plookup: A simplified polynomial protocol for lookup tables

Ariel Gabizon
Aztec

Zachary J. Williamson
Aztec

March 24, 2020

Abstract

We present a protocol for checking the values of a committed polynomial $f \in \mathbb{F}_{<n}[X]$ over a multiplicative subgroup $H \subset \mathbb{F}$ of size n, are contained in the values of a table $t \in \mathbb{F}^d$. Our protocol can be viewed as a simplification of one from Bootle et. al [BCG$^+$] for a similar problem, with potential efficiency improvements when $d \leq n$. In particular, [BCG$^+$]'s protocol requires comitting to several auxiliary polynomials of degree $d \cdot \log n$, whereas ours requires three commitments to auxiliary polynomials of degree n, which can be much smaller in the case $d \sim n$.

One common use case of this primitive in the zk-SNARK setting is a "batched range proof", where one wishes to check all of f's values on H are in a range $[0, \ldots, M]$. We present a slightly optimized protocol for this special case, and pose improving it as an open problem.

1 Introduction

When wanting to use zk-SNARKs to prove statements involving standard primitives like AES-128 or SHA-256, one runs into the problem that the operations involved in these primitives are "SNARK unfriendly", in the sense of having large representations in the native SNARK language, which typically corresponds to low degree equations over a large prime field. Examples of operations with large overhead, are ones involving bit decomposition like bitwise XOR or AND. This has lead to growing research into STARK and SNARK friendly hash functions and symmetric primitives that are based solely on native field operations [GKK$^+$19, AAB$^+$19, AGR$^+$16, ACG$^+$19].

We investigate an alternative approach, where for commonly used operations we precompute a lookup table of the legitimate (input, output) combinations; and the prover argues the witness values exist in this table.

Using randomness, we can reduce to the case of looking up single field elements instead of tupples. Then, in polynomial language, this ultimately boils down to proving polynomials are the same "up to multiplicities". That is, suppose that the values in the

lookup table are $\{t_i\}_{i\in[d]}$ and the values in the witness are $\{f_i\}_{i\in[n]}$. We want to show that the polynomials

$$F(X) := \prod_{i\in[n]} (X - f_i), G(X) := \prod_{i\in[d]} (X - t_i)$$

have the same roots, ignoring multiplicites; i.e. that for some non-negative integers $\{e_i\}_{i\in[d]}$ we have $F(X) = \prod_{i\in[d]}(X - t_i)^{e_i}$. Bootle et. al [BCG$^+$] gave an algorithm for this exact problem, also in the context of efficient SNARK arithmetization of a common operation (in their case, repeatedly checking that field elements correspond to certain convenient sparse representations of boolean strings).

Their algorithm requires committing to a vector of length $d \log n$ that contains for each $i \in [d], j \in [\log n]$ the value $(x - t_i)^{2^j}$ for a random verifier challenge $x \in \mathbb{F}$. They also commit to the binary decomposition of the $\{e_i\}$, and using the two, prove that F is of the desired form.

We present here an arguably simpler protocol for the same problem, that doesn't require explicitly representing the multiplicities. Let us use the notation $f \subset t$ as shorthand for $\{f_i\}_{i\in[n]} \subset \{t_i\}_{i\in[d]}$.

The idea is to look at a sorted version $\{s_i\}$ of the values $\{f_i\}$, and compare the set of non-zero *differences* in $\{s_i\}$ and $\{t_i\}$. Note that if $f \subset t$, and every element of t appears at least once in f, we indeed have that these sets of differences are the same. However, the converse is not true: We can create a sequence of values $\{s_i\}$ having the same difference set as $\{t_i\}$, but with the differences appearing in *different order*; and in this case we won't have $s \subset t$. As an illustrating example, take

$$t = \{1, 4, 8\}, s = \{1, 1, 4, 8, 8, 8\}, s' = \{1, 5, 5, 5, 8, 8\}$$

All three sets have the same difference set $\{3, 4\}$; but since those differences appear in different order in s' and t, we don't have $s' \subset t$. Though not an issue in the above examples, we also didn't address checking the starting point is the same in s and t. Both issues can be solved by comparing *randomized* difference sets: We choose random $\beta \in \mathbb{F}$, and compare the elements in the sequences $\{t_i + \beta t_{i+1}\}_{i\in[d-1]}$, $\{s_i + \beta s_{i+1}\}_{i\in[n-1]}$.

Seemingly, we have taken "one step forward and one back" by adding the random β, as now pairs (s_i, s_{i+1}) with $s_i = s_{i+1}$ will give a non-zero contribution $s_i + \beta \cdot s_{i+1} = (1 + \beta) \cdot s_i$. However, all elements corresponding to a repetition are now multiples of $(1+\beta)$ and we are able to use this to "identify" them and not use them in the comparison with the table. Specifically, we show that this check can be done correctly and efficiently using a "grand product argument" similar to the one used in [GWC19]'s permutation argument.

A technicality is that as mentioned this approach assumes all of t's values appear at least once in f; which is one reason we in fact take s to be the sorted version of the *concatenation* of f and t. This ends up also helping to batch both checks into one product: The check that $f \subset s$, and the check that $s \subset t$.

Precise details of the scheme are given in Section 3.

2 Preliminaries

Terminology For integer d, we denote by $\mathbb{F}_{<d}[X]$ the set of univariate polynomials over $\mathbb{F}$ of degree smaller than d.

Polynomial Protocols We use the ranged polynomial protocol terminology from [GWC19] to describe our protocols. We repeat the definition for reference.

In such a protocol a prover $\mathbf{P}$ sends polynomials of a certain degree bound d to an ideal party $\mathcal{I}$, and at the end of protocol, the verifier can ask whether certain identities hold between the polynomials sent during the protocol, the input polynomials, and the preprocessed polynomials, on a predefined set H.

More precisely,

Definition 2.1. *Fix positive integers d, D, t, ℓ and $H \subset \mathbb{F}$. An H-ranged (d, D, t, ℓ)-*polynomial protocol *is a multiround protocol between a prover $\mathbf{P}$, verifier $\mathbf{V}$ and trusted party $\mathcal{I}$ that proceeds as follows.*

1. *The protocol definition includes a set of* preprocessed and input polynomials $g_1, \ldots, g_\ell \in \mathbb{F}_{<d}[X]$.
2. *The messages of $\mathbf{P}$ are sent to $\mathcal{I}$ and are of the form f for $f \in \mathbb{F}_{<d}[X]$. If $\mathbf{P}$ sends a message not of this form, the protocol is aborted.*
3. *The messages of $\mathbf{V}$ to $\mathbf{P}$ are arbitrary (but we will concentrate on public coin protocols where the messages are simply random coins).*
4. *At the end of the protocol, suppose $f_1, \ldots, f_t$ are the polynomials that were sent from $\mathbf{P}$ to $\mathcal{I}$. $\mathbf{V}$ may ask $\mathcal{I}$ if certain polynomial identities holds* on *H between $\{f_1, \ldots, f_t, g_1, \ldots, g_\ell\}$. Where each identity is of the form*

$$F(X) := G(X, h_1(v_1(X)), \ldots, h_M(v_M(X))) \equiv 0,$$

   for some $h_i \in \{f_1, \ldots, f_t, g_1, \ldots, g_\ell\}$, $G \in \mathbb{F}[X, X_1, \ldots, X_M]$, $v_1, \ldots, v_M \in \mathbb{F}_{<d}[X]$ such that $F \in \mathbb{F}_{<D}[X]$ for every choice of $f_1, \ldots, f_t$ made by $\mathbf{P}$ when following the protocol correctly.
5. *After receiving the answers from $\mathcal{I}$ regarding the identities, $\mathbf{V}$ outputs* acc *if all identities hold, and outputs* rej *otherwise.*

It is shown in Section 4 of [GWC19] how such a protocol can be compiled into one in the algebraic group model using the polynomial commitment scheme of [KZG10].

For such a protocol $\mathscr{P}$, let $\mathfrak{d}(\mathscr{P})$ be the maximum over $f_1, \ldots, f_t$ sent by an honest prover during protocol execution of $\left(\sum_{i \in [t]} \deg(f_i) + 1\right) + D - |H|$.

The prover complexity ends up being closely tied to $\mathfrak{d}(\mathscr{P})$ as this corresponds to the number of group operations to commit to all polynomials (including the quotient polynomial that comes up in translating a ranged polynomial protocol to a polynomial protocol) using [KZG10]. We will thus attempt to minimize $\mathfrak{d}(\mathscr{P})$ in our protocols.

Lagrange bases and multiplicative subgroups In our protocols, we take H to be a multiplicative subgroup of some order N with generator $\mathbf{g}$. For $i \in [N]$, we denote by $L_i \in \mathbb{F}_{<N}[X]$ the i'th Lagrange polynomial for H, that satisfies $L_i(\mathbf{g}^i) = 1$ and $L_i(\mathbf{g}^j) = 0$ for $j \neq i$. These polynomials are covenient when specifying a point check in an H-ranged protocol. For example, requiring $L_i(\mathbf{x})f(\mathbf{x}) = 0$ for all $\mathbf{x} \in H$ is equivalent to $f(\mathbf{g}^i) = 0$.

The generator $\mathbf{g}$ is convenient for specifying constraints on "neighboring values". For example, the constraint $f(\mathbf{g} \cdot \mathbf{x}) - f(\mathbf{x}) = 1$ means that f's value grows by one when going to the "next" point.

We refer the reader to Section 4 of [GWC19] for more details.

3 The main scheme

Notation: Fix integers n, d and vectors $f \in \mathbb{F}^n, t \in \mathbb{F}^d$. We use the notation $f \subset t$ to mean $\{f_i\}_{i\in[n]} \subset \{t_i\}_{i\in[d]}$. Let $H = \{\mathbf{g}, \ldots, \mathbf{g}^{n+1} = 1\}$ be a multiplicative subgroup of order $n + 1$ in $\mathbb{F}$. For a polynomial $f \in \mathbb{F}[X]$ and $i \in [n + 1]$ we sometimes denote $f_i := f(\mathbf{g}^i)$. For a vector $f \in \mathbb{F}^n$, we also denote by f the polynomial in $\mathbb{F}_{<n}[X]$ with $f(\mathbf{g}^i) = f_i$.

When $f \subset t$, we say that f is *sorted by* t when values appear in the same order in f as they do in t. Formally, for any $i < i' \in [n]$ such that $f_i \neq f_{i'}$, if $j, j' \in [d]$ are such that $t_j = f_i, t_{j'} = f_{i'}$ then $j < j'$.

Now, given $t \in \mathbb{F}^d, f \in \mathbb{F}^n, s \in \mathbb{F}^{n+d}$, define bi-variate polynomials F, G as

$$F(\beta, \gamma) := (1 + \beta)^n \cdot \prod_{i\in[n]} (\gamma + f_i) \prod_{i\in[d-1]} (\gamma(1 + \beta) + t_i + \beta t_{i+1})$$

$$G(\beta, \gamma) := \prod_{i\in[n+d-1]} (\gamma(1 + \beta) + s_i + \beta s_{i+1})$$

we have

Claim 3.1. *$F \equiv G$ if and only if*

1. *$f \subset t$, and*
2. *s is (f, t) sorted by t.*

Proof. We write F, G as elements of $\mathbb{F}(\beta)[\gamma]$ while taking out a $(1 + \beta)$ factor as follows.

$$F(\beta, \gamma) = (1 + \beta)^{n+d-1} \cdot \prod_{i\in[n]} (\gamma + f_i) \prod_{i\in[d-1]} (\gamma + (t_i + \beta t_{i+1})/(1 + \beta))$$

$$G(\beta, \gamma) = (1 + \beta)^{n+d-1} \prod_{i\in[n+d-1]} (\gamma + (s_i + \beta s_{i+1})/(1 + \beta))$$

Suppose first that $f \subset t$ and $s \in \mathbb{F}^{n+d}$ is (f, t) sorted by t.

Then for each $j \in [d-1]$, there is a distinct index $i \in [n+d-1]$ such that $(t_j, t_{j+1}) = (s_i, s_{i+1})$. The corresponding factors in F, G are equal. That is,

$$(\gamma + (t_j + \beta t_{j+1})/(1+\beta)) \equiv (\gamma + (s_i + \beta s_{i+1})/(1+\beta))$$

Let $P' \subset [n+d-1]$ the set of these $d-1$ indices i , and let $P := [n+d-1] \setminus P'$. The n indices $i \in P$ are such that $s_i = s_{i+1}$, and $\{s_i\}_{i \in P}$ equals $\{f_i\}_{i \in [n]}$ as multisets. That is, we have a one-to-one map $j : P \to [n]$ such that for each $i \in P$, $s_i = f_{j(i)}$. For each $i \in P$, the corresponding factor of G, will be

$$\gamma + (s_i + \beta s_{i+1})/(1+\beta) = \gamma + s_i,$$

which equals the factor $\gamma + f_{j(i)}$ in F.

For the other direction, assume $F \equiv G$ as polynomials in $\mathbb{F}[\beta, \gamma]$. Then $F \equiv G$ also as elements of $\mathbb{F}(\beta)[\gamma]$. Since $\mathbb{F}(\beta)[\gamma]$ is a unique factorization domain, we know that the linear factors of F, G, as written above must be equal. Thus, for each $i \in [d-1]$, G must have a factor equal to $(\gamma + (t_i + \beta t_{i+1})/(1+\beta))$. In other words, for some $j \in [n+d-1]$,

$$\gamma + (t_i + \beta t_{i+1})/(1+\beta) = \gamma + (s_j + \beta s_{j+1})/(1+\beta),$$

which implies $t_i + \beta t_{i+1} = s_j + \beta s_{j+1}$, and therefore $t_i = s_j, t_{i+1} = s_{j+1}$. Call $P' \subset [n+d-1]$ the set of these $d-1$ indices j. For any index $j \in [n+d-1] \setminus P'$, there must be a factor "coming from f" in F that equals the corresponding factor in G. More precisely, for such j there exists $i \in [n]$ such that

$$\gamma + f_i = \gamma + (s_j + \beta s_{j+1})/(1+\beta),$$

or equivalently

$$f_i + \beta f_i = s_j + \beta s_{j+1}$$

which implies $f_i = s_j = s_{j+1}$.

Thus, we know that whenever consecutive values in s are different, they are exactly equal to two consecutive values in t, and all values of f are values of t.

□

Claim 3.1 motivates the following protocol. It will be convenient to assume $d = n+1$. (If $d \leq n$ pad t with $n - d + 1$ repetitions of the last element.)

Preprocessed polynomials: The polynomial $t \in \mathbb{F}_{<n+1}[X]$ describing the lookup values.

Inputs: $f \in \mathbb{F}_{<n}[X]$

Protocol:

1. Let $s \in \mathbb{F}^{2n+1}$ be the vector that is (f,t) sorted by t. We represent s by $h_1, h_2 \in \mathbb{F}_{<n+1}[X]$ as follows. $h_1(\mathbf{g}^i) = s_i$ for $i \in [n+1]$; and $h_2(\mathbf{g}^i) = s_{n+i}$ for each $i \in [n+1]$.
2. $\mathbf{P}$ computes the polynomials h_1, h_2 and sends them to the ideal party $\mathcal{I}$.
3. $\mathbf{V}$ chooses random $\beta, \gamma \in \mathbb{F}$ and sends them to $\mathbf{P}$.
4. $\mathbf{P}$ computes a polynomial $Z \in \mathbb{F}_{<n+1}[X]$ that aggregates the value $F(\beta,\gamma)/G(\beta,\gamma)$ where F, G are as described above. Specifically, we let
   (a) $Z(\mathbf{g}) = 1$,
   (b) For $2 \le i \le n$
   $$Z(\mathbf{g}^i) = \frac{(1+\beta)^{i-1}\prod_{j<i}(\gamma+f_j)\cdot\prod_{1\le j<i}(\gamma(1+\beta)+t_j+\beta t_{j+1})}{\prod_{1\le j<i}(\gamma(1+\beta)+s_j+\beta s_{j+1})(\gamma(1+\beta)+s_{n+j}+\beta s_{n+j+1})},$$
   and
   (c) $Z(\mathbf{g}^{n+1}) = 1$.
5. $\mathbf{P}$ sends Z to $\mathcal{I}$.
6. $\mathbf{V}$ checks that Z is indeed of the form described above, and that $Z(\mathbf{g}^{n+1}) = 1$. More precisely, $\mathbf{V}$ checks the following identities for all $\mathbf{x} \in H$.
   (a) $L_1(\mathbf{x})(Z(\mathbf{x}) - 1) = 0$.
   (b)
   $$(\mathbf{x} - \mathbf{g}^{n+1})Z(\mathbf{x})(1+\beta)\cdot(\gamma + f(\mathbf{x}))(\gamma(1+\beta)+t(\mathbf{x})+\beta t(\mathbf{g}\cdot\mathbf{x}))$$
   $$= (\mathbf{x}-\mathbf{g}^{n+1})Z(\mathbf{g}\cdot\mathbf{x})(\gamma(1+\beta)+h_1(\mathbf{x})+\beta h_1(\mathbf{g}\cdot\mathbf{x}))(\gamma(1+\beta)+h_2(\mathbf{x})+\beta h_2(\mathbf{g}\cdot\mathbf{x}))$$
   (c) $L_{n+1}(\mathbf{x})(h_1(\mathbf{x}) - h_2(\mathbf{g}\cdot\mathbf{x})) = 0$.
   (d) $L_{n+1}(\mathbf{x})(Z(\mathbf{x}) - 1) = 0$.

   and outputs acc iff all checks hold.

Lemma 3.2. *Suppose that $\{f(\mathbf{g}^i)\}_{i\in[n]}$ is not contained in $\{t(\mathbf{g}^i)\}_{i\in[n+1]}$. Then for any strategy of $\mathcal{A}$ playing the role of $\mathbf{P}$ in the above protocol $\mathscr{P}$, the probability that $\mathbf{V}$ accepts is $\mathsf{negl}(\lambda)$. Furthermore, we have $\mathfrak{d}(\mathscr{P}) = 5n+4$.*

Proof. We start by computing $\mathfrak{d}(\mathscr{P})$. The prover sends $h_1, h_2, Z \in \mathbb{F}_{<n+1}[X]$, which aggregates to $3n+3$. The second identity checked by the verifier is of the highest degree, containing a product of these three and the linear term $X - \mathbf{g}^{n+1}$, which gives us another $3n+2$ from which we subtract $|H| = n+1$. In total we get $5n+4$.

We proceed to prove correctness. The check in step 6c shows that $h_1(\mathbf{g}^{n+1}) = h_2(\mathbf{g})$, and thus h_1, h_2 indeed consistently describe a single vector $s \in \mathbb{F}^{2n+1}$. Using Claim

3.1 we know that when f's range is not contained in t's for any choice of s sent by $\mathbf{P}$ the polynomials $F(X,Y), G(X,Y)$ are different. From the SZ lemma e.w.p $\mathsf{negl}(\lambda)$ $\mathbf{V}$ chooses β, γ such that $F(\beta,\gamma) \neq G(\beta,\gamma)$. In this case we have $Z(\mathbf{g}^{n+1}) \neq 1$ which means $\mathbf{V}$ rejects. □

4 Generalizing to vector lookups and multiple tables

Suppose we have several witness polynomials $f_1, \ldots, f_w \in \mathbb{F}_{<n}[X]$, and a table of values $t^* \in (\mathbb{F}^w)^d$. We wish to check that for each $j \in [n]$ $(f_1(\mathbf{g}^j), \ldots, f_w(\mathbf{g}^j)) \in t^*$. We can use randomization to efficiently reduce to the case of Section 3.

For each $i \in [w]$ we will include in the set of preprocessed polynomials $t_i \in \mathbb{F}_{<d}[X]$ with $t_i(\mathbf{g}^j) = t^*_{i,j}$ for each $j \in [d]$.

The verifier will choose random $\alpha \in \mathbb{F}$.

Then we will define $t := \sum_{i\in[w]} \alpha^i t_i, f := \sum_{i\in[w]} \alpha^i f_i$.

Assume that for some $j \in [n]$, $(f_1(\mathbf{g}^j), \ldots, f_w(\mathbf{g}^j)) \notin t^*$. Then e.w.p $d \cdot w/|\mathbb{F}|$, $f(\mathbf{g}^j) \notin t$. Thus, after the selection of α, we can run the protocol of the previous section on f, t.

As alluded to in the introduction, a natural use case for this vector lookup primitive is a key-value setting, where we have a function f with $w-1$ inputs, and wish to verify a vector is of the form $(x_1, \ldots, x_{w-1}, f(x_1, \ldots, x_{w-1}))$ for some input $(x_1, \ldots, x_{w-1})$.

4.1 Multiple tables

Suppose further that we have in fact have multiple tables $t^*_1, \ldots, t^*_\ell$ and for each $i \in [n]$ wish to check that for some predefined $j = j(i) \in [\ell]$ $(f_1(\mathbf{g}^i), \ldots, f_w(\mathbf{g}^i)) \in t^*_j$. We can reduce to the previous setting as follows. We create a preprocessed table containing $t^*_1, \ldots, t^*_\ell$ as sub-tables, by adding a column specifying the table index.

That is, suppose for simplicity that for each $j \in [\ell]$, $t^*_j \in (\mathbb{F}^w)^{d/\ell}$. We construct $t^* \in (\mathbb{F}^{w+1})^d$, containing for each $j \in [\ell], i \in [d/\ell]$ of the element $(j, (t^*_j)_i)$. We preprocess a polynomial $q \in \mathbb{F}_{<n}[X]$ such that $q_i = j(i)$, where again $j(i)$ is the subtable we wish the i'th value to be in. Now apply the method above to check that for each $i \in [n]$, $(q(\mathbf{g}^i), f_1(\mathbf{g}^i), \ldots, f_w(\mathbf{g}^i)) \in t^*$

5 An optimized solution for continuous ranges

Suppose we wish to check that $f \subset \{0, \ldots, d-1\}$ for some integer $d < n$. We could use our above protocol while setting $t_i = i - 1$ for $i \in [d]$. By using $d = n + 1$, the above protocol allows us to check $f \subset \{0, \ldots, n\}$ with complexity $\mathfrak{d}(\mathscr{P}) = 5n + 4$ as stated in Lemma 3.2. We present an alternative protocol with the same complexity that will allow us to check $f \subset \{0, \ldots, 2n-2\}$. In fact, the protocol naturally generalizes for ranges $\{0, \ldots, c(n-1)\}$ while only increasing the degree of verifier constraints. Thus, one may choose a larger c in protocols where the range proof is a subroutine, according to the maximal constraint degree in other parts of the protocol.

The idea is the following. (We emphasize we are assuming the range is in fact contained in $\mathbb{F}$, i.e. $cn < |\mathbb{F}|$.) Suppose that we enforce that the sorted witness s starts from zero and ends at $c\cdot(n-1)$, i.e. $s_1 = 0$, $s_{2n+1} = c\cdot(n-1)$, and that for each $i \in [2n]$, $s_{i+1} - s_i \leq c$. This sufficies to deduce that $s_i \in \{0, \ldots, c(n-1)\}$ for each $i \in [2n+1]$.

The condition $s_{i+1} - s_i \leq c$ can be enforced by a constraint that plugs in this difference into the degree $c+1$ polynomial that vanishes on $\{0, \ldots, c\}$. Enforcing the increments this way obviates the need to look at a permutation between differences, and instead we can directly check a permutation between the values of (f, t) and s; where t is the table of c'th multiples, i.e. $t_i = c \cdot (i-1)$ for $i \in [n]$.

We proceed to describe the protocol in detail. As in the first protocol, we assume H is a multiplicative subgroup of order $n+1$ with generator $\mathbf{g}$.

The protocol is parameterized by a positive integer parameter c. And we denote by P the polynomial $P(X) := \prod_{i=0}^{c}(X - i)$.

Preprocessed polynomials: The polynomial $t \in \mathbb{F}_{<n}[X]$ with $t_i = c \cdot (i-1)$ for $i \in [n]$.

Inputs: $f \in \mathbb{F}_{<n}[X]$

Protocol:

1. Let $s \in \mathbb{F}^{2n+1}$ be the vector that is (f, t) sorted by t. We represent s by $h_1, h_2 \in \mathbb{F}_{<n+1}[X]$ as follows. $h_1(\mathbf{g}^i) = s_i$ for $i \in [n+1]$; and $h_2(\mathbf{g}^i) = s_{n+i}$ for each $i \in [n]$ and $h_2(\mathbf{g}^{n+1}) = c(n-1)$.
2. $\mathbf{P}$ computes the polynomials h_1, h_2 and sends them to the ideal party $\mathcal{I}$.
3. $\mathbf{V}$ chooses random $\gamma \in \mathbb{F}$ and sends it to $\mathbf{P}$.
4. $\mathbf{P}$ computes a polynomial $Z \in \mathbb{F}_{<n+1}[X]$ that aggregates the value $F(\beta, \gamma)/G(\beta, \gamma)$ where F, G are as described above. Specifically, we let
   (a) $Z(\mathbf{g}) = 1$,
   (b) For $2 \leq i \leq n$
   $$Z(\mathbf{g}^i) = \frac{\prod_{j<i}(\gamma + f_j) \cdot \prod_{1 \leq j < i}(\gamma + t_j)}{\prod_{1 \leq j < i}(\gamma + s_j)(\gamma + s_{n+j})},$$
   and
   (c) $Z(\mathbf{g}^{n+1}) = 1$.
5. $\mathbf{P}$ sends Z to $\mathcal{I}$.
6. $\mathbf{V}$ checks the following identities for all $\mathbf{x} \in H$.
   (a) $L_1(\mathbf{x})(h_1(\mathbf{x})) = 0$.
   (b) $P\left(h_1(\mathbf{g} \cdot \mathbf{x}) - h_1(\mathbf{x})\right) = 0$.

(c) $P\left(h_2(\mathbf{g}\cdot\mathbf{x}) - h_2(\mathbf{x})\right) = 0.$

(d) $L_{n+1}(\mathbf{x})(h_1(\mathbf{x}) - h_2(\mathbf{g}\cdot\mathbf{x})) = 0$

(e) $L_{n+1}(\mathbf{x})(h_2(\mathbf{x})) = c\cdot n.$

(f) $L_1(\mathbf{x})(Z(\mathbf{x}) - 1) = 0.$

(g)

$$(\mathbf{x}-\mathbf{g}^{n+1})(Z(\mathbf{x})(\gamma+f(\mathbf{x}))(\gamma+t(\mathbf{x})) = (\mathbf{x}-\mathbf{g}^{n+1})\cdot Z(\mathbf{g}\cdot\mathbf{x})(\gamma+h_1(\mathbf{x}))(\gamma+h_2(\mathbf{x})))$$

(h) $L_{n+1}(\mathbf{x})(Z(\mathbf{x}) - 1) \equiv 0.$

and outputs acc iff all checks hold.

Remark 5.1. *Note that if we are in a situation where the sorted values of f alone already have a consecutive gap of at most c, we can use a simpler protocol showing a permutation between f and s without needing to use t.*

Lemma 5.2. *Fix positive integer c. Suppose that $\left\{f(\mathbf{g}^i)\right\}_{i\in[n]}$ is not contained in $\{0,\ldots,c(n-1)\}$. Then for any strategy of $\mathcal{A}$ playing the role of $\mathbf{P}$ in the above protocol $\mathscr{P}$, the probability that $\mathbf{V}$ accepts is $\mathsf{negl}(\lambda)$. Furthermore, for $c \geq 2$ we have $\mathfrak{d}(\mathscr{P}) = (3+c)n + 2$.*

Proof. We compute $\mathfrak{d}(\mathscr{P})$: As in the previous protocol $\mathbf{P}$ sends $Z, h_1, h_2 \in \mathbb{F}_{<n+1}[X]$, which aggregates to $3n+3$. Now, for $c \geq 2$ the highest degree constraints are the check $P\left(h_1(\mathbf{g}\cdot X) - h_1(X)\right) \equiv 0$ (and the same check for h_2). Which gives degree $(c+1)\cdot n$ to which we add one and subtract $|H| = n+1$; totalling in $3n+3+(c+1)\cdot n-(n+1) = (3+c)n+2$.

Now for the main claim, assume that for some $i \in [n]$, $f_i \notin \{0,\ldots,cn\}$. The checks in steps 6a-6e imply that the values $\left\{h_1(\mathbf{g}^i), h_2(\mathbf{g}^i)\right\}_{i\in[n]}$ are all in the range $\{0,\ldots,c(n-1)\}$. Define polynomials

$$F(X) := \prod_{i\in[n]}(X - f(\mathbf{g}^i))(X - t(\mathbf{g}^i)), G(X) := \prod_{i\in[n]}(X - h_1(\mathbf{g}^i))(X - h_2(\mathbf{g}^i)))$$

Under the assumption that for some $i \in [n]$, $f_i \notin \{0,\ldots,cn\}$, we have that F and G are distinct polynomials.

The checks in steps 6f-6h imply that $F(\gamma) = G(\gamma)$.

Since $\gamma \in \mathbb{F}$ is chosen uniformly it follows that $\mathbf{V}$ accepts with probability $\mathsf{negl}(\lambda)$.

□

Acknowledgements

We thank Jens Groth for introducing us to the lookup protocol of [BCG$^+$]. We thank Kevaundray Wedderburn for comments and corrections and the name plookup.

References

[AAB+19] A. Aly, T. Ashur, E. Ben-Sasson, S. Dhooghe, and A. Szepieniec. Efficient symmetric primitives for advanced cryptographic protocols (A marvellous contribution). *IACR Cryptology ePrint Archive*, 2019:426, 2019.

[ACG+19] M. R. Albrecht, C. Cid, L. Grassi, D. Khovratovich, R. Lüftenegger, C. Rechberger, and M. Schofnegger. Algebraic cryptanalysis of stark-friendly designs: Application to marvellous and mimc. In *Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part III*, pages 371–397, 2019.

[AGR+16] M. R. Albrecht, L. Grassi, C. Rechberger, A. Roy, and T. Tiessen. Mimc: Efficient encryption and cryptographic hashing with minimal multiplicative complexity. In *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I*, pages 191–219, 2016.

[BCG+] J. Bootle, A. Cerulli, J. Groth, S. K. Jakobsen, and M. Maller. Arya: Nearly linear-time zero-knowledge proofs for correct program execution. In *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part I*, volume 11272 of *Lecture Notes in Computer Science*, pages 595–626.

[GKK+19] L. Grassi, D.l Kales, D. Khovratovich, A. Roy, C. Rechberger, and M. Schofnegger. Starkad and poseidon: New hash functions for zero knowledge proof systems. *IACR Cryptology ePrint Archive*, 2019:458, 2019.

[GWC19] A. Gabizon, Z. J. Williamson, and O. Ciobotaru. PLONK: permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. *IACR Cryptology ePrint Archive*, 2019:953, 2019.

[KZG10] A. Kate, G. M. Zaverucha, and I. Goldberg. Constant-size commitments to polynomials and their applications. pages 177–194, 2010.