Some Improvements for the PIOP for ZeroCheck

Angus Gruen
Polygon Zero

Abstract

Most multivariate proof systems require, at some point, an algebraic check against the rows of the trace. One popular protocol for this is known as zerocheck which is a sumcheck based protocol which proves a constraint function is zero over the n-dimensional Boolean hypercube. One of the drawbacks of running zerocheck over a small field, is that it usually involves a large number of evaluations of the constraint polynomial over a cryptographically large extension field $\mathbb{G}$.

We describe several improvements to the zerocheck protocol over a small field $\mathbb{F}$ which both reduce the number of constraint evaluations the prover needs to perform and shifts some computation from the extension field back into the base field $\mathbb{F}$. Specifically, for a table of length 2^n, integer parameter $1 \leq k \leq n$ and constraint function C of degree d with evaluation costs $C_{\mathbb{F}}, C_{\mathbb{G}}$ we show the protocol can be performed with prover cost roughly

$$2^n \left(1 + \frac{C_{\mathbb{G}}}{2^k C_{\mathbb{F}}}\right)(d-1)C_{\mathbb{F}}.$$

To check the proof, the verifier needs to perform a single interpolation of degree $2^k d$ and n interpolations of degree d. Thus varying k allows for a tradeoff between prover and verifier cost.

1 Introduction

This paper is concerned with optimizing a particular Polynomial Interactive Oracle Proof (PIOP).

Generically, PIOPs allow a prover ($\mathbb{P}$) to convince a verifier ($\mathbb{V}$) of the veracity of some claim. The “Polynomial” keyword implies that this protocol will consist of a set of rounds where in each round $\mathbb{P}$ will send $\mathbb{V}$ a polynomial (or a commitment to a polynomial) and $\mathbb{V}$ will reply with a vector of random field elements. At the end of the protocol, $\mathbb{V}$ can choose to either accept or reject the claim by querying the polynomials at a small number of points and testing a small set of polynomial relations.

The sumcheck protocol [Lun+92] is a simple PIOP which allows $\mathbb{P}$ to prove the summation of a multivariate function f over the Boolean hypercube

$$v = \sum_{\mathbf{X} \in H_n} f(\mathbf{X}).$$

The elegance of sumcheck is that it allows $\mathbb{P}$ to prove that v is the sum with $\mathbb{V}$'s workload only *linear* in n.

Thanks to its simplicity and versatility and, in part, due to a lack of alternatives, the sumcheck protocol is one of the key algorithms behind multivariate SNARKS.

We will be focused on a specific and common use case of sumcheck known as the zerocheck [Che+22]. The zerocheck protocol aims to prove that

$$f(\mathbf{X}) = 0 \quad \forall\, \mathbf{X} \in H_n.$$

As observed in [Set20; Che+22] it is possible to reduce the zerocheck protocol to a sumcheck protocol by taking an inner product with a suitable function over H_n. We will discuss this in mor detail in Section 2.3. The purpose of this paper is to describe some improvements to the zerocheck protocol which reduce how much work $\mathbb{P}$ needs to perform. As a rough guide

- Section 2 introduces notation and gives explicit descriptions of the sumcheck and zerocheck protocols. We also discuss the cost model we will be using to look for improvements.
- Section 3 describes a couple of minor protocol tweaks which improve the efficiency of zerocheck in all cases with no downside. We stress that there is no reason to not apply these tweaks in a performance oriented implementation of zerocheck.
- Section 4 describes how $\mathbb{P}$ can leverage a Gröbner basis decomposition of the constraint function to further reduce their workload. Note that these improvements make no change to the protocol. It is simply a method which allows $\mathbb{P}$ to compute the required information with less effort.
- Section 5 discusses improvements related to skipping rounds in the protocol. These reduce the work of $\mathbb{P}$ but do have a minor trade-off of increasing $\mathbb{V}$'s work somewhat (Though it remains linear in n).
- Section 6 gives an updated zerocheck protocol taking into account the improvements from previous sections.

Note that Sections 3, 4 and 5 are essentially independent and so are presented here roughly in order of the complexity required to implement them. The ideas contained within should also generalize to other sumcheck/zerocheck like protocols though we do not discuss such generalizations here.

1.1 Related Work

The recent paper [Tha23] looked at a similar problem but took an alternative path. Broadly the ideas in [Tha23] focus on reducing the cost of each evaluation of the constraint polynomial. In contrast, the improvements contained in this manuscript are focused on reducing the number of evaluations needed. It goes without saying that these works can and should be applied in conjunction.

2 Preliminaries

2.1 Algebraic Preliminaries

Let $\mathbb{F} = \mathbb{F}_p$ denote a fixed finite field and $\mathbb{G} = \mathbb{F}_{p^m}$ an extension field of cryptographic size[1]. Given a subset $D \subset \mathbb{F}_p$ and a function $f : D \to \mathbb{F}_p$, we use[2] $\hat{f}$ and $[[f]]$ to denote the polynomial interpolate of f and an oracle of f respectively.

The interpolant $\hat{f} \in \mathbb{F}_p[X]$ is the unique polynomial of degree $\leq |D| - 1$ such that $\hat{f}|_D = f$. An oracle $[[f]]$ for f is a commitment[3] which, given an element a, allows us to produce both $f(a)$ and a proof that this is the correct value.

Throughout most of this document we will be dealing with multivariate polynomials $\hat{f} \in \mathbb{F}_p[X_0, \cdots, X_{n-1}]$ which can make notation a little messy. To help with this we will always use a bold font $\mathbf{X}$ to denote vectors with capital letters denoting indeterminates and lower case letters evaluation points. There are also a collection of different ways in which we can evaluate $\hat{f}$. We can evaluate it on n single elements $\hat{f}(X_0, \cdots, X_{n-1})$, a vector $\mathbf{X}$ of length n, $\hat{f}(\mathbf{X})$, or some combination of vectors whose combined length is n, $\hat{f}(\mathbf{X}_0, \cdots, \mathbf{X}_i)$. We will use these three pictures interchangeably.

A monomial is a simple product of the form $\mathbf{X}^{\mathbf{i}} = X_0^{i_0} \cdots X_{n-1}^{i_{n-1}}$. These form a basis for the vector space of multivariate functions, meaning every such function admits a unique decomposition

$$\hat{f}(\mathbf{X}) = \sum_{\mathbf{i}} f_{\mathbf{i}} \, \mathbf{X}^{\mathbf{i}}$$

where the sum is over all $\mathbf{i}$ with $f_{\mathbf{i}} \neq 0$. We define the X_j-degree of $\mathbf{X}^{\mathbf{i}}$ to be i_j and the total degree to be $i_0 + \cdots + i_{n-1}$. The X_j and total degrees of $\hat{f}$ are the maximal X_j and total degrees of any monomial appearing in its decomposition.

[1]In particular this usually means $p^m > 2^{128}$.

[2]In cases where f is already a polynomial or it is unambiguous we occasionally drop the $\hat{}$.

[3]We ignore the specific choice of commitment here.

A special case occurs when all X_i-degrees are ≤ 1, in which case the polynomial is called multilinear.

Remark 2.1. A multilinear polynomial is uniquely determined by its evaluations over the boolean hypercube $H_n = \{0,1\}^n$.

An important class of polynomials are the interpolates of δ functions. Given a domain D, the delta function on D is the unique function $D \times D \to \{0,1\} \subset \mathbb{F}$ given by

$$\delta_D(X,Y) = \begin{cases} 1 & X = Y \\ 0 & X \neq Y \end{cases}.$$

This function is interpolated by the polynomial

$$\hat{\delta}_D(X,Y) = \sum_{d \in D} \prod_{\substack{i \in D \\ i \neq d}} \frac{(X-i)(Y-i)}{(d-i)^2}.$$

This can now be used to define the interpolate for any function on D as

$$\hat{f}(X) = \sum_{d \in D} f(d) \hat{\delta}_D(X,d).$$

These $\hat{\delta}$ also naturally extend to multivariate product domains. If $D = D_0 \times \cdots \times D_{n-1}$ then

$$\hat{\delta}_D(\mathbf{X}, \mathbf{Y}) = \prod_{i=0}^{n-1} \hat{\delta}_{D_i}(X_i, Y_i).$$

These can be used to interpolate multivariate polynomials over product domains in an identical way to the univariate case.

The reason why polynomials are so widely used in proof systems essentially comes down to the following lemma:

Lemma 2.2 (Schwartz-Zippel)**.** *Let $\mathbb{F}$ be a field and $f \in \mathbb{F}[X_0, \cdots, X_{n-1}]$ be a polynomial of total degree d. If we randomly sample n elements $r_0, \cdots, r_{n-1}$ from $\mathbb{F}$ then*

$$\Pr[f(r_0, \cdots, r_{n-1}) = 0] \leq \frac{d}{|\mathbb{F}|}$$

Note that this can be equivalently stated as, given 2 different polynomials f, g of degree d, the probability that $f(r_0, \cdots, r_{n-1}) = g(r_0, \cdots, r_{n-1})$ at a randomly chosen point is $\leq \frac{d}{|\mathbb{F}|}$. Translating, this means that if we work over a big enough field and evaluate 2 low degree polynomials at the same random point. Then, if the evaluations agree, we can conclude with high probability that our polynomials are equal. This lemma is the key cog in all the security analysis we will do in this paper.

2.2 SumCheck

Before getting to zerocheck, we first give a brief overview of the well known sumcheck protocol [Lun+92; Tha22]. Assume a prover $\mathbb{P}$ has committed to a multivariate function $\hat{f}$ and wishes to prove that

$$v = v_0 = \sum_{\mathbf{x} \in H_n} \hat{f}(\mathbf{x}).$$

We assume that from context[4] that the verifier $\mathbb{V}$ knows one global piece of information, namely a positive integer d which satisfies

$$\max_i(\deg_{x_i} \hat{f}) \leq d.$$

The protocol works by iteratively reducing $\mathbb{P}$'s claim to a summation over a smaller hypercube.

1. Setup. $\mathbb{P}$ sends $\mathbb{V}$ a value v and a commitment $[[f]]$.
2. Rounds $0 \cdots n-1$. At the start of the i'th round, $\mathbb{P}$ wishes to prove

   $$v_i = \sum_{\mathbf{x} \in H_{n-i}} \hat{f}(\mathbf{r}, \mathbf{x})$$

   where $\mathbf{r} = (r_0, \cdots, r_{i-1}) \in \mathbb{G}^i$ is a vector of random elements with r_j sent by $\mathbb{V}$ in the j'th step.

   (a) $\mathbb{P}$ computes

   $$v_{i+1}(X) = \sum_{\mathbf{x} \in H_{n-i-1}} \hat{f}(\mathbf{r}, X, \mathbf{x})$$

   for[5] $X = 0, \cdots, d$ and send the values to $\mathbb{V}$.

   (b) $\mathbb{V}$ check that

   $$v_i = v_{i+1}(0) + v_{i+1}(1)$$

   rejects if this fails and otherwise responds by by sending the next random variable $r_i \in \mathbb{G}$.

   (c) Both $\mathbb{P}$ and $\mathbb{V}$ interpolate $v_{i+1}(X)$ to compute $v_{i+1} = v_{i+1}(r_i)$
3. Round n. $\mathbb{V}$ queries the commitment $[[f]]$ at $\mathbf{r} = (r_0, \cdots, r_{n-1})$ and accepts if any only if

   $$v_n = f(\mathbf{r}).$$

As desired, $\mathbb{V}$'s work consists of only an $O(n)$ number of interpolations of degree d.

For a discussion of the completeness and soundness of this protocol, see A.1.

[4]The standard case is that $\mathbb{P}$ has a collection of multilinear functions $f_1, \cdots, f_l$ and $\hat{f} = Q(f_1, \ldots, f_L)$ where Q is public polynomial of small degree d

[5]As $v_{i+1}(X)$ has degree $\leq d$ this determines v_{i+1}. Depending on the situation, a different set of $d+1$ points might be preferable.

2.3 ZeroCheck

Let's turn to the situation that the remainder of the paper will focus on. Assume $\mathbb{P}$ starts with a table T of size[6] $2^n \times l$ and a constraint polynomial C with known total degree $\leq d$ which vanishes on every row of the table

$$C(T_{i,0}, \cdots, T_{i,l-1}) = 0. \tag{1}$$

$\mathbb{P}$ wishes to prove to $\mathbb{V}$ that C vanishes on every row in a way such that $\mathbb{V}$ only performs $O(n)$ work (so the naive method of sending both T and C is out).

There is a standard method to reformulate this problem in the language of polynomials. Reinterpret each column as evaluations of a multilinear polynomial ω_i on the hypercube H_n and define

$$C(\mathbf{X}) = C(\omega_0(\mathbf{X}), \cdots, \omega_{l-1}(\mathbf{X})).$$

Then $\mathbb{P}$ commits to the polynomials $\omega_i(\mathbf{X}) \in \mathbb{F}[X_0, \cdots, X_{n-1}]$, $C(\boldsymbol{\omega}) \in \mathbb{F}[\omega_0, \cdots, \omega_{l-1}]$ and then needs to prove that $C(\mathbf{x}) = 0$ for all $\mathbf{x} \in H_n$. Following [Set20; Che+22], this can be reduced to a sumcheck using

$$\hat{\delta}_{H_n}(X, Y) = \hat{\delta}_n(X, Y) = \prod_{i=1}^{n} (X_i Y_i + (1 - X_i)(1 - Y_i)).$$

Given a random $\boldsymbol{\alpha} \in \mathbb{G}^n$ define the function

$$\hat{F}(\boldsymbol{\alpha}) = \sum_{\mathbf{x} \in H_n} \hat{\delta}_n(\boldsymbol{\alpha}, \mathbf{x})\ C(\mathbf{x}).$$

Then with high probability $\hat{F}(\boldsymbol{\alpha}) = 0$ if and only if $C(\mathbf{X})$ vanishes on H_n.

Thus $\mathbb{P}$ can apply the sumcheck protocol we just introduced to prove that $\hat{F}(\boldsymbol{\alpha}) = 0$ for an $\boldsymbol{\alpha}$ supplied by the verifier. For the technical details regarding the completeness and soundness see A.2.

2.3.1 Prover Costs

The only work outside of the usual sumcheck algorithm that $\mathbb{V}$ needs to perform is in sampling and sending α which is clearly also linear in n.

Thus let us turn to studying the cost to $\mathbb{P}$ of this protocol. This will be our main guiding parameter which we will seek to minimise throughout the rest of the paper.

The majority of the cost for $\mathbb{P}$ to run the zerocheck protocol can be split into 3 distinct parts.

[6]For simplicity we will assume for now that the height of the table is a power of 2. Note that we will be able to work with a weaker assumption for our final protocol.

- Multilinear Interpolation. Given the evaluations of $\omega_i(\mathbf{X})$ over H_n, in the i'th round $\mathbb{P}$ needs to compute $\omega_i(\mathbf{r}, X, \mathbf{x})$ with $\mathbf{r} = (r_0, \cdots, r_{i-1}) \in \mathbb{G}^i$ supplied by the verifier, $X \in \{0, \cdots, d+1\}$ and $\mathbf{x} \in H_{n-i-1}$.

- Constraint Evaluation. Given the evaluations of $\omega_i(\mathbf{r}, X, \mathbf{x})$ for all $i \in \{0, \cdots, l-1\}$, compute

$$C(\mathbf{r}, X, \mathbf{x}) = C(\omega_0(\mathbf{r}, X, \mathbf{x}), \cdots, \omega_{l-1}(\mathbf{r}, X, \mathbf{x}))$$

- Auxiliary Work. The remainder of the computations $\mathbb{P}$ needs to do. This includes multiplying constraints by the δ_n factor and summing up the results over $\mathbf{x} \in H_{n-i-1}$. There is also a small polynomial interpolation of $v_{i+1}(X)$ to compute $v_{i+1}(r_i)$ and potentially some other minor things.

We assume henceforth that the cost to $\mathbb{P}$ is dominated by the second bullet, constraint evaluation. This is often the case when T is the trace of a proof system which is the main use for zerocheck. Thus our goal should be to minimise the number of constraint evaluations we need to do.

That being said there is one more important consideration to make. Arithmetic operations in $\mathbb{F}$ will be much cheaper than those in $\mathbb{G}$. Hence we use $C_{\mathbb{F}}$ ($C_{\mathbb{G}}$) to denote the cost of evaluating C when all entries are in $\mathbb{F}$ ($C_{\mathbb{G}}$). The first case occurs only in round 0 before $\mathbb{V}$ has sent r_0.

In round i, as $v_{i+1}(X)$ has degree $d+1$, $\mathbb{P}$ needs to compute and send $v_{i+1}(X)$ at $(d+2)$ locations each of which requires 2^{n-i-1} evaluations of C (As $v_{i+1}(X)$ is a sum over H_{n-i-1}). Hence the number of constraint evaluations needed will be

$$(d+2)\left(2^{n-1}C_{\mathbb{F}} + \sum_{i=1}^{n-1} 2^{n-i-1}C_{\mathbb{F}}\right) \sim (d+2)2^{n-1}\big(C_{\mathbb{F}} + C_{\mathbb{G}}\big)$$

3 Minor Protocol Tweaks

We start with a couple of small changes to the protocol described above which provide an immediate improvement.

3.1 Sending less data

The simplest tweak involves simply shifting a small amount of extra work onto the verifier. Recall that in round i, $\mathbb{P}$ sends $\mathbb{V}$ the values $v_{i+1}(X)$ for $X = 0, \cdots, d+1$ after which $\mathbb{V}$ checks that $v_i = v_{i+1}(0) + v_{i+1}(1)$. Instead of doing this, $\mathbb{P}$ can simply not send $v_{i+1}(1)$ and let $\mathbb{V}$ fill in this value via[7]

$$v_{i+1}(1) = v_i - v_{i+1}(0).$$

[7]Alternatively, this can equivalently be used by $\mathbb{P}$ to compute $v_{i+1}(1)$ from $v_{i+1}(0)$

Similarly, in the zeroth round, an honest prover will always send $v_1(0) = v_1(1) = 0$ meaning this doesn't actually require any evaluations.

This reduces the cost to

$$d2^{n-1}C_{\mathbb{F}} + (d+1)2^{n-1}C_{\mathbb{G}}$$

3.2 Slight modification of the Protocol

Next, observe that the instance of sumcheck that we perform is more structured than the standard sumcheck. In particular it can be viewed as a weighted sumcheck as discussed in [Mei13; BCS21]. This interpretation allows us to make a minor change which improves both the efficiency and security of the protocol.

Fast forwarding to the i'th round of the protocol the verifier has, over the previous i rounds, sent $\mathbf{r} = (r_0, \cdots, r_{i-1})$. Currently the protocol asks $\mathbb{P}$ to send the polynomial

$$v_{i+1}(X) = \sum_{\mathbf{x} \in H_{n-i-1}} \hat{\delta}_n(\boldsymbol{\alpha}, (\mathbf{r}, X, \mathbf{x}))C(\mathbf{r}, X, \mathbf{x}).$$

But, instead, consider what happens if $\mathbb{P}$ instead sends evaluations of

$$v'_{i+1}(X) = \sum_{\mathbf{x} \in H_{n-i-1}} \hat{\delta}_{n-i-1}((\alpha_{i+1}, \cdots, \alpha_{n-1}), \mathbf{x})C(\mathbf{r}, X, \mathbf{x})$$

These two functions are related via

$$v_i(X) = v'_i(X)\hat{\delta}_{i+1}((\alpha_0, \cdots, \alpha_i), (\mathbf{r}, X))$$

and this δ_{i+1} factor is entirely known by $\mathbb{V}$ as they know both $\boldsymbol{\alpha}$ and $\mathbf{r}$.

The improvement comes from the fact that $\deg v'_i(X) = \deg_X v_i(X) - 1$ meaning $\mathbb{P}$ needs to evaluate v'_i on one fewer point.

The main protocol change comes from the inter-round checks. Instead of the constraint

$$v_i = v_{i+1}(0) + v_{i+1}(1)$$

we have the modified constraint

$$(1 - \alpha_i)v'_{i+1}(0) + \alpha_i v'_{i+1}(1) = v'_i.$$

Hence we can combine this with the previous observation so $\mathbb{P}$ can evaluate and send even less data.

As a side benefit, $\hat{\delta}_{n-i-1}$ is also cheaper[8] to compute than $\hat{\delta}_n$.

With these two improvements combined the cost reduces to

$$\sim 2^{n-1}(d-1)C_{\mathbb{F}} + 2^{n-1}dC_{\mathbb{G}}$$

In the common $d = 3$ case these simple improvements already reduce the cost by a factor of $\frac{5}{3}$.

[8]You can recursively construct a table of evaluations of $\hat{\delta}_{n-i}$ from a table of evaluations of $\hat{\delta}_{n-i+1}$.

3.3 Improved ZeroCheck Protocol:

Let's quickly run through what this new and improved zerocheck protocol looks like. We postpone a discussion of its soundness to section A.3.

1. Setup and Round 0

   (a) $\mathbb{P}$ commits to the constraint function C of degree d which holds over the rows of table of length 2^n.

   (b) $\mathbb{V}$ samples[9] $\boldsymbol{\alpha} \in G^n$ and sends to $\mathbb{P}$.

   (c) $\mathbb{P}$ computes

   $$v_1'(X) = \sum_{\mathbf{x} \in H_{n-1}} \hat{\delta}_{n-1}((\alpha_1, \cdots, \alpha_{n-1}), \mathbf{x}) C(X, \mathbf{x})$$

   for $X \in 2, \cdots, d$ and sends to $\mathbb{V}$.

   (d) $\mathbb{V}$ samples $r_0 \in G$ and sends to $\mathbb{P}$. Additionally, $\mathbb{V}$ interpolates $v_1'(X)$ from the data sent by $\mathbb{P}$ along with $v_1'(0) = v_1'(1) = 0$ and computes $v_1' = v_1'(r_0)$.

2. Rounds $1 \cdots (n-1)$. At the start of the i'th round, $\mathbb{V}$ has just computed $v_i' = v_i'(r_{i-1})$ and sent r_{i-1} to the $\mathbb{P}$ who has added it to $\mathbf{r} = (r_0, \cdots, r_{i-2})$.

   (a) $\mathbb{P}$ computes

   $$v_{i+1}'(X) = \sum_{\mathbf{x} \in H_{n-i-1}} \hat{\delta}_{n-i}((\alpha_{i+1}, \cdots, \alpha_{n-1}), \mathbf{x}) C(\mathbf{r}, X, \mathbf{x})$$

   for $X \in 0, 2, 3, \cdots, d$ and sends to $\mathbb{V}$.

   (b) $\mathbb{V}$ samples $r_i \in G$ and sends to $\mathbb{P}$. Then, $\mathbb{V}$ interpolates $v_{i+1}'(X)$ from the data sent by $\mathbb{P}$ along with

   $$v_{i+1}'(1) = \frac{1}{\alpha_i} \left(v_i' - (1 - \alpha_i) v_{i+1}'(0) \right)$$

   and computes $v_{i+1}' = v_{i+1}'(r_i)$.

3. Final Check

   (a) After r_n has been sent, $\mathbb{P}$ computes $C(\mathbf{r})$ and sends it to $\mathbb{V}$, along with a commitment proof.

   (b) $\mathbb{V}$ accepts if and only if $C(\mathbf{r}) = v_n'$.

[9]We never actually use α_0 so $\mathbb{V}$ can send one element less than usual too.

4 Algebraic Improvements

With some care, it is also possible for an honest prover to make use of computations performed in earlier rounds to avoid some work in later rounds.

4.1 Algebraic Interlude

For an honest prover, $C(x_0, \cdots, x_{n-1})$ does actually vanish on H_n and so admits a decomposition of the form

$$C(x_0, \cdots, x_{n-1}) = \sum_{i=0}^{n-1} x_i(x_i - 1)Q_i(x_0, \cdots, x_{n-1}).$$

This follows as a corollary to the Hilberts Nullstellensatz as the ideal $\langle x_i(x_i - 1)\rangle$ is radical with vanishing set equal to H_n. As stated so far, these Q_i are not unique[10] but we can produce a decomosition with nice properties using polynomial long division with remainder (//). Define $R_n = C$ and for $i = 0, \cdots, n-1$,

$$Q_i, R_i = R_{i+1}//x_i(x_i - 1).$$

Then R_0 is exactly the multilinear extension of the evaluations of C on H_n which is 0 by assumption[11].

If we study the degrees of the Q_i, R_i produced by this process, we find that

$$\deg_{x_i}(Q_j) \leq \begin{cases} \deg_{x_i}(C) & j > i \\ \deg_{x_i}(C) - 2 & i = j \\ 1 & j < i. \end{cases} \tag{2}$$

In particular we observe that if given a random vector $\mathbf{r} = (r_0, \cdots, r_i)$ then $Q_j(\mathbf{r}, \mathbf{X})$ is multilinear in $\mathbf{X}$ for all $j \leq i$. Additionally, defining the sum

$$\overline{Q}_i(\mathbf{r}, \mathbf{X}) = \sum_{j=0}^{i} r_j(r_j - 1)Q_j(\mathbf{r}, \mathbf{X}),$$

this is precisely the unique multilinear polynomial equal to $C(\mathbf{r}, \mathbf{X})$ for all $\mathbf{X} \in H^{n-i}$. We immediately see that for indeterminate X and $\mathbf{x} \in H^{n-i-1}$

$$C(\mathbf{r}, X, \mathbf{x}) - \overline{Q}_i(\mathbf{r}, X, \mathbf{x}) = X(X-1)Q_{i+1}(\mathbf{r}, X, \mathbf{x}).$$

Hence in each round of the protocol we only introduce 1 extra Q whose evaluations can be computed from C and $\overline{Q}$. Additionally, when we come to computing $\overline{Q}_i$ we will have already fixed the first i components by $r_0, \cdots, r_{i-1}$ and so $\overline{Q}_i(\mathbf{r}, X, \mathbf{X})$ will be degree $d - 2$ in X and multilinear in $\mathbf{X}$. Thus it is entirely determined by its evaluations over $[0, \cdots, d] \times H_{n-i-1}$.

[10] A counter example to uniqueness is $C = x_0(x_0 - 1)x_1(x_1 - 1)$ were both $Q_0 = 0, Q_1 = x_0(x_0 - 1)$ and $Q_0 = x_1(x_1 - 1), Q_1 = 0$ are valid decompositions.

[11] If C did not vanish on H_n you could still compute this decomposition but R_0 would be non 0

4.2 Application

Let's see how the prover, $\mathbb{P}$, can make use of this[12] to reduce how many times they need to compute C.

1. Round 0. In the initial round, whenever $\mathbb{P}$ computes $C(X, \mathbf{x})$, they save $\frac{C(X,\mathbf{x})}{X(X-1)}$ in a table. As

   $$C(X, \mathbf{x}) = X(X-1)Q_0(X, \mathbf{x}) \; \forall \mathbf{x} \in H_{n-1}.$$

   after all computations, $\mathbb{P}$ will have a table consisting of $Q_0(X, \mathbf{x})$ for $X = 2, \cdots, d$ and $\mathbf{x} \in H_{n-1}$.

   Then when $\mathbb{V}$ sends r_0, for each $\mathbf{x} \in H_{n-1}$, $\mathbb{P}$ can interpolate $Q_0(X, \mathbf{x})$ to compute and store

   $$\overline{Q}_0(r_0, \mathbf{x}) = r_0(r_0 - 1)Q_0(r_0, \mathbf{x}).$$

2. Round 1. As $\overline{Q}_0(r_0, \mathbf{x}) = C(r_0, \mathbf{x})$ for all $\mathbf{x} \in H_{n-1}$, we can immediately compute $v_2(0)$ without needing any evaluations of C. Essentially we have replaced each evaluation of C with an interpolation[13] of degree $d - 2$ which will usually be much faster.

   Moreover, as $\overline{Q}_0$ is linear in x_1, our knowledge of $\overline{Q}_0(r_0, 0, \mathbf{x}), \overline{Q}_0(r_0, 1, \mathbf{x})$ determines $\overline{Q}_0(r_0, X, \mathbf{x})$ for all X. Then, as for all $\mathbf{x} \in H_{n-2}$,

   $$C(r_0, X, \mathbf{x}) = r_0(r_0 - 1)\overline{Q}_0(r_0, X, \mathbf{x}) + X(X-1)Q_1(r_0, X, \mathbf{x}),$$

   when $\mathbb{P}$ computes $C(r_0, X, \mathbf{x})$ they can also save $Q_1(r_0, X, \mathbf{x})$.

   Now, when $\mathbb{V}$ sends r_1, for each $\mathbf{x} \in H_{n-2}$, $\mathbb{P}$ interpolates[14] $\overline{Q}_0(r_0, X, \mathbf{x})$ and $Q_1(r_0, X, \mathbf{x})$ to compute and store

   $$\overline{Q}_1(r_0, r_1, \mathbf{x}) = r_0(r_0 - 1)Q_0(r_0, r_1, \mathbf{x}) + r_1(r_1 - 1)Q_1(r_0, r_1, \mathbf{x}).$$

3. Round i. At the end of the last round, $\mathbb{P}$ has received the randomness r_{i-1} and computed and stored

   $$\overline{Q}_{i-1}(\mathbf{r}, \mathbf{x}) = \sum_{j=0}^{i-1} r_j(r_j - 1)Q_j(\mathbf{r}, \mathbf{x})$$

[12] The is also a slightly simpler approach which the prover can use where they simply save all the $C(\mathbf{r}, X, \mathbf{x})$ in the i'th round and then interpolate to get $C(\mathbf{r}, r_i, \mathbf{x})$ which can be used in the $i+1$'th round. The downside compared to the approach presented here is that all interpolations will have degree d not $d-2$. This might turn out to be more efficient though as you avoid some auxiliary field multiplications and adds.

[13] Needed to compute $\overline{Q}_0(r_0, \mathbf{x})$.

[14] Note again the first interpolation is degree 1, the second is degree $d-2$.

for all $\mathbf{x} \in H_{n-i}$. Again, $\mathbb{P}$ can immediately compute $v_{i+1}(0)$ as $\overline{Q}_{i-1}(\mathbf{r}, 0, \mathbf{x}) = C(\mathbf{r}, 0, \mathbf{x})$ for all $\mathbf{x} \in H_{n-i-1}$.

Next, similarly to before, as $\mathbb{P}$ computes $v_i'(X)$ for $X = 2, \cdots, d$, they can simultaneously determine $\overline{Q}_{i-1}(\mathbf{r}, X, \mathbf{x})$ and use this to compute and save $Q_i(\mathbf{r}, X, \mathbf{x})$.

Then, when $\mathbb{V}$ sends r_i, for each $\mathbf{x} \in H_{n-i-1}$, $\mathbb{P}$ interpolates $\overline{Q}_{i-1}(\mathbf{r}, X, \mathbf{x})$ and $Q_i(\mathbf{r}, X, \mathbf{x})$ to compute and store

$$\overline{Q}_i(\mathbf{r}, \mathbf{x}) = \overline{Q}_{i-1}(\mathbf{r}, \mathbf{x}) + r_i(r_i - 1)Q_i(\mathbf{r}, \mathbf{x})$$

for all $\mathbf{x} \in H_{n-i-1}$.

Whilst there are some additional costs generated here[15], this will reduce the number of constraint evaluations will to

$$2^{n-1}(d-1)(C_{\mathbb{F}} + C_{\mathbb{G}}).$$

It also has no effect on soundness or completeness no changes are made to what is sent to $\mathbb{V}$.

5 Skipping rounds

In the protocol, the initial round is particularly special as it is the only round in which we are able to evaluate C over the base field $\mathbb{F}$.

Motivated by this it seems natural to wonder if it is possible to perform more work in this first round to save ourselves work in future rounds. The most natural option[16] would be, in the first round, to compute the multivariate polynomial

$$v(X_0, \cdots, X_{k-1}) = \sum_{\mathbf{x} \in H_{n-k}} \hat{\delta}_{n-k}((\alpha_k, \cdots, \alpha_{n-1}), \mathbf{x}) C(X_0, \cdots, X_{k-1}, \mathbf{x}).$$

This polynomial could then either be sent directly to $\mathbb{V}$ or can be used to more easily compute the univariate polynomials for the first k steps.

The problem with this idea is that as v is a polynomial of degree d in each X_i, to determine v we would need to compute it at $(d+1)^k$ points. For obvious reasons this scales badly as k increases. The trick is to do the same idea but keep everything univariate.

[15] Indeed it's possible that in some cases these costs should outweigh benefits. This optimisation is likely to be situation specific

[16] Thanks to Shahar Papini for suggesting this idea and for useful discussions about it.

5.1 The Univariate Skip

The key idea is to replace the base hypercube by some product of the form[17] $D \times H_j$ where D is a multiplicative subgroup[18] of $\mathbb{F}_p$. Given a column of our table we interpret this as the polynomial $f_i(x_0, x_1, \cdots, x_j)$ with

$$\deg_{x_k} f_i(x_0, \cdots, x_j) \leq \begin{cases} |D| - 1 & k = 0 \\ 1 & otherwise. \end{cases}$$

As C has maximal degree d in the f_i, we see that

$$\deg_{x_0}(C) = d(|D| - 1).$$

Thus, in the initial round, $\mathbb{P}$ computes the univariate polynomial

$$v(X) = \sum_{\mathbf{x} \in H_j} \hat{\delta}_j(\boldsymbol{\alpha}, \mathbf{x}) C(X, \mathbf{x})$$

which will be a polynomial of degree $d(|D| - 1)$. If $|D| = 2^k$ (which would correspond to skipping the first k rounds) $\mathbb{P}$ needs to compute only $d(2^k - 1)$ evaluations which is far less than the $(d+1)^k$ evaluations the earlier attempt required.

Evaluations with $x \in D$ are free as the constraint evaluates to 0 on $D \times H_j$ and so the total number of evaluations needed will be

$$2^j(d(|D| - 1) + 1) - 2^j|D|.$$

The simplest case to compare occurs when $|D| = 2^k$ and $j = n - k$ were the cost becomes

$$2^{n-k}(d - 1)(2^k - 1)C_{\mathbb{F}}.$$

This compares very favourably to the standard approach which costs

$$2^{n-1}(d - 1)C_{\mathbb{F}} + 2^{n-k}(d - 1)(2^{k-1} - 1)C_{\mathbb{G}}.$$

for the first k rounds. Hence this methods directly replaces $C_{\mathbb{G}}$ evaluations with $C_{\mathbb{F}}$ ones.

The main drawbacks are that it involves sending more data and increases the amount of work that $\mathbb{V}$ will need to do as they will need to compute a polynomial interpolation of size $\sim |D|d$.

There is however also a potentially massive hidden advantage which is how it interacts with padding. In general padding is needed to ensure that the columns of the table have length 2^n. However, using this approach we can relax this restriction to columns having size $k2^n$ for some k diving $p - 1$. Depending on the situation this may be able to massively reduce the amount of padding which will lead to further large speedups.

[17] It is also possible here to use a collection of polynomials $\hat{p} : D \to H_i$ but the resulting polynomial is usually higher degree.

[18] It's also possible to get this working for more general D.

6 An Efficient Zero Check Protocol

Assume that $\mathbb{P}$ wishes to prove that a degree d polynomial C vanishes on every row of a table T of size[19] $2^n \times l$. Additionally fix an integer parameter k which captures a tradeoff between prover work and verifier work.

1. Setup and Round 0

   (a) $\mathbb{P}$ indexes the rows of the table by $D \times H_{n-k}$. Each column i is interpreted as evaluations of a polynomial $f_i(x_0, x_1, \cdots, x_{n-k})$ satisfying

   $$\deg_{x_j} f_i = \begin{cases} |D| - 1 & j = 0 \\ 1 & j \in [1, \cdots, n-k]. \end{cases}$$

   $\mathbb{P}$ commits to a constraint function

   $$C(x_0, x_1, \cdots, x_{n-k}) = C\big(\omega_0(x_0, \cdots, x_{n-k}), \cdots, \omega_{l-1}(x_0, \cdots, x_{n-k})\big)$$

   of total degree $\leq d$ in the f's and which holds over the rows of the table.

   (b) $\mathbb{V}$ samples a vector $\boldsymbol{\alpha} \in \mathbb{G}^{n-k}$ and sends it to $\mathbb{P}$.

   (c) $\mathbb{P}$ computes

   $$v_0(X) = \sum_{\mathbf{x} \in H_{n-k}} \hat{\delta}_{n-k}(\boldsymbol{\alpha}, \mathbf{x}) C(X, \mathbf{x})$$

   for $(d-1)(|D|-1)+1$ distinct values of X all disjoint from $|D|$ and sends the results to $\mathbb{V}$.

   (d) $\mathbb{V}$ samples $r_0 \in G$ and sends to $\mathbb{P}$. Additionally, $\mathbb{V}$ interpolates $v_0(X)$ from the data sent by $\mathbb{P}$ along with $v_0(X) = 0$ for all $X \in |D|$ and computes $v_0 = v_0(r_0)$.

2. Rounds $1 \cdots n-k$.

   At the start of the i'th round, $\mathbb{V}$ has just computed $v_{i-1} = v_{i-1}(r_{i-1})$ and sent r_{i-1} to the $\mathbb{P}$ who has added it to $\mathbf{r} = (r_0, \cdots, r_{i-1})$. Provided $i < n-k+1$ we repeat:

   (a) $\mathbb{P}$ computes

   $$v_i(X) = \sum_{\mathbf{X} \in H_{n-k-i}} \hat{\delta}_{n-k-i}((\alpha_{i+1}, \cdots, \alpha_{n-k}), \mathbf{x}) C(\mathbf{r}, X, \mathbf{x})$$

   for $X \in 0, 2, 3, \cdots, d$ and sends to $\mathbb{V}$.

[19] As commented earlier, this protocol works identically (And indeed more efficiently) if the table has length $|D|2^{n-k}$ for $|D| < 2^k$. This potentially allows for a zerocheck protocol with far less padding.

(b) $\mathbb{V}$ samples $r_i \in G$ and sends to $\mathbb{P}$. Then, $\mathbb{V}$ interpolates $v_i(X)$ from the data sent by $\mathbb{P}$ along with

$$v_i(1) = \frac{1}{\alpha_i}\left(v_{i-1} - (1-\alpha_i)v_i(0)\right)$$

and computes $v_i = v_i(r_i)$.

3. Final Check

(a) After r_n has been sent, $\mathbb{P}$ computes $C(\mathbf{r})$ and sends it to $\mathbb{V}$, along with a commitment proof.

(b) $\mathbb{V}$ accepts if and only if $C(\mathbf{r}) = v'_n$.

The proof that this protocol is cryptographically sound follows from a simple tweak to the standard argument and can be found in section A.4.

6.1 Prover and Verifier Costs

It is immediate that the verifier ($\mathbb{V}$) needs to perform a only a single interpolation of degree $(|D|-1)d = (2^k-1)d$ and $n-k$ interpolations of degree d.

In round 0, the prover needs to evaluate $C(y, \mathbf{X})$ for all $(y, \mathbf{X}) \in [|D|, \cdots, d(|D|-1)] \times H_{n-k}$. This is $(d-1)(|D|-1)2^{n-k} = (d-1)(2^k-1)2^{n-k}$ evaluations in total all occurring in $\mathbb{F}$.

In round $i > 0$, the prover needs to evaluate $C(\mathbf{r}, y, \mathbf{X})$ for $(y, \mathbf{X}) \in [0, 2, 3, \cdots, d] \times H_{n-k-i}$. This initially looks like $d2^{n-k-i}$ evaluations but is actually $(d-1)2^{n-k-i}$ as the $y = 0$ case can be determined from data computed in previous rounds using the method in Section 4. These evaluations are all over $\mathbb{G}$.

Hence the total number of constraint evaluations for $\mathbb{P}$ is

$$\begin{aligned}(d-1)(2^k-1)2^{n-k}C_{\mathbb{F}} + &\sum_{i=1}^{n-k}(d-1)2^{n-k-i}C_{\mathbb{G}} + C_{\mathbb{G}} \\ &\sim 2^{n-k}(d-1)(2^k-1)C_{\mathbb{F}} + 2^{n-k}(d-1)C_{\mathbb{G}} \\ &\sim 2^n\left(1 + \frac{C_{\mathbb{G}}}{2^k C_{\mathbb{F}}}\right)(d-1)C_{\mathbb{F}}.\end{aligned}$$

Letting $\mathbb{G}$ be an extension of degree 4 we assume that

$$C_{\mathbb{G}} \sim 16C_{\mathbb{F}}$$

due to the increased multiplication cost. Then setting $d = 3$ the cost becomes

$$2^{n+1}\left(1 + \frac{16}{2^k}\right)C_{\mathbb{F}}$$

Compared to the initial cost of

$$5 \times 17 \times 2^{n-1} C_{\mathbb{F}}.$$

Setting $k = 4$ we find that $\mathbb{V}$ needs to do a single interpolation of degree 48 but in exchange we have reduced the work of $\mathbb{P}$ by a factor of

$$\frac{5 \times 17}{8} \sim 10.6$$

Thus even with conservative choices of k, this improved protocol will lead to a decent speed up to any proof systems which use zerocheck.

Acknowledgements

Many thanks to Daniel Lubarov, Ulrich Haböck, Trung Can and Shahar Papini, Ole Spjeldnæs, for useful ideas, comments and discussions.

References

[1] Jonathan Bootle, Alessandro Chiasa, and Katerina Sotiraki. “Sumcheck Arguments and their Applications”. In: *CRYPTO 2021*. https://eprint.iacr.org/2021/333. 2021.

[2] Binyi Chen et al. *HyperPlonk: Plonk with Linear-Time Prover and High-Degree Custom Gates*. Cryptology ePrint Archive, Report 2019/1355. https://eprint.iacr.org/2022/1355. Oct. 2022.

[3] Carsten Lund et al. “Algebraic Methods for Interactive Proof Systems”. In: *J. ACM* 39.4 (Oct. 1992), pp. 859–868. ISSN: 0004-5411. DOI: 10.1145/146585.146605.

[4] Or Meir. “IP = PSPACE using Error-Correcting Codes”. In: *SIAM Journal on Computing*. Vol. 42. Full paper: https://eccc.weizmann.ac.il/report/2010/137. 2013.

[5] Srinath Setty. “Spartan: Efficient and General-Purpose zkSNARKs without Trusted Setup”. In: *CRYPTO 2020*. Full paper: https://eprint.iacr.org/2019/550. 2020.

[6] Justin Thaler. *Proofs, Arguments, and Zero-Knowledge*. Jan. 2022. ISBN: 978-1-63828-124-5. DOI: 10.1561/9781638281252.

[7] Justin Thaler. *The Sum-Check Protocol over Fields of Small Characteristic*. https://api.semanticscholar.org/CorpusID:265352651. Nov. 2023.

A Soundness Proofs

Completeness and soundness are are 2 crucial properties of protocols that always need to be checked. Completeness refers to the probability that $\mathbb{P}$ will be unable to convince $\mathbb{V}$ to accept a true statement and soundness refers to the probability that a malicious prover ($\mathbb{MP}$) will be able to convince $\mathbb{V}$ to accept a false statement. In our applications here, our protocols will all be complete meaning they have a completness of 1. Thus most of the work will be in checking that the protocol is cryptographically sound, meaning its soundness is $< 2^{-\delta}$ for some $\delta > 100$.

The proofs here are all either standard and appear elsewhere in other literature (e.g. [Lun+92; Che+22]) or are relatively simple tweaks to the standard proofs to match the protocol tweaks we propose. We provide them for completeness and for readers less familiar with the topic.

For our soundness proofs we will distinguish between the true function (v_i) and what $\mathbb{MP}$ sends to $\mathbb{V}$ ($\overline{v}_i$) using an overbar. Additionally, for the protocols here, there are no interesting strategies of $\mathbb{MP}$ other than simply hoping by random chance $\overline{v}_i$ and v_i are equal at the chosen sampled point.

A.1 SumCheck

Completeness is essentially trivial. If the claim is true and $\mathbb{P}$ follows the protocol then $\mathbb{V}$ will always accept. To check soundness, assume that $\mathbb{MP}$ wishes to falsely prove that

$$\overline{v}_0 = \sum_{\mathbf{x} \in H_n} \hat{f}(\mathbf{x}).$$

In order to not get caught in round 0, $\mathbb{MP}$ must send $\overline{v}_1(0), \overline{v}_1(1)$ such that $\overline{v}_1(0) + \overline{v}_1(1) = \overline{v}_0 \neq v_0$. Hence either $\overline{v}_1(0) \neq v_1(0)$ or $\overline{v}_1(1) \neq v_1(1)$ or both.

Applying the Schwartz-Zippel lemma to $v_1, \overline{v}_1$, this means that with probability[20] $\frac{d}{|\mathbb{G}|}$, $\overline{v}_1(r_0) \neq v_1(r_0)$. If these are equal then $\mathbb{MP}$ has succeeded in fooling the verifier and if not, we can repeat this argument for round 1 and so on so forth. Applying the union bound[21], the probability that $\mathbb{MP}$ can fool the verifier will be

$$\frac{nd}{|\mathbb{G}|} = \frac{nd}{|\mathbb{F}|^m}$$

and so the protocol is sound provided $\mathbb{G}$ is sufficiently large.

A.2 ZeroCheck

Note that if $C(T_{i,0}, \cdots, T_{i,l-1})$ starts as a polynomial of total degree $\leq d$, then all individual degrees $C(\mathbf{X})$ are also bounded by d and so

$$\deg_{X_i} \hat{\delta}_n(\boldsymbol{\alpha}, \mathbf{X}) C(\mathbf{X}) \leq d + 1 \tag{3}$$

which is known by $\mathbb{V}$ assuming they already know d. Hence $\mathbb{P}$ and $\mathbb{V}$ are able to run the protocol as claimed[22].

Next observe that if $C(\mathbf{X}) = 0$ for all $\mathbf{X} \in H_n$ then $\hat{F}(\boldsymbol{\alpha})$ is precisely the 0 polynomial. Thus completeness follows from the completeness of sumcheck.

On the other hand, if $\exists\ \mathbf{y} \in H_n$ such that $C(\mathbf{y}) \neq 0$, then

$$\hat{F}(\mathbf{y}) = \sum_{\mathbf{x} \in H_n} \delta(\mathbf{y}, \mathbf{x}) C(\mathbf{x}) = C(\mathbf{y}) \neq 0$$

and so $\hat{F}(\boldsymbol{\alpha})$ is not the zero polynomial. As $\hat{F}(\boldsymbol{\alpha})$ is a multilinear polynomial in $\boldsymbol{\alpha}$, the Schwartz–Zippel lemma then shows that

$$Pr\left[(\hat{F}(\boldsymbol{\alpha}) = 0 \mid \boldsymbol{\alpha} \in \mathbb{G}^n\right] \leq \frac{n}{|\mathbb{G}|}.$$

[20] It is essential here that the r_j are sampled in $\mathbb{G}$ and not $\mathbb{F}$

[21]
$$\Pr\left(\bigcup_i E_i\right) \leq \sum_i \Pr(E_i)$$

[22] We need to check this as the sumcheck protocol does not work if $\mathbb{P}$ is allowed to send polynomials of arbitrary degree to $\mathbb{V}$.

Hence by taking a union bound, the soundness of zerocheck will be $\frac{n}{|\mathbb{G}|}$ plus the soundness of sumcheck which, will be $\frac{n(d+1)}{|\mathbb{G}|}$ due to the degree bound computed in Equation (3) giving an overall soundness of

$$\frac{n(d+2)}{|\mathbb{G}|}.$$

A.3 Tweaked ZeroCheck

As before if the claim is true and $\mathbb{P}$ is honest they can simply[23] follow the steps above and $\mathbb{V}$ will accept.

Soundness is also abstractly clear if we recall that, at every stage $\mathbb{V}$ is able to compute what $\mathbb{P}$ would have sent in the original protocol. Hence the soundness must as worst be equal. However we run through the argument again here just to triple check everything.

Assume that $\mathbb{MP}$ wishes to falsely prove that every row satisfies a constraint C. This means that with probability $\frac{n}{|\mathbb{G}|}$,

$$v_0 = \sum_{\mathbf{x} \in H_n} \hat{\delta}_n(\boldsymbol{\alpha}, \mathbf{x}) C(\mathbf{x}) \neq 0.$$

As

$$v_0 = \alpha_0 v_1'(1) + (1 - \alpha_0) v_1'(0) \neq 0,$$

at least one of $v_1'(0), v_1'(1)$ must be non 0. By construction, $\mathbb{V}$ always assumes $v_1'(0) = v_1'(1) = 0$ and so, regardless of what $\mathbb{MP}$ sends for the other values, with probability $\frac{d}{|\mathbb{G}|}$, $\overline{v}_1'(r_0) \neq v_1'(r_0)$. If these are equal then $\mathbb{MP}$ has succeeded in fooling the verifier and if not, we can repeat a similar argument. In round i,

$$v_i' = \alpha_i v_{i+1}'(1) + (1 - \alpha_i) v_{i+1}'(0)'$$

and so, if $v_i' \neq \overline{v}_i'$ then $(v_{i+1}'(0), v_{i+1}'(1)) \neq (\overline{v}_{i+1}'(0), \overline{v}_{i+1}'(1))$. Thus, again with probability $\frac{d}{|\mathbb{G}|}$, $\overline{v}_i'(r_i) \neq v_i'(r_i)$.

Applying the union bound, the probability that $\mathbb{MP}$ can fool the verifier is bounded above by[24]

$$\frac{n}{|\mathbb{G}|} + n\frac{d}{|\mathbb{G}|} = \frac{n(d+1)}{|\mathbb{G}|}.$$

Hence not only has the cost decreased, the soundness has also improved and so this is a strict improvement over the standard zerocheck protocol.

[23] We do need to assume that α_i is never 0 but this doesn't change the soundness analysis in the slightest so is fine to do. It's a good general policy to sample $\boldsymbol{\alpha} \in G^n \backslash \{0, 1\}$ anyway.

[24] This analysis can be improved slightly in several directions. E.g. we actually do not need to send/consider α_0. But the improvements are all small.

A.4 Improved ZeroCheck

As each f_i has degree $|D|-1$ in x_0, C has degree $d(|D|-1)$ in x_0 and thus $v_0(X)$ has degree $d(|D|-1)$. Hence $\mathbb{V}$ is able to interpolate $v_0(X)$ from the data sent in round 1 along with the assumption $v_0(y)=0$ for all $y \in D$. In future rounds, v_i has degree d and the prover sends values for $0,2,3,\cdots d$. Thus as $\mathbb{V}$ knows $\boldsymbol{\alpha}$ they can compute $v_i(X)$ and so the verifier is able to follow the protocol as stated and if an honest prover simply follows the protocol it will always be accepted.

Assume again that a malicious prover ($\mathbb{MP}$) wishes to falsely prove that every row satisfies a constraint C. This means that we can find a row $(y,\mathbf{x}) \in |D| \times H_{n-k}$ with $C(y,\mathbf{x}) \neq 0$ and so with probability $\frac{n-k}{|\mathbb{G}|}$,

$$v_0(y) = \sum_{\mathbf{x} \in H_{n-k}} \hat{\delta}_{n-k}(\boldsymbol{\alpha},\mathbf{x}) C(y,\mathbf{x}) \neq 0.$$

As $\mathbb{V}$ assumes that $v_0(y)=0$ for all $y \in |D|$ when it interpolates $v_0(X)$, regardless of what $\mathbb{MP}$ sends for the other values, with probability $\frac{d(|D|-1)}{|\mathbb{G}|}$, $\overline{v}_0(r_0) \neq v_0(r_0)$. If these are equal then $\mathbb{MP}$ has succeeded in fooling the verifier and if not, then we are in an identical situation to the tweaked zerocheck protocol so the soundness proof from there carries through.

Applying the union bound, the probability that $\mathbb{MP}$ can fool the verifier is bounded above by

$$\begin{aligned} \frac{n-k}{|\mathbb{G}|} + \frac{d(|D|-1)}{|\mathbb{G}|} + (n-k)\frac{d}{|\mathbb{G}|} &= \frac{d(|D|-1)+(n-k)(d+1)}{|\mathbb{G}|} \\ &= \frac{d(2^k-1)+(n-k)(d+1)}{|\mathbb{G}|} \end{aligned}$$