{k_{x,\text{col}}} \cdot \sum_{j=1}^{k_{x,\text{col}}} \delta_j \geq \tau.$$

Due to the properties of V_U, we know that the verifier V' queries every row of N_x with the same probability - let us denote this probability by p. Moreover, we know that V' always queries at least one row of N_x, and therefore $p \geq 1/k_{x,\text{col}}$. For every $j \in [k_{x,\text{col}}]$, let us denote by $\mathbf{1}_j$ an indicator variable indicating whether the j-th row is queried by V'. Since every row of N_x that is queried by V' forms $1/8b$ fraction of I', it follows that

$$\begin{aligned}
\mathbb{E}\left[\operatorname{dist}\left((x\circ\pi)_{|I'}, \mathbf{SAT}(\psi')\right)\right] &\geq \frac{1}{8b}\cdot\mathbb{E}\left[\sum_{j=1}^{k_{x,\text{col}}}\delta_j\cdot\mathbf{1}_j\right]\\
&= \frac{1}{8b}\cdot\sum_{j=1}^{k_{x,\text{col}}}\delta_j\cdot p\\
&\geq \frac{1}{8b}\cdot\frac{1}{k_{x,\text{col}}}\cdot\sum_{j=1}^{k_{x,\text{col}}}\delta_j\\
&\geq \frac{1}{8b}\cdot\tau\\
&= \Omega(\rho(n)/b^3)\cdot\varepsilon,
\end{aligned}$$

as required.

N_x and N_π are τ-close to N_x^{dec} and N_π^{dec}. We begin by noting that in this case, it must hold that M_x is τ/R_C-close to M_x^{dec}. To see it, note that M_x forms R_C fraction of the coordinates of N_x, since R_C the rate of C_x, and recall that we assume that N_x is τ-close to N_x^{dec}. We also note that $\tau/R_C < \varepsilon/2$.

Recall that we say that a row of N_π decodes well if the codeword of $C_{\pi,\text{row}}$ to which it is closest is the corresponding row of N_π^{dec}, and that the same goes for the columns. In addition, we will say that a row (respectively, a column) of M_π decodes well if the corresponding (respectively, a column) of N_π decodes well.

Let us denote by x^{dec} the matrix M_x^{dec} when viewed as an assignment to φ, and let π^{dec} be the matrix M_π^{dec} viewed as a proof string of V. In this notation, we get that $\operatorname{dist}\left(x, x^{\text{dec}}\right) < \varepsilon/2$. On the other hand, $\operatorname{dist}(x, \mathbf{SAT}(\varphi)) = \varepsilon$, and therefore by the triangle inequality it holds that $\operatorname{dist}\left(x^{\text{dec}}, \mathbf{SAT}(\varphi)\right) \geq \varepsilon/2$. By the rejection ratio of V_U, it holds that ψ_U rejects $\left(x^{\text{dec}}\circ\pi^{\text{dec}}\right)_{|I_U}$ with probability at least $\rho_U\cdot\varepsilon/2$. We now show that with probability at least $\rho_U\cdot\varepsilon/4$, it holds that

1. ψ_U rejects $\left(x^{\text{dec}}\circ\pi^{\text{dec}}\right)_{|I_U}$, and
2. all the rows and columns of M_π whose coordinates are contained in I_U decode well.

We will later show that whenever the above two conditions hold, it also holds that $(x\circ\pi')_{|I'}$ is $\Omega(1/b)$-far from $\mathbf{SAT}(\psi')$, and this will imply the required robustness.

By Claim 4.11, it holds that a uniformly distributed row of N_π decodes well with probability at least

$$1 - \tau/\delta_C$$

and that the same holds for a uniformly distributed column of N_π. Since the rows and columns of M_π constitute R_C fraction of the rows and columns of N_π, it holds a uniformly distributed row of M_π decodes well with probability at least

$$1 - \frac{\tau}{R_C\cdot\delta_C}$$

and that the same holds for a uniformly distributed column of M_π. Now, recall that V_U queries each row of M_π with equal probability, and that the same goes for the columns of M_π. Furthermore, V_U queries at most b rows and columns of M_π. By the union bound, it follows that all the rows and columns of M_π that are queried by V_U decode well with probability at least

$$1 - \frac{b \cdot \tau}{R_C \cdot \delta_C} = 1 - \frac{\rho_U \cdot \varepsilon}{4}.$$

By combining the latter bound with the fact that ψ_U rejects $\left(x^{\mathrm{dec}} \circ \pi^{\mathrm{dec}}\right)_{|I_U}$ with probability at least $\rho_U \cdot \varepsilon/2$, and using the union bound, it follows that with probability at least $\rho_U \cdot \varepsilon/4$ both Conditions 1 and 2 above hold simultaneously. We conclude by showing that whenever those two conditions hold, it holds that $(x \circ \pi')_{|I'}$ is $\Omega(1/b)$-far from $W_{\psi'}$, and this will imply the required robustness.

Fix a sequence ω of random bits for V' for which both Conditions 1 and 2 hold. Recall that we denote by ψ_U, ψ', I_U, and I' be the predicates and queries tuples that V_U and V' output. Let w' be a satisfying assignment of ψ' closest to $(x \circ \pi')_{|I'}$, and let w_U be the satisfying assignment for ψ_U obtained by taking the corresponding rows and columns of M_x and M_π from w'. By assumption, ψ_U rejects $\left(x^{\mathrm{dec}} \circ \pi^{\mathrm{dec}}\right)_{|I_U}$, and therefore there exists a coordinate on which $\left(x^{\mathrm{dec}} \circ \pi^{\mathrm{dec}}\right)_{|I_U}$ and w_U disagree - let us denote this coordinate by i.

Let us assume that i belongs to M_π^{dec} - the case where i belongs to M_x^{dec} is similar but simpler, because one needs not worry about whether the rows of M_x decode well. Without loss of generality, assume that the this coordinate belongs to a row of M_π^{dec} that is queried by I_U - if the coordinate belongs to a column of M_π^{dec}, then the argument is analogous. Let j be the index of the row of M_π^{dec} to which$i$ belongs.

Now, by assumption, the j-th row of N_π decodes well, and therefore its nearest codeword of $C_{\pi,\mathrm{row}}$ is the j-th row of N_x^{dec}. On the other hand, the assignment w' contains an assignment to the j-th row[8] that is another codeword of $C_{\pi,\mathrm{row}}$. It therefore follows that the j-th row of N_π is at least $\delta_C/2$-far from the corresponding part of w'. Finally, the copies of the j-th of N_π constitute $1/8 \cdot b$ fraction of I', so it follows that

$$\mathrm{dist}\left(\left(x \circ \pi'\right)_{|I'}, \mathbf{SAT}(\psi')\right) \geq \frac{1}{8 \cdot b} \cdot \frac{\delta_C}{2} = \Omega(1/b),$$

as required.

5 Construction of Linear PCPPs with $\sqrt{n}$ Queries

In this section, we explain how to construct a linear PCPP that have proof length $\tilde{O}(n)$, query complexity $\tilde{O}(\sqrt{n})$, and rejection ratio $\Omega(1)$, and that has $O(1)$-row/column access. Later, in Section 6, we will compose this linear PCPP with itself for $O(\log\log n)$ times to obtain a linear PCP with constant query complexity and proof length $n \cdot (\log n)^{O(\log\log n)}$. More formally, our main result in this section is the following:

Theorem 5.1. *There exists a linear PCPP with randomness complexity $\frac{1}{2}\log n + O(\log\log n)$, decision complexity $\tilde{O}(\sqrt{n})$, minimial field size $\tilde{O}(\sqrt{n})$, rejection ratio $\Omega(1)$, and $O(1)$-row/column access.*

[8] In fact, it contains multiple copies of such an assignment, but since w' satisfies ψ', all those copies must be equal to each other.

This section is organized as follows: In Section 5.1 below, we define an auxiliary object called linear simultaneous PCPP (abbreviated SPCPP), and state a construction of such object (Theorem 5.5). Then, in Section 5.2, we prove Theorem 5.1 based on Theorem 5.5. Finally, in Section 5.3, we prove Theorem 5.5 by constructing the required SPCPPs.

5.1 Simultaneous PCPPs

Intuitively, an SPCPP is a variant of a linear PCPP verifier, which verifies many linear assertions rather than just one. The verification is *simultaneous*, in the sense that if *at least one* of the linear assertions is far from being correct, then the verifier rejects with noticeable probability. A bit more formally, an SPCPP verifier takes an input linear circuits $\varphi_1, \ldots, \varphi_t$, gets oracle access to vectors $x_1, \ldots, x_t$ and to a proof string π, and is required to reject if x_i is far from $\mathbf{SAT}(\varphi_i)$ for *at least* one i. In our application, we will use the SPCPP to verify $t = \tilde{O}(\sqrt{n})$ linear assertions of size $s = \tilde{O}(\sqrt{n})$ each, by using $\tilde{O}(\sqrt{n})$ queries.

Definition 5.2. Let $r, d, \ell, F : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, $\rho : \mathbb{N} \times \mathbb{N} \to (0,1)$. A linear simultaneous PCPP (SPCPP) verifier V with randomness complexity r, decision complexity d, proof length ℓ, rejection ratio ρ, and minimal field size F is a probabilistic polynomial time machine that satisfies the following requirements:

1. **Input:** The verifier V takes as input a finite field $\mathbb{F}$ and t linear circuits $\varphi_1, \ldots, \varphi_t : \mathbb{F}^m \to \mathbb{F}^p$ of size s. The field $\mathbb{F}$ is required to be of size at least $F(t, s)$.

2. **Output:** The verifier V outputs a tuple I of coordinates in $[t \cdot m + \ell(t, s)]$, and a linear circuit $\psi : \mathbb{F}^{|I|} \to \mathbb{F}^{p'}$ of size at most $d(t, s)$.

3. **Randomness complexity:** On every finite field $\mathbb{F}$ and input circuits $\varphi_1, \ldots, \varphi_t$, and on every sequence of coin tosses, V tosses at most $r(t, s)$ coins.

4. **Completeness:** For every $x_1 \in \mathbf{SAT}(\varphi_1), \ldots, x_t \in \mathbf{SAT}(\varphi_t)$, there exists a string $\pi \in \mathbb{F}^{\ell(t,s)}$ such that
$$\Pr\left[\psi\left((x_1 \circ \ldots \circ x_t \circ \pi)_{|I}\right) \text{ accepts}\right] = 1,$$
where ψ and I are generated by the verifier V on input $(\mathbb{F}, \varphi_1, \ldots, \varphi_t)$. We refer to π as the proof of $x_1, \ldots, x_t$, or as the proof that convinces V that $\varphi_1, \ldots, \varphi_t$ accept $x_1, \ldots, x_t$.

5. **Soundness:** For every $x_1, \ldots, x_t \in \mathbb{F}^m$ such that x_i is ε-far from $\mathbf{SAT}(\varphi_i)$ for some $i \in [t]$ and some $\varepsilon > 0$, and for every string $\pi \in \mathbb{F}^{\ell(t,s)}$, it holds that
$$\Pr\left[\psi\left((x_1 \circ \ldots \circ x_t \circ \pi)_{|I}\right) \text{ rejects}\right] \geq \rho(t, s) \cdot \varepsilon,$$
where ψ and I are generated by the verifier V on input $(\mathbb{F}, \varphi_1, \ldots, \varphi_t)$. We refer to $x_1, \ldots, x_t$ as the tested assignments and to π as the proof string.

Remark 5.3. The above definition of linear SPCPP can easily be generalized to a definition of a general (non-linear) simultaneous PCPP. We are not aware of such a notion in the literature. However, in retrospect, the tensor product lemma of [Mei09] (following [DR06]) may be viewed as a construction of a general SPCPP.

SPCPPs with row/column access. Recall that our goal is to construct linear PCPPs that have the *row/column access* property (Definition 4.4), which is important since we use it later to make the linear PCPP robust. To this end, we will also want our SPCPPs to have row/column access as well. Recall that a linear PCPP verifier is said to have b-row/column access if the coordinates of the tested assignment and the proof string can be arranged in matrices M_x and M_π, such that the verifier always queries at most b rows and columns of M_x and M_π. The same definition can be applied to linear SPCPPs. Actually, we will want our SPCPPs to satisfy the following stronger property, which will be used shortly below in Theorem 5.2.

Definition 5.4. An SPCPP verifier V is said to have strong b-row/column access if it has b-row/column access, and furthermore, the rows of the assignment matrix are the tested assignments $x_1, \ldots, x_t$ (i.e., the i-th row is the vector x_i).

Our construction of SPCPPs. The main technical part of this section is the construction of the following SPCPPs.

Theorem 5.5. *There exists an SPCPP verifier with randomness complexity* $\log(t+s)+O(\log\log(s))$, *decision complexity* $\tilde{O}(t+s)$, *minimial field size* $O(s)$, *rejection ratio* $\Omega(1)$, *and strong* $O(1)$-*row/column access.*

5.2 Proof of Theorem 5.1 from Theorem 5.5

5.2.1 Proof idea

We wish to construct a linear PCPP of randomness complexity $\approx \frac{1}{2}\log n$ and decision complexity $\approx \sqrt{n}$, using the SPCPPs of Theorem 5.5. Fix a linear circuit φ of size n, tested assignment x, and a proof string π. We wish to verify that x is close to $\mathbf{SAT}(\varphi)$.

The most straightforward way to verify that x is close to $\mathbf{SAT}(\varphi)$ is to invoke the SPCPP with with φ being the only input circuit and x being the only tested assignment. In other words, we invoke the the SPCPP with $t = 1$, $s = n$. Unfortunately, using this choice of t and s with Theorem 5.5 would result in randomness complexity $\approx \log n$ and decision complexity $\approx n$, which are too large.

The "right" way to set the parameters of the SPCPP is $t \approx \sqrt{n}$, $s \approx \sqrt{n}$: this would give us randomness complexity $\approx \frac{1}{2}\log n$ and decision complexity $\approx \sqrt{n}$, as required. In order to be able to invoke the SPCPP with $t \approx \sqrt{n}$, $s \approx \sqrt{n}$, we need to transform φ into an "equivalent" collection of circuits $\psi_1, \ldots, \psi_{\approx\sqrt{n}}$, each being of size $\approx \sqrt{n}$. This can be done by an object called robust circuit decomposition, constructed in [Mei09] (abbreviated decomposition). Using our terminology, the decomposition of [Mei09] is a linear PCPP that does not satisfy the standard soundness property, but is rather only guaranteed to reject non-satisfying assignments with *non-zero* probability. However, when it does reject a non-satisfying assignment, it does so *robustly.* More formally, the decomposition satisfies the following property:

- Suppose the decomposition is given input circuit φ, tested assignment x and proof string π. If x is ε-far from $\mathbf{SAT}(\varphi)$, then with *non-zero* probability, the decomposition outputs an output circuit ψ and queries tuple I such that $(x \circ \pi)_{|I}$ is $\Omega(\varepsilon)$-far from $\mathbf{SAT}(\psi)$.

Moreover, the decomposition of [Mei09] has randomness complexity $\frac{1}{2}\log n + O(\log\log n)$ and decision complexity $\tilde{O}(\sqrt{n})$.

Given the decomposition of [Mei09], we construct our linear PCPP as follows: suppose our linear PCPP verifier is given an input circuit φ, tested assignment x, and proof string π. Our verifier will start by invoking the decomposition of [Mei09] on φ, x, and π, and *on all $t = \tilde{O}(\sqrt{n})$ possible sequences of random coins*. The result is a sequence of output circuits $\psi_1, \ldots, \psi_t$ of size $\tilde{O}(\sqrt{n})$ and queries tuples $I_1, \ldots, I_t$. Next, for every $j \in [t]$, we define $x_j \overset{\text{def}}{=} (x \circ \pi)_{|I_i}$. Our verifier will finish by invoking the SPCPP of Theorem 5.5 on input circuits $\psi_1, \ldots, \psi_t$ and tested assignments $x_1, \ldots, x_t$.

The soundness of this linear PCPP is argued as follows: If x is ε-far from $\mathbf{SAT}(\varphi)$, then by the property of the decomposition, there exists some $i \in [t]$ such that x_i is $\Omega(\varepsilon)$-far from $\mathbf{SAT}(\psi_i)$. This implies, by the soundness of the SPCPP, that the SPCPP verifier rejects with probability $\Omega(\varepsilon)$, as required.

The non-trivial issue is to show that the linear PCPP has $O(1)$-row/column access, which is discussed next. To this end, we use the following additional property of the decomposition, called "matrix access": The tested assignment x and the proof string π can be arranged in matrices M_x and M_π such that every x_i consists of $O(1)$ whole rows of those matrices, in a way that satisfies few additional technical properties. By combining the latter property with the strong row/column access of the SPCPP, it is possible to show that our linear PCPP has $O(1)$-row/column access.

5.2.2 The formal proof

We begin by giving a formal statement of the robust circuit decomposition of [Mei09] in the following lemma, which can be proved using tools from [Mei09] (see discussion at the end of this section for details).

Lemma 5.6 (Follows from [Mei09])**.** *There exists a linear PCP D with randomness complexity $\frac{1}{2}\log n + O(\log\log n)$, and decision complexity $\tilde{O}(\sqrt{n})$. Instead of satisfying the standard soundness requirement, D satisfies the following soundness requirement:*

- *For every $x \in \mathbb{F}^m$ and every string $\pi \in \mathbb{F}^{\ell(n)}$, there exists some sequence of random bits on which D outputs a linear circuit ψ and queries tuple I such that*

$$\operatorname{dist}\left((x \circ \pi)_{|I}, \mathbf{SAT}(\psi)\right) \geq \rho_D \cdot \operatorname{dist}\left(x, \mathbf{SAT}(\varphi)\right),$$

  where ρ_D is a universal constant.

Furthermore, D has the following "matrix access" property: This means that D has b'-block access for some arbitrarily large b', and moreover, it satisfies the following properties.

- *There is some universal constant b_D such that D always query at most b_D* distinct *blocks (though it may query many copies of the same block).*
- *All the assignments blocks are of the same length and all the proof blocks are of the same length.*
- *D always queries the same number of assignment blocks and the same number of proof blocks. In other words, the queries tuple I always contains at most the same number of assignment blocks and the same number of proof blocks.*
- *In the tuple I, the blocks are written consecutively. In other words, the tuple I consists of the coordinates of first block, then the coordinates of the second block, etc. Moreover, the order of the coordinates inside the block is the same for all its occurrences in every tuple I, for every sequence of random bits of D.*
- *In each tuple I, the assignment blocks always precede the proof blocks.*

Remark 5.7. The definition of "matrix access" may seem arbitrary. The motivation for this definition will become apparent when we prove below that the linear PCPP we construct has $O(1)$-row/column access.

We turn to proving Theorem 5.1, restated next.

Theorem (5.1, restated)**.** *There exists a linear PCPP with randomness complexity $\frac{1}{2}\log n+O(\log\log n)$, decision complexity $\tilde{O}(\sqrt{n})$, rejection ratio $\Omega(1)$, and $O(1)$-row/column access.*

Let D be the decomposition of Lemma 5.6, and let V the the SPCPP of Theorem 5.5. We construct a linear PCPP verifier V' with the desired parameters. Fix an input circuit φ of size n, a tested assignment x for φ, and a proof string π' for V'. The verifier V' expects π' to consist of a proof string π_D for D and a proof string π_V for V.

The action of V'. When given φ as input, the verifier V' acts as follows:

1. First, V' invokes D on φ and on all possible sequences of random bits. Let $\psi_1,\ldots,\psi_t$ and $I_1,\ldots,I_t$ be the linear circuits and queries tuples that D outputs on all the possible sequences of random bits. Note that since D has randomness complexity $\frac{1}{2}\log n+O(\log\log n)$, it holds that $t=\tilde{O}(\sqrt{n})$, and enumerating over all t sequences of random bits can be done in polynomial time. Furthermore, the linear circuits $\psi_1,\ldots,\psi_t$ are of size $\tilde{O}(\sqrt{n})$ due to the decision complexity of D.
2. Next, V' emulates V on input $\psi_1,\ldots,\psi_t$, thus obtaining a linear circuit η and a queries tuple J.
3. Note that the queries tuple J contains coordinates of the string
$$u \stackrel{\text{def}}{=} (x\circ\pi_D)_{|I_1}\circ\ldots\circ(x\circ\pi_D)_{|I_t}\circ\pi_V. \tag{2}$$
On the other hand, the verifier V' is required to output a queries tuple J' that contains coordinates of the following string:
$$v \stackrel{\text{def}}{=} x\circ\pi_D\circ\pi_V$$
The verifier V' now constructs J' by translating the coordinates of u in J into the corresponding coordinates of v in the obvious way.
4. Ther verifier V' modifies J' such that every coordinate in J' appears at most once. That is, if a coordinate appeared in J' more than once, then all its copies except one are removed. The verifier V' also modifies the linear circuit η such that input gates that correspond to different copies of the same coordinate are replaced by a single input gate, thus obtaining a linear circuit η'.
5. The verifier V' concludes by outputting the linear circuit η' and queries tuple J'.

The parameters of V'. Let r, d, F and ρ be the randomness complexity, decision complexity, minimial field size, and rejection ratio of V respectively. It is not hard to see that V' has randomness complexity $r(\tilde{O}(\sqrt{n}),\tilde{O}(\sqrt{n}))=\frac{1}{2}\log n+O(\log\log n)$, decision complexity $d(\tilde{O}(\sqrt{n}),\tilde{O}(\sqrt{n}))=\tilde{O}(\sqrt{n})$, and minimal field size $F(\tilde{O}(\sqrt{n}),\tilde{O}(\sqrt{n}))=\tilde{O}(\sqrt{n})$. It remains to analyze the rejection ratio of V'.

Suppose that x is ε-far from $\mathbf{SAT}(\varphi)$. Then, by the soundness of the decomposition D, there exists some index $i \in [t]$ such that

$$\operatorname{dist}\left((x \circ \pi)_{|I_i}, \mathbf{SAT}(\psi_i)\right) \geq \rho_D \cdot \varepsilon.$$

Recall the definition of the string u in Equation 2 above. By the soundnes of the SPCPP V, it follows that η' rejects $u_{|J'}$ with probability at least

$$\rho(\tilde{O}(n), \tilde{O}(n)) \cdot \rho_D \cdot \varepsilon = \Omega(\varepsilon).$$

This implies that V' has rejection ratio $\Omega(1)$, as required.

The row/column access of V'. We conclude by showing that V' has $O(1)$-row/column access. Recall that D has matrix access, which means that the tested assignment x and proof string π_D can be arranged in matrices M_x and M_D such that every queries tuple I_i queries at most b_D distinct rows of those matrices. Furthermore, recall that V has strong b_V-row/column access for some universal constant b_V. This means that the string π_V can be arranged in a matrix M_V, such that the V queries at most b_V rows and columns of the matrix M_V, and of the matrix M_I whose rows are the strings $(x \circ \pi_D)_{|I_1}, \ldots, (x \circ \pi_D)_{|I_t}$.

We begin by defining the assignment and proof matrices of V'. We define the assignment matrix of V' to be M_x, the assignment matrix of D. We define the proof matrix of V' to be the matrix constructed from M_D and M_V by writing one of them above the other, and adding dummy coordinates to make sure that all the rows are of the same length. Let us denote the resulting proof matrix by $M_{\pi'}$.

We now show that V' queries at most $O(b_D \cdot b_V) = O(1)$ rows and columns of M_x and $M_{\pi'}$. We start with the rows. By the row/column access of V, the queries tuple J' contains at most b_V rows of the matrices M_I and M_V. The rows of M_V are rows of $M_{\pi'}$, so they pose no problem. As for rows of M_I, recall that each row of M_I is a string $(x \circ \pi_D)_{|I_i}$. By the matrix access of D, the queries tuple I_i contains at most b_D *distinct* rows of M_x and M_D. Since rows of M_D are rows of $M_{\pi'}$, it follows that the queries tuple J contains at most $b_D \cdot b_V$ *distinct* rows of M_x and $M_{\pi'}$. Finally, since V' constructs J' such that it contains at most one copy of each coordinate, we get that J' contains at most $b_D \cdot b_V$ rows, as required.

We turn to upper bound the number of columns queried by V'. By the row/column access of V, the queries tuple J' contains at most b_V columns of M_I and M_V. Columns of M_V are contained in columns of $M_{\pi'}$, so they pose no problem. The tricky part is to handle the columns of M_I. The crucial observation here is that due to the matrix access of D, every column of M_I is contained in a column of M_x or M_D. Indeed, this observation was the reason that matrix access was defined this way in [Mei09].

To see why the latter observation is correct, fix a queries tuple I_i of D, let $h \in [|I_i|]$, and let k be the h-th coordinate in I_i. Now, observe that given the index h alone, one can decide whether the coordinate k belongs to an assignment block or a proof block of D, and compute the offset of k within the block. This can be done without knowing k itself due to the matrix access of D. This can be seen as follows:

- Recall that the matrix access implies that all the assignment blocks of D are of the same length, say l_a.
- Moreover, every tuple I_i reads the same number of assignment blocks, say p_a, and the assignment blocks precede the proof blocks in I_i.

- Thus, the coordinate k belongs to an assignment block if and only if $h \leq l_a \cdot p_a$.
- Furthermore, if k does belong to an assignment block, then its offset in the block exactly $((h-1) \bmod l_a) + 1$.
- If k belongs to a proof block, its offset in the block can be determined similarly.

Finally, this means that for every column h of M_I, one of the following holds:

- All the coordinates of M_I belong to *assignment* blocks of D, and have the same offset in the corresponding blocks.
- All the coordinates of M_I belong to *proof* blocks of D, and have the same offset in the corresponding blocks.

In the first case the h-th column of M_I is contained in a column of M_x, and in the second case it is contained in a column of M_D. This concludes the proof of Theorem 5.1.

Deriving Lemma 5.6. A few comments on how to derive Lemma 5.6 from [Mei09] are in place:

1. While all the definitions and proofs in [Mei09] are written in terms of boolean (non-linear) circuits, they work equally well for our case of linear circuits.
2. While the Lemma 5.6 is not stated explicitly in [Mei09], it can be proved easily by combining Definitions 6.1 and 6.3, Lemma 6.4, and Proposition 8.5 there.
3. A small issue that needs attention is that Proposition 8.5 of [Mei09] does not state that the fact that D queries at most b_D *distinct* blocks. However, this property can be observed by examining the proof. In short, the reason that this property holds is that the decomposition that Proposition 8.5 takes as input has $O(1)$-block access, and the decomposition that Proposition 8.5 outputs queries only copies of the blocks that were queried by the input decomposition, and copies of their encodings.

5.3 Construction of SPCPPs

In the rest of this section, we construct the linear sPCPPs of Theorem 5.5, restated next.

Theorem (5.5, restated)**.** *There exists an SPCPP verifier with randomness complexity* $\log(t+s) + O(\log\log(s))$*, decision complexity* $\tilde{O}(t+s)$*, minimial field size* $O(s)$*, rejection ratio* $\Omega(1)$*, and strong* $O(1)$*-row/column access.*

We construct the SPCPP in four steps:

1. We first describe how to construct an SPCPP for the simple special case in which all the input circuits $\varphi_1, \ldots, \varphi_t$ are the same. This is done in Section 5.3.1.
2. Then, we show how to construct an SPCPP for a more interesting case, which we call colorable constraint systems, by reducing it to few instances of the foregoing simple case. This is done in Section 5.3.2.
3. Next, we show how to transform arbitrary linear circuits $\varphi_1, \ldots, \varphi_t$ into a colorable constraint system. This is done in Section 5.3.3.
4. Finally, we combine the results of the two last steps to construct an SPCPP for the genral case. This is done in Section 5.3.4

5.3.1 A simple case

We begin with constructing an SPCPP verifier V for the case in which all the input circuits $\varphi_1, \ldots, \varphi_t$ are the same. The following construction works for every field, and places no limitation on the field size.

Lemma 5.8. *There exists an SPCPP verifier that works only when all its input circuits are equal, and has randomness complexity* $\log(t+s)+O(1)$*, decision complexity* $\tilde{O}(t+s)$*, rejection ratio* $\Omega(1)$*, and strong* $O(1)$*-row/column access.*

Fix an input circuit $\varphi \overset{\text{def}}{=} \varphi_1 = \ldots = \varphi_t$ of size s, a finite field $\mathbb{F}$, tested assignments $x_1, \ldots, x_t \in \mathbb{F}^m$, and a proof string π. Let $W = \mathbf{SAT}(\varphi)$. We construct an SPCPP verifier V that verifies that $x_1, \ldots, x_t \in W$.

Let M_x be the $t \times m$ matrix whose rows are the tested assignments $x_1, \ldots, x_t$. Let $C : \mathbb{F}^t \to \mathbb{F}^l$ be a systematic linear code (as in Fact 2.15) of message length t, rate R_C and relative distance δ_C. Let N be the $l \times m$ matrix obtained by encoding the columns of M_x via C. Note that since C is systematic, the first t rows of N are just the matrix M_x. The verifier V expects the proof matrix M_π to consist of the last $l-t$ rows of N.

The motivation for defining the matrix N lies in the following observation: if $x_1, \ldots, x_t \in W$, then all the rows of N belong to W as well, including the rows that are not in M_x. This is true since all the rows of N are linear combinations of rows of M_x. As we will see in Claim 5.9 below, if one of the x_i's does not belong to W, then many rows of N do not belong to W.

This motivates the following construction of V: Let M_x and M_π be the assignment and proof matrices of V. Let N' be the $l \times m$ matrix whose first t rows are M_x and whose last $l-t$ rows are M_π. The matrix N' is expected to be equal to N, but might be different. The verifier V performs the following two checks while recycling randomness:

1. V chooses a row of N' uniformly at random and checks that it belongs to W.

2. V chooses a column of N' uniformly at random and checks that it is a legal codeword of C.

The verifier V accepts if and only if both checks accept. It is easy to see that the randomness complexity of V is indeed at most $\log(t+s)$, and that it has strong 2-row/column access. To see that V has decision complexity $\tilde{O}(t+s)$, observe that checking that a row of N' is in W can be done by the input circuit φ itself (which has size s), and that checking that a column of N' is a codeword of C can be done by a linear circuit of size $\tilde{O(t)}$ by Fact 2.15. It remains to analyze the rejection ratio of V. The crux of our argument is the following (standard) result.

Claim 5.9. *Let* $W_0 \subseteq \mathbb{F}^{m_0}$ *be a linear space, and let* N_0 *be a* $l \times m_0$ *matrix whose columns are legal codewords of* C*. If at least one row of* N_0 *does not belong to* W_0*, then at least* δ_C *fraction of the rows of* N_0 *do not belong to* W_0*.*

Before proving Claim 5.9, let us first see how to use it to analyze the rejection ratio of V. Suppose that for some $\varepsilon > 0$ there exists $i \in [t]$ such that x_i is ε-far from W. As a warm-up, first assume that all the columns of N' are legal codewords of C, i.e., that $N' = N$. Note that the i-th row of N', which is just x_i, does not belong to W by assumption. Thus, by applying Claim 5.9 with $N_0 = N'$ and $W_0 = W$, it follows that at least δ_C fraction of the rows of N' do not belong to W. This implies that V rejects with probability at least $\delta_C \geq \Omega(\varepsilon)$, as required.

Next, suppose that some of the columns of N' are not legal codewords of C. Let T denote the set of those columns. If $|T|/m \geq \varepsilon$, then V rejects with probability at least ε, and we are done.

Suppose that $|T|/m < \varepsilon$. We choose N_0 be the matrix obtained from N' by removing the columns in T, and let W_0 be the vector space obtained by projecting W to the coordinates in $[m] - T$.

Now, observe that the $x_{i|[m]-T}$ can not belong to W_0, or otherwise x_i would have been ε-close to W. Thus, N_0 is a matrix whose columns are all legal codewords of C, and whose i-th row does not belong to W_0. It follows by Claim 5.9 that at least δ_C fraction of the rows of N_0 do not belong to W_0. This implies that at least δ_C fraction of the rows of N' do not belong to W, and hence V rejects with probability at least $\delta_C \geq \Omega(\varepsilon)$, as required.

Proof of Claim 5.9. Without loss of generality, assume that the first row of N_0 does not belong to W_0, and denote this row by x. Let $w^{\perp}$ be any vector that is orthogonal to W_0 but is not orthogonal to x (such $w^{\perp}$ must exist, by the assumption that $x \notin W$). Now, let $v \overset{\text{def}}{=} N_0 \cdot w^{\perp}$, i.e., v is the result of the matrix multiplication of N_0 and $w^{\perp}$). Observe that

1. v is a codeword of C, since it is a linear combination of columns of N_0, which are codewords of C.

2. v is a non-zero vector. To see it, the note that first coordinate of v must be non-zero, since the first row of N_0 is not orthogonal to $w^{\perp}$.

We conclude that v is a non-zero codeword of C, and therefore at least δ_C fraction of its coordinates are non-zero. However, for each non-zero coordinate of v, the corresponding row of N_0 is not orthogonal to $w^{\perp}$ and thus does not belong to W_0. Hence, at least δ_C fraction of the rows of N_0 do not belong to W_0, as required. ■

5.3.2 The case of colorable constraint systems

We proceed to show a construction of an SPCPP for circuits $\varphi_1, \ldots, \varphi_t$ that are a colorable constraints system (CCS), to be defined below. We will later show that any sequence of circuits $\varphi_1, \ldots, \varphi_t$ can be transformed into a CCS.

Informally, we say that a collection of circuits $\varphi_1, \ldots, \varphi_t : \mathbb{F}^m \to \mathbb{F}^p$ forms a CCS if the subspaces $\mathbf{SAT}(\varphi_1), \ldots, \mathbf{SAT}(\varphi_t)$ can be described by a collection $\mathcal{S}$ of linear constraints that can be "legally colored" using few colors. Roughly, we say that a coloring of the constraints in $\mathcal{S}$ is legal if constraints of the same color do not share coordinates. The idea that underlies our construction of an SPCPP for a CCS is that in a CCS can be decomposed to few monochromatic systems of constraints, such that each system can be reduced to the simple case discussed above.

Colorable constraint systems. We begin by formally defining the notion of a linear constraint, and the set of linear constraints associated with a linear circuit. Given a vector space $W \subseteq \mathbb{F}^m$ and a vector $s \in \mathbb{F}^m$, we say that s is a (linear) constraint of W if s is orthogonal to all the vectors in W. We say that a constraint s touches a coordinate $i \in [m]$ if $s_i \neq 0$. We say that a vector space W is described by a set S of constraints of W if S spans all the constraints of W (i.e., S spans the dual space $W^{\perp}$).

Let φ be a linear circuit, and let $W = \mathbf{SAT}(\varphi)$. Recall that a vector $v \in \mathbb{F}^m$ belongs to W if and only if $\varphi(v)$ is the all-zeroes vector. Now, observe that every output gate of φ corresponds to a constraint of W, simply by taking the linear combination that the gate computes and presenting it as a vector. We define the set of constraints of φ to be the set of all the constraints that correspond to output gates of φ. We are now ready to define the notion of a CCS.

Definition 5.10 (Colorable constraints system)**.** Let $\chi \in \mathbb{N}$, let $\varphi_1, \ldots, \varphi_t : \mathbb{F}^m \to \mathbb{F}^p$ be linear circuits, and let $S_1, \ldots, S_t \subseteq \mathbb{F}^m$ be the corresponding sets of constraints. We say that $\varphi_1, \ldots, \varphi_t$

form a χ-colorable constraints system (abbreivated χ-CCS) if the union $\mathcal{S} \stackrel{\text{def}}{=} S_1 \cup \ldots \cup S_t$ satisfies the following requirement: The constraints in $\mathcal{S}$ can be colored by χ colors, such that

- For each S_i, if $s_1 \neq s_2 \in S_i$ have the same color, then s_1 and s_2 touch disjoint sets of coordinates.
- For each S_i and S_j, $i \neq j$, if $s_1 \in S_i$ and $s_2 \in S_j$ have the same color, then they either touch disjoint sets of coordinates, or touch exactly the same coordinates (but may differ on the coefficients).

In this section we prove the following theorem, which constructs a SPCPP that only works for CCSs.

Lemma 5.11. *There exists an SPCPP verifier V that works only when its input circuits form a χ-CCS, and when it is given the corresponding coloring as an additional input. The verifier V has randomness complexity* $\log(t+s) + O(\log\log s)$*, decision complexity* $\tilde{O}(\chi \cdot (t+s))$*, minimial field size* $O(s)$*, rejection ratio* $\Omega(1)$*, and strong* $O(\chi)$*-row/column access.*

In the rest of this section, we describe the construction of the verifier V. Fix input circuits $\varphi_1, \ldots, \varphi_t$, tested assignments $x_1, \ldots, x_t \in \mathbb{F}^m$, and proof string π. Let $S_1, \ldots, S_t$ be the corresponding sets of constraints, let $\mathcal{S} \stackrel{\text{def}}{=} S_1 \cup \ldots \cup S_t$. Suppose that we are given a coloring of $\mathcal{S}$ using χ colors as in Definition 5.10, represented as a list containing the color of each output gate of each circuit φ_i.

High level view of the construction. Our strategy is to construct for each color $c \in [\chi]$ a collection of vectors $x_1^c, \ldots, x_t^c \in \mathbb{F}^m$ and a subspace $U^c \subseteq \mathbb{F}^m$ such that the following holds: The circuits $\varphi_1, \ldots, \varphi_t$ accept $x_1, \ldots, x_t$ (respectively) if and only if the vectors $x_1^c, \ldots, x_t^c$ all belong to U^c *for every color* $c \in [\chi]$. The point is that verifying that $x_1^c, \ldots, x_t^c$ belong to U^c can be done using the simple-case SPCPP of Lemma 5.8 above. The proof string π will be required to contain the vectors $x_1^c, \ldots, x_t^c$, as well as additional information that allows verifying their consistency with $x_1, \ldots, x_t$, and proof strings for the invocation of the simple-case SPCPP.

For each color $c \in [\chi]$ and $i \in [t]$, we define the vector x_i^c as follows: Let $z_i^c \in \mathbb{F}^m$ be the vector obtained by summing all the constraints $s \in S$ of color c. Then, we define $x_i^c \stackrel{\text{def}}{=} x_i \cdot z_i^c$, where the multiplication is done coordinate-wise. In other words, we define x_i^c by setting each coordinate $j \in [m]$ of x_i^c as follows. If the coordinate j is touch by a constraint $s \in S_i$ of color c, then we define $(x_i^c)_j = (x_i)_j \cdot s_j$. Otherwise, we define $(x_i^c)_j = 0$. Note that those two definitions are indeed equivalent, since every coordinate j can be touched by at most one constraint of color c in S_i.

Next, we define the subspace $U^c \subseteq \mathbb{F}^m$ as follows. Let $s_1, \ldots, s_k \in \mathcal{S}$ be all the constraints of color c (over all input circuits). For each constraint s_i, let s_i' be the inidicator vector of the support of s_i, In other words, for each $j \in [m]$, we define $(s_i')_j = 1$ if $(s_i)_j \neq 0$, and $(s_i')_j = 0$ otherwise. We now define U^c to be the subspace described by the constraints $s_1', \ldots, s_k'$. As we prove below in Claim 5.12, it holds that $\varphi_1, \ldots, \varphi_t$ accept $x_1, \ldots, x_t$ (respectively) if and only if the vectors $x_1^c, \ldots, x_t^c$ all belong to U^c *for every color* $c \in [\chi]$. The crucial observation here is that for every x_i and $s \in S_i$ of color c, it holds that $\langle x_i, s \rangle = \langle x_i^c, s' \rangle$. Thus, if the proof string π indeed contains vectors $x_1^c, \ldots, x_t^c$ that are constructed as defined above, we are done.

It remains to verify that the vectors $x_1^c, \ldots, x_t^c$ are obtained from $x_1, \ldots, x_t$ as expected. To this end, we use multiplication codes. Let (C_A, C_B, C_M) be the multiplication codes of Fact 2.16. The verifier expects the proof string π to contain, for every $i \in [t]$ and $c \in [\chi]$, codewords $c_A \in C_A$

and $c_M \in C_M$ that are supposed to be equal to $C_A(x_i)$ and $C_A(x_i) \cdot C_B(z_i^c)$ respectively. Note that x_i^c is supposed to be a substring of $C_A(x_i) \cdot C_B(z_i^c)$ because the codes are systematic, so if $c_M = C_A(x_i) \cdot C_B(z_i^c)$, we can extract x_i^c from it. In order to check that $c_M = C_A(x_i) \cdot C_B(z_i^c)$, the verifier chooses a random coordinate j and checks that $(c_M)_j = (c_A)_j \cdot C_B(z_i^c)_j$ (note that the verifier can compute $C_B(z_i^c)$ on its own). If c_A and c_M are indeed codewords of C_A and C_M respectively, then the distance of C_M guarantees the soundness of this test.

Finally, the verifier should verify that c_A and c_M are indeed legal codewords of C_A and C_M respectively, and this should be done for all x_i's simultaneously. However, this check can be done using the simple-case SPCPP of Lemma 5.8. We conclude this overview by proving our claim regarding the subspaces U^c.

Claim 5.12. *For every $i \in [t]$, it holds that $x_i \in \mathbf{SAT}(\varphi_i)$ if and only if $x_i^c \in U^c$ for every $c \in [\chi]$.*

Proof. Fix $i \in [t]$. For the rest of the proof, we say that a vector s' is the indicator of a constaint $s \in \mathcal{S}$ if s' is the indicator vector of the support of s, as described above in the definition of U^c.

For the first direction, assume that $x_i^c \in U^c$ for every $c \in [\chi]$. We prove that $x_i \in \mathbf{SAT}(\varphi_i)$. Let $s \in S_i$ be an arbitrary constraint of $\mathbf{SAT}(\varphi_i)$. It suffices to prove that $\langle x_i, s\rangle = 0$. Suppose that s is of color c, and let s' be the indicator of s. By assumption, $x_i^c \in U^c$, and therefore $\langle x_i^c, s'\rangle = 0$. Now, by the construction of x_i^c, it holds that $\langle x_i, s\rangle = \langle x_i^c, s'\rangle = 0$, as required.

For the second direction, assume that $x_i \in \mathbf{SAT}(\varphi_i)$. Fix $c \in [\chi]$. We prove that $x_i^c \in U^c$. Let s' be one of the constraints that we used to describe U^c. It suffices to prove that $\langle x_i^c, s'\rangle = 0$. We now consider two cases:

1. Suppose there exists some constraint $s \in S_i$ of color c such that s' is the indicator of s. Then, by the construction of x_i^c, it holds that $\langle x_i^c, s'\rangle = \langle x_i, s\rangle = 0$, where the second equality follows since $x_i \in \mathbf{SAT}(\varphi_i)$.

2. Otherwise, let $s \in \mathcal{S} - S_i$ be of color c such that s' is the indicator of s. We claim that for every coordinate $j \in [m]$ that is touched by s, it holds that j is not touched by any constraint of color c in S_i. By the definition of x_i^c, this implies that all the coordinates touched by s' are zeroed in x_i^c, and therefore $\langle x_i^c, s'\rangle = 0$, as required.
   To see why the above claim is true, suppose that j was touched by some constraint $s_0 \in S_i$ of color c. Then, s_0 and s would have shared the coordinate j. On the other hand, s and s_0 can not touch exactly the same coordinates, since if they did, s' would have been an indicator of $s_0 \in S_i$. It follows that s and s_0 share a coordinate without touching exactly the same coordinate, thus contradicting the definition of legal coloring.

■

The construction of V. We turn to give a detailed description of the SPCPP verifier V. Let V_S be the simple-case SPCPP of Lemma 5.8. Let (C_A, C_B, C_M) be the triplet of multiplication codes of Fact 2.16, such that C_A and C_B have message length m, and let us denote the block length of C_A, C_B, C_M by l. Recall that C_A, C_B, C_M have rate R_C and relative distance δ_C, where R_C and δ_C are universal constants.

The verifier V expects the string $x_1 \circ \ldots \circ x_t \circ \pi$ to consist of the following matrices:

1. A $t \times l$ matrix $\overline{N}$, whose rows are the encodings of $x_1, \ldots, x_t$ via C_A. Note that since C_A is systematic, the first m coordinates of the i-th row of $\overline{N}$ contain x_i.

2. For each color $c \in [\chi]$, a matrix $\overline{N^c}$ whose i-th row is $C_A(x_i) \cdot C_B(z_i^c)$, where z_i^c is defined as in the high level description above.

3. A proof matrix for each of the invocations of the simple-case SPCPP V_S described below.

Let us denote by N, N^c the matrices that are actually given in the string $x_1 \circ \ldots \circ x_t \circ \pi$, which might or might not be equal to the matrices $\overline{N}$, $\overline{N^c}$ defined above. The verifier V also constructs, for each color $c \in [\chi]$, a $t \times l$ matrix Z^c whose i-th row is $C_B(z_i^c)$ - note that V can compute Z_c from the inputs circuits $\varphi_1, \ldots, \varphi_t$ directly, without making any queries. The verifier V now performs the following checks while recycling randomness:

1. The verifier V invokes V_S to verify that all the rows of N are codewords of C_A.

2. For each color $c \in [\chi]$, the verifier V invokes V_S to verify that all the rows of N^c are codewords of C_M, whose projection to the first m coordinates belongs to U^c. Here, U^c is defined as in the high level description above.

3. The verifier V chooses a uniformly distributed coordinate $j \in [l]$. Then, for every color $c \in [\chi]$, the verifier V checks that the j-th column of N^c equals to the coordinate-wise multiplication of the j-th columns of N and Z^c.

Observe that the foregoing invocations of V_S indeed correspond to the simple case of Section 5.3.1: in each such invocation, all the rows of the corresponding matrix are checked for membership in the same subspace. This concludes the description of the SPCPP verifier V. We turn to analyze its parameters.

The minimial field size. In order for the codes C_A, C_B and C_M to exist, the field $\mathbb{F}$ must be of size at least $l = O(m) = O(s)$. Thus, the minimial field size of V is $O(s)$.

The randomness and decision complexity of V. In order to analyze the randomness and decision complexity of V, we first need to bound the size of the linear circuits on which V invokes V_S:

1. For the first check above, V invokes V_S on a linear circuit that verifies membership in C_A. By Fact 2.16, V can construct such a linear circuit of size $\tilde{O}(m) = \tilde{O}(s)$.

2. For the second check above, V invokes V_S on a conjunction of a linear circuit that verifies membership in C_M and a linear circuit that verifies membership in U^c. Again, by Fact 2.16, a linear circuit that verifies membership in C_M can be of size $\tilde{O}(m) = \tilde{O}(s)$. As for verifying membership in U^c, note observe that verifying membership in U^c can be done by a linear circuit of size $O(m) = O(s)$: to see it, U^c is described by a collection of constraints such that no two of them share a coordinate (since they are of the same color).

Therefore, V_S is only invoked on linear circuits of size at most $\tilde{O}(s)$. Now, let $r_S(t, s)$ and $d_S(t, s)$ denote the randomness and decision complexity of V_S respectively. In order to analyze the randomness complexity of V, observe that the invocations of V_S require $r_S(t, \tilde{O}(s)) \leq \log(t + s) + O(\log \log s)$ random bits, and choosing the random coordinate $j \in [l]$ requires $\log l \leq \log s + O(1)$ random bits. Since V recycles the randomness used in the different checks, we get that the randomness complexity of V is $\log(t + s) + O(\log \log s)$, as required.

We turn to analyze the decision complexity of V. Each of the invocations of V_S outputs a linear predicate ψ of size $d_S(t, \tilde{O}(s)) = \tilde{O}(t+s)$. Moreover, the third check can be implemented by a linear circuit of size $O(t)$: in particular, note that computing the multiplication of a column of N and a column of Z^c is a linear operation, since the column Z^c can be computed by V and be hard-wired to the linear circuit as a constant scalar. Summing up the sizes of all those linear circuits, it follows that the decision complexity of V is $\tilde{O}\left(\chi \cdot (t + s)\right)$, as required.

The rejection ratio of V. Suppose that there exists an assignment x_i that is ε-far from $\mathbf{SAT}(\varphi_i)$. We prove that V rejects with probability at least $\Omega(\varepsilon)$. Let us denote by ρ_S the (constant) rejection ratio of V_S. Let

$$\tau \overset{\text{def}}{=} \min\left\{R_C \cdot \varepsilon, \delta_C/3\right\}.$$

Clearly, if one of the invocations of V_S rejects with probability at least $\rho_S \cdot \tau$, then we are done. Thus, we may assume without loss of generality that all the invocations of V_S reject with probability at most $\rho_S \cdot \tau$. This implies that:

1. Every row of N is τ-close to C_A.
2. For each color $c \in [\chi]$, every row of N^c is τ-close to a codeword of C_M, whose projection to the first m coordinates belongs to U^c.

Let N^{dec} be the matrix whose rows are obtained by decoding the rows of N to the closest codeword of C_A. For every color $c \in [\chi]$, let $N^{c,\text{dec}}$ be defined similarly with respect to N^c and C_M. We consider two cases:

- For the first case, we assume that for every $c \in [\chi]$, the matrix $N^{c,\text{dec}}$ is equal to the *coordinate-wise* multiplication of the matrices N^{dec} and Z^c. Let us denote by x_i^{dec} the projection of the i-th row of N^{dec} to the first m coordinates, and for each color $c \in [\chi]$, let $x_i^{c,\text{dec}}$ denote the projection of the i-th row of $N^{c,\text{dec}}$. By assumption, for each color $c \in [\chi]$, it holds that $x_i^{c,\text{dec}} \in U^c$, and that $x_i^{c,\text{dec}} = x_i^{\text{dec}} \cdot z_i^c$. Thus, by Claim 5.12, it follows that $x_i^{\text{dec}} \in \mathbf{SAT}(\varphi_i)$. Now, by assumption, the i-th row of N is τ-close to the i-th row of N^{dec}. Since the rate of C_A is R_C, it follows that x_i is ε-close to x_i^{dec} (as $\tau/R_C \geq \varepsilon$). However, this implies that x_i is ε-close to $\mathbf{SAT}(\varphi_i)$, which contradicts the definition of x_i.
- For the second case, we assume the opposite. That is, we assume that for some $c \in [\chi]$, the matrix $N^{c,\text{dec}}$ is different from the *coordinate-wise* multiplication of the matrices N^{dec} and Z^c. For every matrix M, let us denote by $M_{k\to}$ the k-th row of M. Then, for some index $k \in [t]$ it holds that
$$N^{c,\text{dec}}_{k\to} \neq N^{\text{dec}}_{k\to} \cdot Z^c_{k\to} = N^{\text{dec}}_{k\to} \cdot C_B(z^c_k).$$
Now, $N^{c,\text{dec}}_{k\to}$ is a codeword of C_M by definition, and $N^{\text{dec}}_{k\to} \cdot C_B(z^c_k)$ is a codeword of C_M by the multiplication property of C_A, C_B, and C_M. Thus, $N^{c,\text{dec}}_{k\to}$ and $N^{\text{dec}}_{k\to} \cdot C_B(z^c_k)$ must differ on at least δ_C fraction of their coordinates. By the triangle inequality, it follows that $N^c_{k\to}$ and $N_{k\to} \cdot C_B(z^c_k)$ must differ on at least $\delta_C - 2 \cdot \tau \geq \delta_C/3$ fraction of the coordinates. This implies that the third check of V rejects with probability $\delta_C/4 = \Omega(\varepsilon)$, as required.

This concludes the analysis of the rejection of V.

The row/column access of V. We conclude by showing that V has strong $O(\chi)$-row/column access. To this end, we first define the assignment and proof matrices of V. Observe that the assignment matrix M_x is determined by the "strongness" requirement to be the the matrix whose rows are $x_1, \ldots, x_t$. Moreover, observe that the assignment matrix is exactly the m leftmost columns of the matrix N. We define the proof matrix M_π to be the matrix obtained by concatenating the $l - m$ rightmost columns of the matrix N, the columns of the matrices N^c, and the columns of the proof matrices for the invocations of V_S. If necessary, we add dummy coordinates to make M_π rectangular.

Now, since V_S has strong 2-row/column access, every invocation of V_S query at most 2 rows and columns of the assignment and proof matrices of this invocation. Observe that for the invocation of V_S on the matrix N, the matrix N serves as the assignment matrix of V_S. Therefore, whenever this invocation of V_S queries a row (respectively, a column) of its assignment matrix, this query can be emulated by querying two rows (respectively, one column) of M_x and M_π. Moreover, whenever V_S queries a row or column of its proof matrix, this query can be emulated by querying the corresponding row or column of M_π.

For each invocation of V_S on a matrix N^c, the matrix N^c serves as the assignment matrix of V_S. Thus, each row or column of its assignment and proof matrices is contained in the a corresponding row or column of M_π.

Summing up, it follows that the invocations of V_S can emulated by querying at most $O(\chi)$ rows and columns of M_x and M_π. In addition, the third check of V queries an a column of N and a column of N^c for each $c \in [\chi]$, and all those columns are columns of either M_x or M_π. We conclude that V has $O(\chi)$-row/column access, as required.

5.3.3 Reducing the general case to CCS

In this section, we show that how to transform arbirarily linear circuits $\varphi_1, \ldots, \varphi_t$ to $O(1)$-CCS (i.e., CCS with $O(1)$ colors). We will use this reduction to construct SPCPPs in Section 5.3.4 below. The basic idea of the reduction is that every constraint system can be embedded on any routing network, and in particular one may choose a routing network whose edges can be colored using few colors. It should be mentioned that while the idea of embedding a constraint system on a routing network has been used in several prior works (e.g. [BFLS91, PS94]), the use of this technique for reducing a general constraint system to a CCS is new.

In order to embed a system of linear constraints on a routing network, we add auxiliary variables to the system in a way that essentially does not change the solution space. This notion of adding auxiliary variables while "essentially" not changing the solution space is captured by the following notion of *extension*.

Definition 5.13. Let $W \subseteq \{0,1\}^m$ and let $l \in \mathbb{N}$. We say that a subspace $W' \subseteq \{0,1\}^{m+l}$ is an extension of W if it satisfies the following property: A vector $x \in \{0,1\}^m$ belongs to W if and only if there exists a vector $y \in \{0,1\}^l$ such that $x \circ y \in W'$.

In the foregoing Definition 5.13, the vector y represents the assignment to the auxiliary variables, and the fact that the solution space remains intact is captured by the fact that the projection of W' to the coordinates in $[m]$ yields W. Our reduction of a general constraint system to a CCS can now be stated as follows.

Lemma 5.14. *There exists a polynomial time algorithm that takes as input linear circuits $\varphi_1, \ldots, \varphi_t : \mathbb{F}^m \to \mathbb{F}^p$ of size s, and outputs circuits $\varphi'_1, \ldots, \varphi'_t : \mathbb{F}^{m'} \to \mathbb{F}^{p'}$ that form an $O(1)$-CCS, such that for each $i \in [t]$ the subspace $\mathbf{SAT}(\varphi'_i)$ is an extension of $\mathbf{SAT}(\varphi_i)$. Furthermore, the circuits $\varphi'_1, \ldots, \varphi'_k$ are of size $\tilde{O}(s)$, and it holds that $m' = \tilde{O}(s)$.*

In the rest of this section we sketch prove Lemma 5.14. Fix a collection of circuits $\varphi_1, \ldots, \varphi_t$ of size s, let $W_i = \mathbf{SAT}(\varphi_i)$ for every i, and let $S_1, \ldots, S_k$ be sets of linear constraints corresponding to $\varphi_1, \ldots, \varphi_t$. For each W_i, we denote by $W'_i \stackrel{\text{def}}{=} \mathbf{SAT}(\varphi'_i)$ the extension of W_i that we seek to construct.

Warm-up. As a warm-up, consider the case in which every set S_i is a collection of disjoint equality constraints. More formally, we consider the case in which every constraint $s \in S_i$ takes the value 1 in one coordinate, the value -1 in a second coordinate, and 0 everywhere else. Moreover, we assume that every coordinate is touched by at most one constraint in S_i. We show that in this case, the sets $S_1, \ldots, S_t$ can be extended to a 4-CCS.

We would like to construct for each W_i a subspace $W'_i \subseteq \mathbb{F}^{m'}$ which is an extension of W_i, such that $W'_1, \ldots, W'_t$ form a 4-CCS. The idea that underlies the construction is the following. We identify every coordinate in $[m']$ with a vertex of a routing network G (see Section 2.5). We then embed each equality constraint of the set S_i on a path in G that connects the coordinates touched by the constraint. The 4-colorability of the corresponding constraints system follows from the fact that G is 4-edge colorable. We turn to explain this construction in detail.

Let $G \stackrel{\text{def}}{=} G_m$ be a routing network of order m obtained from Fact 2.20. Recall that G has $m' \stackrel{\text{def}}{=} \tilde{O}(m)$ vertices, and that it has a special set of vertices T with the following property: For every permutation $\sigma : T \to T$, one can find in polynomial time a collection $\mathcal{P}$ of vertex-disjoint paths that connect each vertex $v \in T$ to $\sigma(v)$. We identify the vertices of G with the coordinates in $[m']$, and identify the vertices of T with the coordinates of $[m]$.

We now construct a set S'_i of constraints that describes the subspace $W'_i \subseteq \mathbb{F}^{m'}$ for each $i \in [t]$ as follows. We find a collection $\mathcal{P}$ of vertex-disjoint paths on G, such that for each equality constraint $s \in S_i$, there is a path in $\mathcal{P}$ that connects the coordinates that s touches. Then, for each edge e of G, we put in S'_i an equality constraint between the endpoints of e if and only if e participates in one of the aforementioned vertex-disjoint paths.

It should be clear that each W'_i is an extension of W_i. To see that $W'_1, \ldots, W'_t$ form a 4-CCS, let $\mathcal{S}' \stackrel{\text{def}}{=} S'_1 \cup \ldots \cup S'_k$ and observe that every constraint in $\mathcal{S}'$ is an equality constraint between the endpoints of some edge of G. Now, recall that by Fact 2.20, the network G is 4-edge colorable. This implies the constraints in $\mathcal{S}'$ can be colored using 4 colors such that no two constraints of the same color touch the same coordinate. Thus, $W'_1, \ldots, W'_k$ and $\mathcal{S}'$ satisfy Definition 5.10 with $\chi = 4$. This concludes the construction.

Handling arbitrary linear circuits. It remains to show how to transform arbitrary linear circuits $\varphi_1, \ldots, \varphi_t$ to an $O(1)$-CCS as in Claim 5.14. The proof strategy is to construct for each subspace W_i an extension $U_i \subseteq \mathbb{F}^{2s}$ that is described by constraints of two types: The constraints of the first type form an $O(1)$-CCS. The constraints of the second type are equality constraints. We then complete the proof by handling the second type equality constraints in the same way as in the "warm-up" case of Section 5.3.3.

We turn to describe how to construct the subspace $U_i \subseteq \mathbb{F}^{2s}$. For each wire v of φ_i, we associate v with two coordinates in $[2s]$, denoted v^{in} and v^{out}. A vector y belongs to U_i if and only if it satisfies the following two types of constraints

1. **Consistency constraints:** For every wire v in φ_i, we have the linear constraint $y_{v^{\text{in}}} = y_{v^{\text{out}}}$.
2. **Computation constraints:** For each gate g in φ_i, we have the constraints defined as follows. Recall that g computes a linear combination of its inputs. Let us denote by $\alpha_1, \alpha_2 \in \mathbb{F}$ the coefficients of this combination, such that when given inputs x_1 and x_2, the gate g outputs $\alpha_1 \cdot x_1 + \alpha_2 \cdot x_2$. Let v_1 and v_2 be the incoming wires of g and let v_3 and v_4 be its outgoing wires. Then, the constraints that correspond to g are $y_{v_3^{\text{out}}} = \alpha_1 \cdot y_{v_1^{\text{in}}} + \alpha_2 \cdot y_{v_2^{\text{in}}}$ and $y_{v_4^{\text{out}}} = \alpha_1 \cdot y_{v_1^{\text{in}}} + \alpha_2 \cdot y_{v_2^{\text{in}}}$.
   In addition, for each output wire v of φ_i, we have a constraint of the form $y_{v^{\text{out}}} = 0$.

Observe that each U_i is indeed an extension of W_i.

Now, let us re-arrange the coordinates in $[2s]$ such that, if a gate g of φ_i has incoming wires v_1, v_2 and outgoing wires v_3, v_4, then the coordinates v_1^{in}, v_2^{in}, v_3^{out}, v_4^{out} are consecutive as numbers.

The crucial observation is that when using this ordering, the computation constraints touch the same coordinates for every subspace U_i. In other words, in terms of coloring, the only difference between distinct subspaces U_i, U_j is the consistency constraints. We use this property to argue that the computation constraints form a CCS.

More specifically, observe that using the latter ordering, the coordinates of $[2 \cdot s]$ can be partitioned to disjoint quadruples, such that each computation costraint of any subspace U_i touches only coordinates within a single quadruple. This means that the support of each constraint in any U_i can be one of at most 16 possible supports - which are all the possible subsets of a set of four coordinates. It follows that we can color the computation constraints using 16 colors, such that the following holds: for every two computations constraints, not necesarily of the same subspace U_i, that have the same color, either the constraints have the same support, or they have disjoint supports. In other words, the computation constraints of the subspaces $U_1, \ldots, U_t$ are $O(1)$-CCS[9].

It remains to handle the consistency constraints. This is done as in the "warm-up" case of Section 5.3.3, by embedding the consistency constraints on a routing network, and coloring them using additional 4 colors. Note that this embedding requires to construct for each U_i an extension $W_i' \subseteq \mathbb{F}^{s'}$ for $m' = \tilde{O}(s)$. Finally, we take $W_1', \ldots, W_t'$ to be the required 20-CCS. Note that each W_i' is indeed an extension of W_i, since W_i' is an extension of U_i which is in turn an extension of W_i.

5.3.4 Proof of Theorem 5.5

In this section, we finish the proof of Theorem 5.5, restated next, by combining Lemma 5.11 with Lemma 5.14.

Theorem (5.5, restated)**.** *There exists an SPCPP verifier with randomness complexity* $\log(t+s) + O(\log\log(s))$*, decision complexity* $\tilde{O}(t+s)$*, minimial field size* $\tilde{O}(s)$*, rejection ratio* $\Omega(1)$*, and strong* $O(1)$*-row/column access.*

We construct a SPCPP verifier V with the required parameters. Fix linear circuits $\varphi_1, \ldots, \varphi_t : \mathbb{F}^m \to \mathbb{F}^p$ of size s, tested assignments $x_1, \ldots, x_t$, and proof string π. The verifier V starts by invoking the algorithm of Lemma 5.14 on the circuits $\varphi_1, \ldots, \varphi_t$, thus obtaining linear circuits $\varphi_1', \ldots, \varphi_t' : \mathbb{F}^{m'} \to \mathbb{F}^{p'}$ that are $O(1)$-CCS, and such that $\mathbf{SAT}(\varphi_i')$ is an extension of $\mathbf{SAT}(\varphi_i)$ for each $i \in [t]$. Moreover, it holds that $m' = \tilde{O}(s)$ and the circuits $\varphi_1', \ldots, \varphi_t'$ are of size $\tilde{O}(s)$.

We would now like to invoke the SPCPP for CCSs of Lemma 5.11 on $\varphi_1', \ldots, \varphi_t'$. However, before we do it, we first apply a standard "reweighting" trick to make sure that the original tested assignments form half of the new tested assignments. To this end, for each linear circuit φ_i', the verifier V constructs a new linear circuit $\varphi_i'' : \mathbb{F}^{m''} \to \mathbb{F}^{p''}$ that is defined as follows. Let $l = \lceil m'/m \rceil$. The circuit φ_i'' takes as input a vector x'' of length $m'' = (l-1) \cdot m + m'$, and accepts if and only if the following two conditions hold:

- The first $l \cdot m$ coordinates of x'' contain l identical copies of a vector $x \in \mathbb{F}^m$.
- The last m' coordinates of x_i'' form a vector $x' \in \mathbb{F}^{m'}$ that is accepted by φ_i'. Note that the first m coordinates of x' contain the vector x from the first condition.

Clearly, the linear circuit φ_i'' can be implemented in size $\tilde{O}(s)$, and $\mathbf{SAT}(\varphi_i'')$ is an extension of $\mathbf{SAT}(\varphi_i)$. It is also easy to see that the linear circuits $\varphi_1'', \ldots, \varphi_t''$ form an $O(1)$-CCS: we can use the coloring of $\varphi_1', \ldots, \varphi_t'$ to color the constraints of the second condition above, and an additional color to color the equality constraints of the first condition.

[9] In fact, a more careful examination shows that 5 colors are sufficient.

Let us denote by V_C the SPCPP verifier for CCSs of Lemma 5.11. The verifier V expects the proof string π to contain vectors $y_1, \ldots, y_t$ such that $x_i \circ y_i \in \mathbf{SAT}(\varphi_i')$ for each $i \in [t]$ (*note that we use here* φ_i' *and not* φ_i''). Now, for each $i \in [t]$, let

$$x_i'' = \underbrace{x_i \circ \ldots \circ x_i}_{l} \circ y_i.$$

The proof string π is also expected to contain a proof string π_C that convinces V_C that $\varphi_1'', \ldots, \varphi_t''$ accept $x_1'', \ldots, x_t''$. The verifier V finishes by emulating V_C on tested assignments $x_1'', \ldots, x_i''$ and proof string π_C, and and accept if and only V_C accepts. Observe that this emulation can be performed simply by redirecting the queries of V_C to the copies of x_i in x_i'' to the corresponding coordinates of x_i.

It is not hard to see that the randomness complexity, decision complexity, and minimal field size of V are $\log(t+s) + O(\log\log(s))$, $\tilde{O}(t+s)$, and $\tilde{O}(s)$ respectively. Furthermore, the strong $O(1)$-row/column access of V follow easily from the strong $O(1)$-row/column of V_C. It remains to analyze rejection ratio of V.

Let ρ_C be the (constant) rejection ratio of V_C. Suppose that x_i is ε-far from $\mathbf{SAT}(\varphi_i)$ for some $i \in [t]$ and $\varepsilon > 0$. Then, since the copies of x_i form half of the vector x_i'', it follows that x_i'' is $\varepsilon/2$-far from $\mathbf{SAT}(\varphi_i'')$. This implies that V_C rejects $x_1'', \ldots, x_t''$ with probability at least $\rho_C \cdot \varepsilon/2$. It follows that V rejects with probability at least $\rho_C \cdot \varepsilon/2 \geq \Omega(\varepsilon)$, as required.

6 Proof of the main theorems

In this section, we show how to prove our main theorems, restated below, by combining the tools that were developed in the previous sections.

Theorem (1.1, main theorem, restated)**.** *For every time-constructible* $t : \mathbb{N} \to \mathbb{N}$ *and every language* $L \in \mathbf{NTIME}(t)$*, there exists a PCP verifier for* L *with proof length* $t \cdot (\log(t))^{O(\log\log t)}$*, randomness complexity* $\log t + O(\log^2 \log t)$*, query complexity* $O(1)$*, and rejection probability* $\Omega(1)$*.*

Theorem (2.5, main PCPP theorem, restated)**.** *Let* $t : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ *and let* PL *be a pair-language that is decidable in time* t*. Then, there exists a PCPP verifier for* PL *with randomness complexity* $\log t + O(\log^2 \log t)$*, decision complexity* $O(1)$*, and rejection ratio* $\Omega(1)$*.*

As we showed in Section 2.3.2, Theorem 2.5 implies Theorem 1.1, so it suffices to prove Theorem 2.5. Fix a time-constructible function $t : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ and a pair-language PL that is decidable in time t. We construct PCPPs with the required parameters for PL. Our starting point is the linear PCPP that was constructed in Theorem 5.1, which we denote here by V_1. The verifier V_1 has randomness complexity $\frac{1}{2}\log n + O(\log\log n)$, decision complexity $\tilde{O}(\sqrt{n})$, minimial field size $\tilde{O}(\sqrt{n})$, rejection ratio $\Omega(1)$, and $O(1)$-row/column access. We begin by applying the robustization technique of Theorem 4.7 to V_1, resulting in a linear PCPP verifier V_2 that has randomness complexity $\frac{1}{2}\log n + O(\log\log n)$, decision complexity $\tilde{O}(\sqrt{n})$, and *robustness* $\Omega(1)$.

Our next step is to compose V_2 with itself for $\log\log n$ times[10], resulting in a linear PCPP verifier V_3 that has randomness complexity $\log n + O(\log^2 \log n)$, decision complexity $O(1)$, and rejection ratio $1/\operatorname{poly}\log n$. Then, we apply the transformation of linear PCPPs to (general) PCPPs of Theorem 3.1, resulting in a PCPP verifier V_4 for PL, which has randomness complexity $\log t + O(\log^2 \log t)$, decision complexity $O(\log t)$, and rejection ratio $1/\operatorname{poly}\log t$.

[10]Formally, this is done by applying for $\log\log n$ times the universal algorithm whose existence is stated in Theorem 2.11, each time setting V_{out} to be the result of the previous iteration.

Finally, we apply transformation of Theorem 2.13 to V_5, which reduces the decision complexity to $O(1)$ and amplifies the rejection ratio to $\Omega(1)$ using Dinur's gap amplification technique. This results in a PCPP verifier V_6 for PL, which has randomness complexity $\log t + O(\log^2 \log t)$, constant decision complexity, and constant rejection ratio. Note that by the standard bounds that werre discussed in Section 2.3.1, the verifier V_6 has proof length $t \cdot (\log t)^{O(\log \log t)}$ and constant query complexity. We conclude that V_6 is the required PCPP verifier.

7 Variations of our construction

7.1 Alternative construction

In this section we sketch an alternative proof for our main theorems. This advantage of this alternative proof is that it might give more intuition for why our construction works, and may be easier to understand at the high level. The disadvantage of this proof is that it is less modular than the construction presented in the foregoing sections. We turn to sketch the proof in more detail.

We would like to construct a PCPP verifier with randomness complexity $\log t + O(\log^2 \log t)$, decision complexity $O(1)$, and rejection ratio $\Omega(1)$. We will focus on constructing PCPP for the pair-language of CIRCUITEVAL, which consists of pairs (φ, x) where $\varphi : \{0,1\}^m \to \{0,1\}$ is a boolean circuit of size n, and $x \in \{0,1\}^m$ is a satisfying assignment of φ. It suffices to construct a PCPP for CIRCUITEVAL that has randomness complexity $\log n + O(\log^2 \log n)$, decision complexity $O(1)$, and rejection ratio $\Omega(1)$, since such a PCPP would imply the required PCPPs for any pair-language via standard techniques.

This time, we construct the PCPP directly, rather than first constructing a linear PCPP and then transforming it to a general PCPP. As in our previous construction, it suffices to construct a PCPP for CIRCUITEVAL with randomness complexity $\frac{1}{2} \log n + O(\log \log n)$, decision complexity $\tilde{O}(\sqrt{n})$, rejection ratio $\Omega(1)$, and $(\text{poly} \log n)$-row/column access. Given such a PCPP, one can obtain the final PCPP by applying robustization, composition, and gap amplification, as we did in Section 6 above. Moreover, as in Section 5, it actully suffices to construct SPCPPs. Here, we use a variant of SPCPPs that works with boolean circuits rather linear circuits defined as follows.

Definition 7.1. Let $r, d, \ell, F : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, $\rho : \mathbb{N} \times \mathbb{N} \to (0,1)$. A (general) simultaneous PCPP (SPCPP) verifier V with randomness complexity r, decision complexity d, proof length ℓ, and rejection ratio ρ is a probabilistic polynomial time machine that satisfies the following requirements:

1. **Input:** The verifier V takes as input t boolean circuits $\varphi_1, \ldots, \varphi_t : \{0,1\}^m \to \{0,1\}$ of size s.
2. **Output:** The verifier V outputs a tuple I of coordinates in $[m + \ell(t,s)]$, and a boolean circuit $\psi : \{0,1\}^{|I|} \to \{0,1\}$ of size at most $d(t,s)$.
3. **Randomness complexity:** On every input circuits $\varphi_1, \ldots, \varphi_t$, and on every sequence of coin tosses, V tosses at most $r(t,s)$ coins.
4. **Completeness:** For every $x_1 \in \mathbf{SAT}(\varphi_1), \ldots, x_t \in \mathbf{SAT}(\varphi_t)$, there exists a string $\pi \in \{0,1\}^{\ell(t,s)}$ such that
$$\Pr\left[\psi\left((x_1 \circ \ldots \circ x_t \circ \pi)_{|I}\right) = 1\right] = 1,$$
where ψ and I are generated by the verifier V on input $(\varphi_1, \ldots, \varphi_t)$.

5. **Soundness:** For every $x_1, \ldots, x_t \in \{0,1\}^m$ such that x_i is ε-far from $\mathbf{SAT}(\varphi_i)$ for some $i \in [t]$ and some $\varepsilon > 0$, and for every string $\pi \in \{0,1\}^{\ell(t,s)}$, it holds that

$$\Pr\left[\psi\left((x_1 \circ \ldots \circ x_t \circ \pi)_{|I}\right) = 0\right] \geq \rho(t,s) \cdot \varepsilon,$$

where ψ and I are generated by the verifier V on input $(\varphi_1, \ldots, \varphi_t)$.

As in Section 5, it suffices to construct the SPCP stated in the following theorem. In what follows, the definition of strong row/column access is adapted to general SPCPPs in the natual way.

Theorem 7.2. *There exists a (general) SPCPP verifier with randomness complexity* $\log(t+s) + O(\log\log(s))$*, decision complexity* $\tilde{O}(t+s)$*, rejection ratio* $\Omega(1)$*, and strong* $(\operatorname{poly}\log n)$*-row/column access.*

Now, the main difference between the construction presented in this section and the previous construction is the way we construct the SPCPPs of Theorem 7.2. We first recall how we constructed the linear PCPPs in previous construction: We first constructed SPCPPs for the simple case in which all the input circuits $\varphi_1, \ldots, \varphi_t$ were identical, which was relatively easy. Then, the bulk of the work went into reducing the general case to a constant number of instances of this simple case. In contrast, in the current construction, reducing the general case to the foregoing simple case is almost trivial, and the bulk of the work is required to handle the simple case.

We first explain how to reduce the general case to the simple case in which $\varphi_1 = \ldots = \varphi_t$. The key tool that we use is universal circuits. Roughly, a universal circuit U is a boolean circuit that takes as input a boolean circuit φ and an assignment x to φ, and outputs $\varphi(x)$. It is possible to construct such circuits of quasilinear size. Now, suppose we wish to construct a general SPCPP that is invoked on arbitrary circuits $\varphi_1, \ldots, \varphi_t$ and tested assignments $x_1, \ldots, x_t$. Then, instead of trying to verify directly that each input circuit φ_i accepts the assignment x_i, we can verify that the universal circuit U accepts the assignments $(\varphi_1, x_1), \ldots, (\varphi_t, x_t)$. Using the latter idea, we are left with the task of verifying that t identical copies of U verify t tested assignments, which brings us to the simple case discussed above, as required. We note that in order to implement this idea some more technical work is required, and refer the reader to [Mei09, Section 5.7] for a detailed implementation of a similar idea.

Constructing an SPCPP for the simple case. It remains to construct an SPCPP verifier V that is invoked on a circuit φ of size s and tested assignments $x_1, \ldots, x_t$ for φ, and verifies that all the tested assignments satisfy φ (as in Definition 7.1). As noted above, this task would have been easy if φ was linear. Thus, our strategy will be to reduce φ to a linear circuit φ_L. To this end, we use the same technique that we used in Theorem 3.1 to reduce the construction of general PCPPs to linear PCPPs.

In order to use this technique in the current context, we would like to take a different view of Theorem 3.1. Rather than viewing this theorem as a construction a general PCPP using a linear PCPP, we we view this theorem as a construction of the following "weird PCPP", which is described in the following theorem. Basically, this "weird PCPP" outputs small queries tuple I and boolean predicate ψ, just like a standard PCPP, but in addition outputs a large linear circuit φ_L which may access the whole tested assignment and proof string. The weird verifier accepts only if both the small predicate ψ and the linear circuit φ_L accept.

Theorem 7.3 (Alternative view of Theorem 3.1)**.** *Let* $\ell : \mathbb{N} \to \mathbb{N}$ *be a function such that* $\ell(n) = \tilde{O}(n)$*. There exists probabilistic polynomial time machine* V_L *(the "weired PCPP") that satisfies the following requirements:*

1. ***Input:*** *The verifier V_L takes as input a boolean circuit $\varphi : \{0,1\}^m \to \{0,1\}$ of size n.*

   (a) ***Output:*** *The verifier V_L outputs a tuple I_L of coordinates in $[m + \ell(n)]$, a boolean circuit $\psi_L : \{0,1\}^{|I_L|} \to \{0,1\}$ of size at most $\operatorname{poly}\log n$, and a linear circuit $\eta_L : \mathbb{F}^{m+\ell(n)} \to \mathbb{F}^p$ over a field $\mathbb{F}$ of size $O(n)$.*

2. ***Randomness complexity:*** *On every input circuit φ, and on every sequence of coin tosses, V_L tosses at most $\log n + O(\log\log n)$ coins.*

3. ***η_L is deterministic:*** *The linear circuit η_L is independent of the coin tosses of V_L.*

4. ***Completeness:*** *For every $x \in \mathbf{SAT}(\varphi)$, there exists a string $\pi \in \{0,1\}^{\ell(n)}$ such that $\eta_L(x \circ \pi)$ accepts, and*

$$\Pr\left[\psi_L\left((x \circ \pi)_{|I_L}\right) \text{ accepts}\right] \quad = \quad 1$$

   where ψ_L and I_L are generated by the verifier V_L on input φ, and where in the second equation, the binary string $x \circ \pi$ is viewed as encoding a vector over $\mathbb{F}$.

5. ***Soundness:*** *For every $x \in \{0,1\}^m$ and every string $\pi \in \{0,1\}^{\ell(n)}$, one of the following holds*

$$\begin{aligned} \Pr\left[\psi_L\left((x \circ \pi)_{|I_L}\right) \text{ rejects}\right] &\geq \rho(n) \cdot \operatorname{dist}(x, \mathbf{SAT}(\varphi)) \\ \operatorname{dist}(x \circ \pi, \mathbf{SAT}(\eta_L)) &\geq \Omega(\operatorname{dist}(x, \mathbf{SAT}(\varphi))) \end{aligned}$$

   where ψ_L and I_L are generated by the verifier V_L on input φ, and where in the second equation, the binary string $x \circ \pi$ is viewed as encoding a vector over $\mathbb{F}$.

Let us go back to the construction of the SPCPP verifier V. When invoked on input circuit φ, tested assignments $x_1, \ldots, x_t$, and proof string π, the verifier V starts by invoking the weird verifier V_L on φ, thus obtaining a queries tuple I_L, a boolean predicate ψ_L of size $\operatorname{poly}\log s$, and a linear circuit η_L of size $O(s)$. The SPCPP verifier V expects the proof string π to contain proof strings $\pi_1, \ldots, \pi_t$, where π_i is a proof string for V_L that satisfies that:

- ψ_L accepts $(x_i \circ \pi_i)_{I_L}$.
- η_L accepts $x_i \circ \pi_i$, when viewed as encoding a vector over $\mathbb{F}$.

Now, the point is that the first condition can be verified simultaenously for all the tested assignments $x_1, \ldots, x_t$ by using only $t \cdot \operatorname{poly}\log s$ queries, simply by invoking ψ_L separately on each $(x_i \circ \pi_i)_{I_L}$. The second condition can be verified simultaenously for all the tested assignments $x_1, \ldots, x_t$ by using the *linear SPCPP for the simple case* of Section 5.3.1, i.e., by invoking it on input circuit η_L and tested assignments $x_1 \circ \pi_1, \ldots, x_t \circ \pi_t$.

More formally, the verifier V invokes the linear SPCPP V_S of Lemma 5.8 on η_L, thus obtaining linear circuit ψ_S of size $\tilde{O}(t + s)$ and a queries tuple I_S. The verifier V also expects the proof string π to contain a proof string π_S for this invocation of V_S, where π_S is thought of as a vector over $\mathbb{F}$ this is represented in binary in π. Finally, the verifier V outputs:

- A queries tuple I which consists of the coordinates of all the strings $(x_i \circ \pi_i)_{I_L}$, and also of the coordinates in the queries tuple I_S.
- A boolean predicate ψ that accepts if and only if the following conditions hold:

– ψ_L accepts $(x_i \circ \pi_i)_{I_L}$ for each $i \in [t]$.

– ψ_S accepts

$$((x_1 \circ \pi_1) \circ \ldots \circ (x_t \circ \pi_t) \circ \pi_S)_{|I_S}$$

when viewed as a vector over $\mathbb{F}$.

It is not hard to see that V has decision complexity $\tilde{O}(t+s)$. It is also not hard to prove that V has rejection ratio $\Omega(1)$, by combining the soundness properties of the weird verifier V_L and the linear SPCPP V_S. It is also possible to show, using a careful implementation, that V has strong poly $\log(n)$-row/column access.

The only issue that requires some attention is the randomness complexity. The construction of V as we described it above has randomness complexity $\approx 2\log(t+s)$, which is too much. The reason is that we need $\approx \log(s)$ random bits to invoke the weird verifier V_L, and another $\approx \log(t+s)$ random bits to invoke the linear SPCPP V_S. The solution is to use the same random bits in both invocations. However, at a first glance, one may worry that this would harm the soundness of V: Recall that V invokes V_S on η_L, which was output by V_L. If V_S uses the same random bits as V_L, it may seem as if the random bits of V_S depend on its input η_L, in which case V_S is not guaranteed to be sound. The crucial observation here is that by Theorem 7.3, the linear circuit η_L is independent of the random bits used by V_L. This means that the random bits used by V_S are indeed independent of η_L, even though they are shared with V_L. Therefore, the soundness analysis can be performed as before. This concludes our proof sketch of Theorem 7.2.

7.2 Purely combinatorial construction

In this section, we show that it is possible to modify our construction such that it does not use polynomials at all, thus obtaining a purely combinatorial construction. However, this modification comes at the cost of increasing the proof length to $t^4 \cdot (\log n)^{O(\log\log n)}$. We believe that using the ideas of Section 7.1, it is possible to improve the foregoing proof length to $t^2 \cdot (\log t)^{O(\log\log t)}$, but we have not verified it. While this result is far from the state-of-the-art PCPs, it is still an improvement over the previous combinatorial PCPs of [Din07], which have proof length $n^{O(1)}$, where the power is an unspecified constant that can be expected to be very large.

Recall that we used polynomials only to construct the multiplication codes of Fact 2.16. Now, it was shown in [Mei10] that it is possible to construct multiplication codes of quadratic length without using polynomials. Thus, in order to remove the polynomials from our construction, we use the multiplication codes of [Mei10] instead of the polynomial codes of Fact 2.16. The following fact states the properties of the multiplication codes that we use, which can be obtained via a simple modification of [Mei10, Proposition 3.10].

Fact 7.4 (Variant of [Mei10, Proposition 3.10]). *There exist constants $\delta_C, R_C > 0$ such that for every finite field $\mathbb{F}$ and every $k \in \mathbb{N}$ the following holds: there exists a triplet (C_A, C_B, C_M) of systematic linear codes over $\mathbb{F}$ that have the following properties:*

1. ***Multiplication:*** *For every $c_A \in C_A$ and $c_B \in C_B$ it holds that $c_A \cdot c_B \in C_M$.*
2. *C_A and C_B have message length k, and C_M has message length k^2.*
3. *C_A, C_B, and C_M all have block length $\ell \overset{\text{def}}{=} k^2/R_C$, and relative distance δ_C.*

Furthermore, the exists an algorithm that when given as input k and $\mathbb{F}$, runs in time that is polynomial in k and $\log|\mathbb{F}|$, and outputs linear circuits φ_A, φ_B, and φ_M of size $\tilde{O}(k^2)$ that recognize C_A, C_B, and C_M respectively.

Recall that we used the multiplication codes in two places: we used them in order to transform a linear PCPP to a general PCPP in Section 2.16, and in order to construct linear SPCPPs for CCSs in Section 5.3.2. By substuting the codes of Fact 7.4 into those constructions, we obtain the following analogues of Theorem 3.1 and Lemma 5.11.

Theorem 7.5 (Purely combinatorial variant of Theorem 3.1). *Suppose that there exists a linear PCPP verifier V with randomness complexity $r(n)$, decision complexity $d(n)$, rejection ratio $\rho(n)$, and minimal field size $F(n)$. Then, for every time-constructible $t : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ and every pair-language PL that is decidable in time t, there exists a PCPP verifier V' for PL with randomness complexity* $\max\left\{2\log t, r\left(\tilde{O}(t^2)\right)\right\}+O(1)$*, decision complexity* $O\left(d\left(\tilde{O}(t^2)\right)\cdot \operatorname{poly}\log\left(F(t)\right)\right)$*, and rejection ratio* $\Omega\left(\rho\left(\tilde{O}(t^2)\right)\right)$.

Lemma 7.6 (Purely combinatorial variant of Lemma 5.11). *There exists an SPCPP verifier V that works only when its input circuits form a χ-CCS, and when it is given the corresponding coloring as an additional input. The verifier V has randomness complexity* $\log(t+s^2)+O(\log\log s)$*, decision complexity* $\tilde{O}(\chi\cdot(t+s^2))$*, rejection ratio* $\Omega(1)$*, and strong* $O(\chi)$*-row/column access.*

Next, we discuss how to construct a linear PCPP using Lemma 7.6. By substituting Lemma 7.6 into our construction of SPCPPs in Section 5.3.4, we obtain SPCPPs with the same parameters as in Lemma 7.6. In order to obtain linear PCPPs, we would like to combine those SPCPPs with the circuit decomposition of Lemma 5.6. However, it is now no longer optimal to decompose a linear circuit of size n to $t=\tilde{O}(\sqrt{n})$ circuits of size $\tilde{O}(\sqrt{n})$. Instead, the optimal choice is $t=\tilde{O}(n^{2/3})$ and $s=\tilde{O}(n^{1/3})$. Fortunately for us, it is not hard to modify to proof of Lemma 5.6 in [Mei09] to yield such a decomposition. Using such a decomposition and the foregoing SPCPPs, we obtain the following construction of linear PCPPs.

Theorem 7.7 (Purely combinatorial variant of Theorem 5.1). *There exists a linear PCPP with randomness complexity* $\frac{2}{3}\log n+O(\log\log n)$*, decision complexity* $\tilde{O}(n^{2/3})$*, rejection ratio* $\Omega(1)$*, and* $O(1)$*-row/column access.*

Note that Theorem 7.7 does not mention a minimial field size - the reason is that this construction works for every field size, since the codes of Fact 7.4 exist for every field size. In other words, the minimal field size can be taken to be 2.

We proceed as in Section 6. We robustize the linear PCPPs of Theorem 7.7 using Theorem 4.7, thus obtaining a linear PCPP with robustness $\Omega(1)$, randomness complexity $\frac{2}{3}\log n+O(\log\log n)$, and decision complexity $\tilde{O}(n^{2/3})$. We then compose the latter verifier with itself for $\log_{\frac{2}{3}}\log n$ times, thus obtaining a linear PCPP with randomness complexity[11] $2\log n+O(\log^2\log n)$, decision complexity $O(1)$, and rejection ratio $1/\operatorname{poly}\log n$. Next, we apply Theorem 7.5 to transform the latter linear PCPP to a general PCPP of randomness complexity $4\log t+O(\log^2\log t)$, decision complexity $O(1)$, and rejection ratio $1/\operatorname{poly}\log t$. Finally, we apply Theorem 2.13 to the latter PCPP, thus obtaining the following result.

Theorem 7.8 (Purely combinatorial variant of Theorem 2.5). *Let $t:\mathbb{N}\times\mathbb{N}\to\mathbb{N}$ and let PL be a pair-language that is decidable in time t. Then, there exists a PCPP verifier for PL with proof length* $t^4\cdot(\log t)^{O(\log\log t)}$*, randomness complexity* $4\log t+O(\log^2\log t)$*, decision complexity* $O(1)$*, and rejection ratio* $\Omega(1)$.

[11]The reason the randomness complexity is $2\log n+O(\log^2\log n)$ is as follows: the $O(\log^2\log n)$ comes from accumulating a term of $O(\log\log n)$ every iteration for $O(\log\log n)$ iterations. As for the $2\log n$ term, note that the i-th composition increases the randomness complexity by $(2/3)^i\cdot\log n$. Thus, the total randomness complexity accumulated this way is at most $\sum_{i=1}^{\infty}(2/3)^i\cdot\log n=2\log n$, as required.

Note that while our linear PCPP had proof length $n^2 \cdot (\log n)^{O(\log\log n)}$, our final PCPP has proof length $t^4 \cdot (\log t)^{O(\log\log t)}$. The reason for this loss is the use of Theorem 7.5. We believe that it is possible to avoid this loss by constructing general PCPPs directly, as done in Section 7.1, rather than first constructing linear PCPPs and then converting them to general PCPPs. However, we have not verified this claim.

Acknowledgement. The author is grateful to Oded Goldreich for many useful discussions and ideas, and for comments that significantly improved the presentation of this work.

References

[ALM$^+$98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and intractability of approximation problems. *Journal of ACM*, 45(3):501–555, 1998. Preliminary version in FOCS 1992.

[AS98] Sanjeev Arora and Shmuel Safra. Probabilistic checkable proofs: A new characterization of NP. *Journal of ACM volume*, 45(1):70–122, 1998. Preliminary version in FOCS 1992.

[BFLS91] Lászĺo Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *STOC*, pages 21–31, 1991.

[BGH$^+$05] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil P. Vadhan. Short pcps verifiable in polylogarithmic time. In *IEEE Conference on Computational Complexity*, pages 120–134, 2005.

[BGH$^+$06] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil Vadhan. Robust PCPs of proximity, shorter PCPs and applications to coding. *SIAM Journal of Computing*, 36(4):120–134, 2006.

[BHLM09] Eli Ben-Sasson, Prahladh Harsha, Oded Lachish, and Arie Matsliah. Sound 3-query PCPPs are long. *TOCT*, 1(2), 2009.

[BKK$^+$13] Eli Ben-Sasson, Yohay Kaplan, Swastik Kopparty, Or Meir, and Henning Stichtenoth. Constant rate pcps for circuit-sat with sublinear query complexity. *Electronic Colloquium on Computational Complexity (ECCC)*, 85, 2013.

[BS06] Eli Ben-Sasson and Madhu Sudan. Robust locally testable codes and products of codes. *Random Struct. Algorithms*, 28(4):387–402, 2006. Preliminary version in APPROX-RANDOM 2004.

[BS08] Eli Ben-Sasson and Madhu Sudan. Short PCPs with polylog query complexity. *SIAM J. Comput.*, 38(2):551–607, 2008. Preliminary version in STOC 2005.

[BV09a] Eli Ben-Sasson and Michael Viderman. Composition of semi-ltcs by two-wise tensor products. In *APPROX-RANDOM*, pages 378–391, 2009.

[BV09b] Eli Ben-Sasson and Michael Viderman. Tensor products of weakly smooth codes are robust. *Theory of Computing*, 5(1):239–255, 2009.

[Din07] Irit Dinur. The PCP theorem by gap amplification. *Journal of ACM*, 54(3):241–250, 2007. Preliminary version in STOC 2006.

[DM10] Irit Dinur and Or Meir. Derandomized parallel repetition of structured PCPs. In *IEEE Conference on Computational Complexity*, pages 16–27, 2010. Full version can be obtained as ECCC TR10-107.

[DR06] Irit Dinur and Omer Reingold. Assignment testers: Towards combinatorial proof of the PCP theorem. *SIAM Journal of Computing*, 36(4):155–164, 2006.

[DSW06] Irit Dinur, Madhu Sudan, and Avi Wigderson. Robust local testability of tensor products of ldpc codes. In *APPROX-RANDOM*, pages 304–315, 2006.

[GS00] Oded Goldreich and Shmuel Safra. A combinatorial consistency lemma with application to proving the PCP theorem. *SIAM J. Comput.*, 29(4):1132–1154, 2000.

[GS06] Oded Goldreich and Madhu Sudan. Locally testable codes and PCPs of almost linear length. *Journal of ACM*, 53(4):558–655, 2006. Preliminary version in FOCS 2002, pages 13-22.

[Köt92] Ralf Kötter. A unified description of an error locating procedure for linear codes. In *Proceedings of the International Workshop on Algebraic and Combinatorial Coding Theory*, pages 113–117, 1992.

[Lei92] F. Thomson Leighton. *Introduction to parallel algorithms and architectures: array, trees, hypercubes*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1992.

[Mei09] Or Meir. Combinatorial pcps with efficient verifiers. In *FOCS*, pages 463–471, 2009. To appear in SIAM Journal on Computing. A more elaborated version is available as ECCC TR11-104.

[Mei10] Or Meir. IP = PSPACE using error correcting codes. *Electronic Colloquium on Computational Complexity (ECCC)*, (137), 2010. To appear in SIAM Journal on Computing.

[MS88] Florence Jessie MacWilliams and Neil James Alexander Sloane. *The theory of error correcting codes*. Elsevier/North-Holland, Amsterdam, 1988.

[Pel92] Ruud Pellikaan. On decoding by error location and dependent sets of error positions. *Discrete Mathematics*, 106-107:369–381, 1992.

[PF79] Nicholas Pippenger and Michael J. Fischer. Relations among complexity measures. *J. ACM*, 26(2):361–381, 1979.

[PS94] Alexander Polishchuk and Daniel A. Spielman. Nearly-linear size holographic proofs. In *STOC*, pages 194–203, 1994.

[PY91] Christos H. Papadimitriou and Mihalis Yannakakis. Optimization, approximation, and complexity classes. *J. Comput. Syst. Sci.*, 43(3):425–440, 1991.

[SS96] Michael Sipser and Daniel A. Spielman. Expander codes. *IEEE Transactions on Information Theory*, 42(6):1710–1722, 1996.

[Sud01] Madhu Sudan. Algorithmic introduction to coding theory (lecture notes), 2001. Available from `http://theory.csail.mit.edu/~madhu/FT01/`.

[Sze99] Mario Szegedy. Many-valued logics and holographic proofs. In *ICALP*, pages 676–686, 1999.

[Val77] Leslie G. Valiant. Graph-theoretic arguments in low-level complexity. In *MFCS*, pages 162–176, 1977.

[Vid11] Michael Viderman. A combination of testability and decodability by tensor products. *Electronic Colloquium on Computational Complexity (ECCC)*, 18:87, 2011.

A Expander-replacement for linear PCPPs with row/column access

In this appendix, we prove Lemma 4.14, restated below, which converts a linear PCPP verifier V that has row/column access into one that queries the rows and the columns according to the uniform distribution.

Lemma (4.14, restated). *Suppose that there exists a linear PCPP verifier V that has randomness complexity $r(n)$, decision complexity $d(n)$, rejection ratio $\rho(n)$, and minimal field size $F(n)$, and that has b-row/column access. Then, there exists a linear PCPP verifier V_U has randomness complexity $r_U(n) = r(n) + O(1)$, decision complexity $d_U(n) = O(b \cdot d(n) + 2^{r(n)})$, rejection ratio $\rho_U(n) = \Omega(\rho(n)/b^2)$, minimal field size $F(n)$, and $O(b)$-row/column access, and satisfies the following properties:*

- *V_U queries every row of the assignment matrix with equal probability, and always queries at least one such row.*
- *V_U does not query the columns of the assignment matrix.*
- *V_U queries every row of the proof matrix with equal probability, and same goes for the columns.*

The following definition will be useful.

Definition A.1. We say that a linear PCPP verifier V that has row/column access queries the assignment matrix rows uniformly if it satisfies the first two properties of Lemma 4.14 above.

Fix a linear verifier V as in Lemma 4.14. We construct the verifier V_U in two main steps: In the first step (Section A.2),we construct from V an intermediate verifier V_I that queries the assignment matrix rows uniformly. Then, in the second step (Section A.3), we construct V_U from V_I.

A.1 Expanders

Before diving into the proof of Lemma 4.14, we review the basics of expanders, which are used in the proof. Expanders are graphs with certain properties that make them extremely useful for many applications in theoretical computer science. Below we give a definition of expanders that suits our needs.

Definition A.2. Let $G = (V, E)$ be a d_G-regular graph. Let $E\left(S, \overline{S}\right)$ be the set of edges from a subset $S \subseteq V$ to its complement. We say that G has edge expansion h if for every $S \subseteq V$ such that $|S| \leq |V|/2$ it holds that

$$\left|E(S, \overline{S})\right| \geq h \cdot d_G \cdot |S|.$$

A useful fact is that there exist constant degree expanders over any number of vertices:

Fact A.3 ([Din07, Lemma 2.2]). *There exist $d_G \in \mathbb{N}$ and $h_G > 0$ such that there exists a polynomial-time constructable family $\{G_n\}_{n\in\mathbb{N}}$ of d_G-regular graphs G_n on n vertices that have edge expansion h_G (such graphs are called expanders).*

A.2 The intermediate verifier V_I.

In this section, we construct the intermediate verifier V_I that queries the assignment matrix uniformly.The construction is rather simple, and in order to save space, we only provide a sketch of the construction. The following proposition summarizes the properties of V_U.

Proposition A.4. *Suppose that there exists a linear PCPP verifier V that has randomness complexity $r(n)$, decision complexity $d(n)$, rejection ratio $\rho(n)$, and minimal field size $F(n)$, and that has b-row/column access. Then, there exists a linear PCPP verifier V_I that queries the assignment matrix rows uniformly, and which has randomness complexity $r_I(n) = r(n) + O(1)$, decision complexity $d_I(n) = O(b \cdot d(n))$, rejection ratio $\rho_I(n) = \Omega(\rho(n))$, minimal field size $F(n)$, and $O(b)$-row/column access.*

The basic idea that underlies the construction of V_I is to modify the proof matrix such that it contains a copy of the assignment matrix. Then, the queries of V to the assignment matrix are redirected by V_U to the copy of the assignment matrix in the proof matrix. V_U also performs additional queries to verify the consistency of the assignment matrix and its copy. The point is that the latter consistency check can be done while querying the assignment matrix uniformly.

The most straightforward way to verify the consistency of the assignment matrix and its copy in the proof matrix is to choose a random column of the assignment matrix and verify that it is consistent with the corresponding column in the copy. This would have worked, except that the randomness complexity $r(n)$ may be too small to afford choosing a random column. In order to resolve this issue, we observe that since the verifier must query every assignment coordinate with non-zero probability, it means that if the randomness complexity is small, then the decision complexity must be large. This means that we can afford to read a set of columns rather than just a single column. Therefore, we can economize on the randomness complexity by partitioning the matrix to disjoint sets of columns, and query a random set. More details follow.

Let V be a linear PCPP verifier that has randomness complexity $r(n)$, decision complexity $d(n)$, rejection ratio $\rho(n)$, and minimal field size $F(n)$, and that has b-row/column access. Fix a linear circuit φ of size n over m inputs, and let us denote by M_x and M_π the assignment matrix and proof matrix of V with respect to φ.

The proof strings of V_I. Let x be a satisfying assignment to φ. We describe the proof string π_I that convinces V that φ accepts x by describing the corresponding proof matrix M_{π_I}: The rows of matrix M_{π_I} consist of the rows of M_π, and of copies of the rows of M_x. Since the rows of M_x and M_π may be of different lengths, the rows of M_{π_I} also include dummy proof coordinates that complete the length of the shorter rows to the length of the longer rows. This concludes the description of π_I.

The action of V_I. Let $x \in \mathbb{F}^m$ be an assignment to φ, and let π_I be a proof string of V_I. We sketch the action of V_I on φ, x and π_I. Assume that the dimensions of the assignment matrix are $r_x \times c_x$. Without loss of generality, we assume that $c_x \geq r_x$ (otherwise, we modify V to work with the transpose of the assignment matrix). Let $a \overset{\text{def}}{=} \min\{b, \lfloor d(n)/r_x \rfloor\}$. Let M_x be the assignment matrix of V_I, and let us denote by M'_x the purported copy of M_x that is contained in M_{π_I}. The verifier V_I performs the following two checks, while recycling the randomness:

1. The verifier V_I partitions the assignment matrix M_x arbitrarily into disjoint sets of at most a columns, Then, V_I chooses set of columns uniformly at random, and verifies that those columns in M_x are equal to the corresponding rows of M'_x.

2. The verifier V_I emulates V on φ, and whenever V queries a row or a column of its assignment matrix or proof matrix, the verifier V_I redirects the query to the corresponding row or column of M_{π_I} When the emulation of V is finished, V_I verifies that V accepted.

The parameters of V_I. It should be clear that V_I has $O(b)$-row/column access, that it queries every row of M_x with equal probability, and that it does not queries columns of M_x. To see that V_I has decision complexity $O(b \cdot d(n))$, observe that every row and column of M_x or M_{π_I} that V_I queries is of length at most $d(n)$, and that V_I queries at most b rows and columns. This implies that V_I outputs circuits ψ with at most $O(b \cdot d(n))$ input gates, and it is easy to verify that those circuits have at most $O(b \cdot d(n))$ non-input gates.

In order to show that V_I has rejection ratio $\Omega(\rho(n))$, let x be an assignment to φ that is ε-far from $\mathbf{SAT}(\varphi)$. Let x' be the assignment to φ that can be extracted from the matrix M'_x in M_{π_I}, and let M_π be the proof matrix of V that is contained inside M_{π_I}. We consider two cases: If $\text{dist}(x', \mathbf{SAT}(\varphi)) > \varepsilon/2$, then V rejects M'_x and M_π with probability at least $\rho(n) \cdot \varepsilon/2$. Otherwise, it holds that $\text{dist}(x, x') > \varepsilon/2$, and therefore the consistency check of V must reject with probability at least $\varepsilon/2$. In both cases, the rejection probability is at least $\Omega(\rho(n)) \cdot \varepsilon$, so the rejection ratio of V_I is at least $\Omega(\rho(n))$.

Finally, we show that V_I has randomness complexity $r(n) + O(1)$. It should be clear that V_I has randomness complexity $\max\{r(n), \log(\lceil c_x/a \rceil)\}$. Now, observe that

$$\begin{aligned}
\left\lceil \frac{c_x}{a} \right\rceil &= O(\frac{c_x}{a}) \\
&= O(\frac{r_x \cdot c_x}{d(n)}) \\
&= O(\frac{m}{d(n)}).
\end{aligned}$$

Next, note that $2^{r(n)} \cdot d(n) \geq m$, since V_I must query every assignment coordinate with non-zero probability, and $2^{r(n)} \cdot d(n)$ is the maximal number of coordinates that V_I queries in all of its possible invocations. This implies that

$$\left\lceil \frac{c_x}{a} \right\rceil = O(\frac{m}{d(n)}) = O(2^{r(n)}).$$

It follows that the randomness complexity of V_I is at most

$$\max\{r(n), \log(\lceil c_x/a \rceil)\} \leq \max\left\{r(n), \log\left(O(2^{r(n)})\right)\right\} = r(n) + O(1)$$

as required.

A.3 The final verifier V_I

We turn to construct the final verifier V_U from the verifier V_I, by using the expander replacement technique of [PY91]. The crux of the construction is the following transformation, which modifies a linear verifier such that it queries every row of the proof matrix with the same probability. Of course, due to symmetry, the latter transformation can also be applied to the columns rather than the rows. We thus construct the verifier V_U by first applying the latter transformation to V_I and to the rows, and then applying the transformation again to the resulting verifier and to the columns. We turn to state the latter transformation formally.

Proposition A.5. *Suppose that there exists a linear PCPP verifier V that has randomness complexity $r(n)$, decision complexity $d(n)$, rejection ratio $\rho(n)$, minimal field size $F(n)$, and b-row/column* *access, and that queries the assignment matrix rows uniformly. Then, there exists a linear PCPP verifier V' that has randomness complexity $r(n) + O(1)$, decision complexity $O(d(n) + 2^{r(n)})$, rejection ratio $\Omega(\rho(n)/b)$, minimal field size $F(n)$, and $O(b)$-row/column access, that queries the assignment matrix rows uniformly, and that satisfies the following additional properties for every linear circuit φ: let M_π and $M_{\pi'}$ be the proof matrices of V and V' with respect to φ respectively. Then,*

- *V' queries every row of $M_{\pi'}$ with equal probability.*
- *$M_{\pi'}$ and M_π have the same number of columns, and for each column of $M_{\pi'}$ the probability that it is queried by V' is equal to the probability that the corresponding column of M_π is queries by V.*

It remains to prove Proposition A.4. Fix a linear verifier V.

The proof strings of V'. Let $x \in \mathbb{F}^m$ be a satisfying assignment for φ. We describe the proof string π' that convinces V' that x satisfies φ, by describing the corresponding proof matrix $M_{\pi'}$. Let π be the proof string that convinces V that x satisfies φ, and let M_π be the corresponding proof matrix. The proof matrix $M_{\pi'}$ consists of several copies of each row of M_π, where the number of copies of a given row of M_π is proportional to the probability that the row is queried by V.

More formally, we define the matrix $M_{\pi'}$ as follows: Let r_π be the number of rows of M_π. For every $j \in [r_\pi]$, and for every sequence of random bits ω on which V queries the j-th row of M_π, the matrix $M_{\pi'}$ contains a row that is identical to the j-th row of M_π. We index this row of $M_{\pi'}$ by (j, ω). For every $j \in [r_\pi]$, we refer to the corresponding rows of $M_{\pi'}$ as the copies of the j-th row of M_π in $M_{\pi'}$, and denote the set of indices $(j, \cdot)$ by C_j.

Remark A.6. Note that every row of M_π appears in $M_{\pi'}$ at least once. To see why, recall that the definition of row/column access requires that V queries every row and column of the proof matrix with non-zero probability.

The action of V'. We turn to describe the action of V'. Let $M_{\pi'}$ be the proof matrices and V' with respect to φ. Let M_x be the assignment matrix, which is common to V and V', with respect to φ. Informally, V' performs the steps:

1. The verifier V' tosses a sequence ω of random bits for V.
2. The verifier V' emulates V on input φ and randomness ω, and redirects the queries of V as follows:
   (a) We denote by M_π the hypothetical proof matrix that the emulation of V queries.
   (b) Whenever V queries the j-th row of M_π, the verifier V' queries the (j, ω)-th row of $M_{\pi'}$.
   (c) Queries to the assignment matrix M_x are left without a change.
   (d) Queries to columns of M_π are emulated in the natural way: Whenever V queries the i-th column of M_π, the verifier V' queries the i-th column of $M_{\pi'}$. Then, V' checks that for each coordinate j of the i-th column of M_π, all the purported copies of this coordinate in the i-th column of $M_{\pi'}$ are equal, and rejects otherwise. Finally, V' extracts from the i-th column of $M_{\pi'}$ a corresponding assignment to the i-th column of M_π in the natural way, and feeds it to the emulation of V as an answer to the query.

3. If the emulation of V rejects, then V' rejects.

4. For each row j of M_π that is queried by V on ω, the verifier V' does the following: the verifier V' chooses a random pair of copies pair of copies of the j-th row of M_π in $M_{\pi'}$, and checks that those copies are indeed equal. The latter pair of copies is chosen in a randomness-efficient manner using an expander, see details below.

5. The verifier V' accepts if the foregoing copies are equal, and rejects otherwise.

We conclude the description of V' by explaining how the pair of copies of the j-th row is chosen in Step 4 above. Recall that the copies of the j-th row of M_π in $M_{\pi'}$ are indexed by pairs of the form (j, ω') where ω' is a seqeunce of random bits, and that we denote by C_j the set of those indices. Now, the first copy in the pair is (j, ω), where ω is the sequence of random bits used in the emulation of V. In order to choose the second copy in the pair, the verifier V' constructs an expander graph (Fact A.3) whose vertices are identified with the indices in C_j. Then, the second copy in the pair is a random neighbor of (j, ω) in the expander.

It is easy to see that V' has the required randomness complexity and decision complexity, and that it satisfies the additional properties required by Proposition A.5. It remains to analyze the rejection ratio of V'.

The rejection ratio of V'

Let $x \in \mathbb{F}$ be an assignment to φ that is ε-far from $\mathbf{SAT}(\varphi)$, and let π' be a proof string of V'. Let $\rho \overset{\text{def}}{=} \rho(|\varphi|)$ be the rejection ratio of V. Let M_x and $M_{\pi'}$ be the corresponding proof matrices. We show that V' rejects $x \circ \pi'$ with probability at least $\Omega\left(\rho/b\right) \cdot \varepsilon$. Let M_π be the matrix that is constructed from $M_{\pi'}$ by plurality voting, that is, the j-th row of M_π is equal to the string that appears a maximal number of times among the rows in C_j. By assumption, V rejects $x \circ M_\pi$ with probability at least $\rho \cdot \varepsilon$.

We first introduce some notation. If (j, ω) is an index of a row of $M_{\pi'}$, we say that (j, ω) is a faulty copy if the corresponding row of $M_{\pi'}$ differs from the j-th row of M_π, which is just the plurality value. For each j, let p_j be the probability that V queries the j-th row, let α_j denote the fraction of faulty copies among the copies of the j-th row of M_π in $M_{\pi'}$. We say that V queries a faulty copy if the emulation of V tosses a sequence ω of random bits and queries a row j such that (j, ω) is faulty copy. Let us denote by Q_j the event that V queries the j-th row of M_π, by F_j the probability that it queries a *faulty copy* of the j-th row of M_π, and by $F = \bigcup F_j$ the probability that it queries any faulty copy.

Let $\alpha = \sum_j p_j \cdot \alpha_j$. We consider two cases: the case in which $\alpha \le \rho \cdot \varepsilon/2$ and the case in which $\alpha > \rho \cdot \varepsilon/2$. Intuitively, the first case means that there is a negligible fraction of faulty copies. In this case we claim that the action of V' on $M_{\pi'}$ behaves similarly to the action of V on M_π, and therefore V' rejects with sufficiently large probability. The second case means that there is a noticeable fraction of faulty copies, in which case the consistency checks make V' reject with sufficiently large probability. We turn to analyze each of the cases formally.

The case of $\alpha \le \rho \cdot \varepsilon/2$. Clearly, if V rejects $x \circ M_\pi$ but does not query any faulty copy, then V' must reject as well. We therefore get that

$$\begin{aligned} \Pr\left[V' \text{ rejects}\right] &\ge \Pr\left[V \text{ rejects and } \neg F\right] \\ &\ge \Pr\left[V \text{ rejects}\right] - \Pr\left[F\right] \\ &\ge \rho \cdot \varepsilon - \Pr\left[F\right]. \end{aligned}$$

We now show that $\Pr[F] \le \alpha \le \rho \cdot \varepsilon/2$, and this will imply that V' rejects with probability at least $\rho \cdot \varepsilon/2$:

$$\begin{aligned}
\Pr[F] &\le \sum_j \Pr[F_j] \\
&= \sum_j \Pr[F_j \text{ and } Q_j] && (3)\\
&= \sum_j \Pr[F_j | Q_j] \cdot \Pr[Q_j] \\
&= \sum_j \Pr[F_j | Q_j] \cdot p_j \\
\text{(see below)} \quad &= \sum_j \alpha_j \cdot p_j && (4)\\
&= \alpha,
\end{aligned}$$

where Equality 4 follows by observing that, conditioned on the event that V queries the j-th row, the copy of the j-th row that it queries is uniform over the all the copies.

The case of $\alpha > \rho \cdot \varepsilon/2$. We begin the analysis of this case by proving, for every j, a lower bound on the probability that V' rejects conditioned on the event that V queried the j-th row. Fix an index j, and let us condition on the event that V queries the j-th row (i.e., Q_j). Recall that V' checks the consistency of a pair (j, ω), (j, ω') of copies of the j-th row. By definition, if $M_{\pi'}$ assigns different values to the rows in (j, ω) and (j, ω'), then V' rejects.

Next, recall that (j, ω') is a uniformly distributed neighbor of (j, ω) in an expander over the set of copies C_j. Moreover, as we observed at the end of the previous case, the index (j, ω) is uniformly distributed over C_j. Therefore, the pair of copies (j, ω), (j, ω') is distributed like a uniformly distributed edge of the expander. This implies that the probability that V' rejects is lower bounded by the probability that a random edge in the expander connects two copies that assign different values to the j-th row of M_π. To lower bound the latter probability, we use the edge expansion of the expander. Specifically, if we denote by d_G and h_G the (constant) degree and edge expansion of the expanders from Fact A.3, we show in Claim A.7 below that the latter probability is at least $\frac{1}{2} \cdot h_G \cdot d_G \cdot \alpha_j$. This implies that

$$\Pr\left[V' \text{ rejects} | Q_j\right] \ge \frac{1}{2} \cdot h_G \cdot d_G \cdot \alpha_j .$$

It follows that

$$\begin{aligned}
\sum_j \Pr\left[V' \text{ rejects} | Q_j\right] \cdot \Pr[Q_j] &\ge \frac{1}{2} \cdot h_G \cdot d_G \cdot \sum_j \alpha_j \cdot p_j \\
&= \frac{1}{2} \cdot h_G \cdot d_G \cdot \alpha \\
&\ge h_G \cdot d_G \cdot \rho \cdot \varepsilon/4.
\end{aligned}$$

Finally, we claim that

$$\sum_j \Pr\left[V' \text{ rejects} | Q_j\right] \cdot \Pr[Q_j] \le b \cdot \Pr\left[V' \text{ rejects}\right],$$

and this would imply that V' rejects with probability at least $\Omega(\rho/b) \cdot \varepsilon$, as required. In order to establish the latter inequality, we note that every possible sequence of random bits of V' is counted in at most b terms $\Pr[V' \text{ rejects}|Q_j] \cdot \Pr[Q_j]$. To see it, observe that a sequence of coin tosses for V' is counted in a term $\Pr[V' \text{ rejects}|Q_j] \cdot \Pr[Q_j]$ only if on thissequence, the emulation of V queries the j-th row. Since the emulation of V always queries at most b rows, the required bound follows.

Claim A.7. *Fix a row j. The probability that a random edge $((j,\omega),(j,\omega'))$ in the expander over C_j satisfies $(M_{\pi'})_{(j,\omega)} \neq (M_{\pi'})_{(j,\omega')}$ is at least $\frac{1}{2} \cdot h_G \cdot d_G \cdot \alpha_j$.*

Proof. Let us partition the C_j to sets $C_{j,1}, \ldots, C_{j,t}$ such that two copies (j,ω), (j,ω') are in the same set $C_{j,i}$ if and only if $(M_{\pi'})_{(j,\omega)} = (M_{\pi'})_{(j,\omega')}$. In other words, each set $C_{j,i}$ corresponds to a single value that is assigned to M_π by some row of $M_{\pi'}$. Without loss of generality, assume that the larget set is C_1, which means that C_1 is the set of non-faulty copies. Let us denote by $\alpha_{j,i}$ the density of the set $C_{j,i}$ in C_j. Observe that

$$\alpha_j = \sum_{i=2}^{t} \alpha_{j,i},$$

and that $\alpha_{j,i} \leq \frac{1}{2}$ for every $i \geq 2$ (since C_1 is the larget set).

The probability that we wish to lower bound is the fraction of edges whose endpoints lie in two distinct sets. Now, by the edge expansion, the fraction of edges leaving each set $C_{j,i}$ for $i \geq 2$ is at least $h_G \cdot d_G \cdot \alpha_{j,i}$ (here we used the fact that $\alpha_{j,i} \leq \frac{1}{2}$). Thus, the latter probability is lower bounded by

$$\frac{1}{2} \cdot \sum_{i=2}^{t} h_G \cdot d_G \cdot \alpha_{j,i} = \frac{1}{2} \cdot h_G \cdot d_G \cdot \alpha_j,$$

as required (the factor of $\frac{1}{2}$ is in order to avoid double-counting of edges). ■