A Novel Power-Sum PRG with Applications to Lattice-Based zkSNARKs

Charanjit S. Jutla* Eamonn W. Postlethwaite† Arnab Roy‡

December 4, 2023

Abstract

zkSNARK is a cryptographic primitive that allows a prover to prove to a resource constrained verifier, that it has indeed performed a specified non-deterministic computation correctly, while hiding private witnesses. In this work we focus on lattice based zkSNARK, as this serves two important design goals. Firstly, we get post-quantum zkSNARK schemes with $O(\log(\text{Circuit size}))$ sized proofs (without random oracles) and secondly, the easy verifier circuit allows further bootstrapping by arbitrary (zk)SNARK schemes that offer additional or complementary properties. However, this goal comes with considerable challenges. The only known lattice-based bilinear maps are obtained using multi-linear maps of Garg, Gentry, and Halevi 2013 (GGH13), which have undergone considerable cryptanalytic attacks, in particular annihilation attacks.

In this work, we propose a (level-2) GGH13-encoding based zkSNARK which we show to be secure in the weak-multilinear map model of Miles-Sahai-Zhandry assuming a novel pseudo-random generator (PRG). We argue that the new PRG assumption is plausible based on the well-studied Newton's identity on power-sum polynomials, as well as an analysis of hardness of computing Grobner bases for these polynomials. The particular PRG is designed for efficient implementation of the zkSNARK.

Technically, we leverage the 2-linear instantiation of the GGH13 graded encoding scheme to provide us with an analogue of bilinear maps and adapt the Groth16 (Groth, Eurocrypt 2016) protocol, although with considerable technical advances in design and proof. The protocol is non-interactive in the CRS model.

Keywords— zero-knowledge, succinct proof, weak multilinear model, linear interactive proofs, elimination theory, Grobner basis, Newton's Identity.

*IBM T. J. Watson Research Center, NY, USA

†CWI, Cryptology Group, Amsterdam, The Netherlands

‡Mysten Labs, CA, USA

Contents

1 Introduction

Verifiable Computation (VC) protocols allow a computationally bounded client to outsource the computation of an expensive function on some given inputs to a more powerful worker in such a way that provides assurances about the validity of the outputs returned by the worker. Regardless of the method used to provide these assurances, it should require less work on the part of some verifier to enact than to simply evaluate the outsourced function on a given input.

Groth16 [Gro16] is a scheme that achieves, for certain tasks, this notion of efficiency on the part of a verifier, as well as some attractive theoretical properties. In Groth16 the worker, given a function, and an input and evaluation key for said function, supplies a proof of computation. The aforementioned attractive properties include that Groth16 is public verifier, meaning any party in possession of public information can verify such a proof of computation, and that the proof can be made zero knowledge, in the sense that the verifier can verify the computation without learning anything about potential inputs of the worker. On the asymptotic efficiency side, these properties include that the proof of computation has constant size, regardless of the function, that the setup cost is linear in the cost of evaluating the function (so that the client may amortize this over many inputs given to the worker), and that verification has cost linear in the number of inputs and outputs of the function.

The Groth16 proofs themselves rely on a characterization of NP called Quadratic Arithmetic Programs (QAP) [GGPR12], and on encoding these using bilinear maps. From a technical standpoint, the assumption on which Groth16 bases its security (which roughly equates to the worker not being able to create a proof that passes verification for an incorrect output) is a generic bilinear group model, which is stronger than discrete log type assumptions on elliptic curve groups. However, discrete log type assumptions are susceptible to polynomial time quantum attacks. A natural question then arises – can a similar scheme to Groth16 be built that relies on hard problems that are currently conjectured to be post-quantum? One way to answer this in the affirmative would be to build a similar scheme from something approximating a bilinear map based on the (quantum) hardness of lattice type problems. The work of [GGH13] provided such approximate maps. However, these have been subjected to attacks such as annihilation attacks [MSZ16]. The paper also provides a (generic) weak multilinear map model which captures such attacks. In a later work, [GMM+16] give an obfuscation (iO) scheme that resists annihilation attacks based on the existence of PRFs in NC^1, and proven secure in the weak multilinear map mpodel. While, recently [JLS21] show iO without the generic model and instead using well-founded assumptions, the scheme can best be described as a feasibility result.

In this work, we give a first verifiable computation and zkSNARK protocol using post-quantum techniques (without random oracle), namely lattice-based multilinear maps, and with proof size logarithmic in circuit size[1]. We prove the scheme secure in the weak multilinear model assuming the security of a novel PRG. The size of the proof is $O(\log|C|)$ for circuits C. The spirit in which the novel PRG is employed in similar to [GMM+16] in that the PRG is not actually used in the scheme, but rather used to prove that annihilation attacks do not work on the multilinear setting of the scheme.

In more detail, we first describe how Groth [Gro16] adapts the linear interactive proof (LIP) methodology of [BCI+13] to the generic bilinear group model. In the LIP methodology, one first proves security of a base scheme against affine adversaries. This base scheme can then be lifted to a scheme secure against efficient adversaries by encoding the public quantities in the "exponent" of a generator(s) of a bilinear group to form a common reference string (CRS). In lattice based schemes one can use GGH13 encodings. But, since these encodings are susceptible to annihilation attacks, one must now upgrade the generic bilinear group model to the weak multilinear map model. In other words, while in the elliptic-curve bilinear-group setting it was considered safe to prove that the CRS was disclosure-free in the generic bilinear-group model, one must now show that it is disclosure-free in an enhanced sense, namely that efficient annihilation of ring elements corresponding to level-two zeroes is also considered a fatal disclosure.

In the weak multilinear map model, the level-two zeroes are also considered to be efficiently obtainable only in a generic manner. However, since the generic adversary knows the algebraic structure, the potential

[1]We actually obtain a proof size that is $O(\log^*|C|)$ since our scheme has a very simple verification circuit, which can then be recursively treated as a verifiable circuit problem.

exists that the adversary can annihilate these algebraic expressions (more precisely, in our scheme, multivariate polynomials). We prove that in the generic model, the only level-two zeroes that an adversary can obtain must have a particular structure which can be modeled as the output of an efficient multivariate candidate pseudo-random generator (PRG). A careful design of the scheme has led to this candidate PRG to be a set of secret-weighted symmetric power-sum of secret variables. In particular, this is in contrast to the Goldreich-PRG of set of low-degree low-weight polynomials [Gol11].

Consider a finite field $\mathbb{F}$ and integer parameters m, n. Define the function f from $\mathbb{F}^{m+n*m}$ to $\mathbb{F}^{n(n-1)/2}$ as

$$f(x_1, ..., x_m, s_{1,1}, ..., s_{n,m})_{(i1,i2)} = \sum_{j=1}^{m} x_j^{i1} * s_{i2,j} + x_j^{i2} * s_{i1,j},$$

where the degrees $i1, i2$ range from one to n and $i1 \neq i2$. Now, let λ be a security parameter. Let $\mathbb{F}$ be a field with at least 2^λ elements, The candidate pseudo-random generator is given by function f with m at least $\sqrt{\lambda}$ and $n > m$.

Since, symmetric power-sums and elementary symmetric polynomials have been extensively studied in mathematics and the famous Newton identity annihilates symmetric power-sums, the candidate PRG above uses secret and linear-weighted symmetric power-sums which thwarts this line of attack and we conjecture that annihilation of the the weighted expressions would require generic Grobner-basis elimination techniques. We also show that the above parameters have been conservatively chosen to thwart generic Grobner-basis attacks,

Related Works. While zk-SNARKs have been built from lattices before [BISW17, GMNO18, BISW18], the verification algorithm was not public. For example, Gennaro et al [GMNO18] built a lattice zk-SNARK based on Square Span Programs, but it was designated verifier; they encoded using Regev-style encryption and the verifier needed the secret key to decode and perform multiplications.

Several works [Kil92, Mic00] can be built from lattices, but they are significantly inefficient and based on the Random Oracle Model. Works based on the Interactive Oracle Proofs [BCS16] framework have a dependence on circuit size and also rely on the Random Oracle Model for non-interactivity. For example, Aurora [BCR+19] has $O(log^2|C|)$ size for circuit C.

There has been progress on basing VC schemes on falsifiable assumptions, e.g. [KPY19], but there are fundamental barriers to achieve the same for zkSNARKs [GW11]. Our scheme achieves zero-knowledge in a public-verifier public-prover setting and hence the reliance on non-standard non-falsifiable lattice assumptions seems justified.

A recent work [ACL+22] has built publicly-verifiable, pre-processing and recursively composable lattice-based SNARKs. There are some essential differences between their approach and this paper:

1. Their approach is to first reduce the NP problem to a system of quadratic equations and then essentially give a functional commitment to these set of equations. Whereas, we translate a QAP-based SNARK from a bilinear group setting to a quasi-bilinear lattice setting.
2. We take the more traditional approach of first constructing a core linear interactive proof system, but enhanced with a multilinear annihilation attack model.
3. Our proof size is $O(log|C|)$ with competitive constant factors, whereas their's is $O(polylog|C|)$ with constants hard to estimate due to the existence of some recent attacks [CLM23]. [ACL+22] does have verification run-time that is $O(log|C|)$ (which is same as our runtime). More details of our proof size can be found in Section 7, including the potential of using other complementary schemes for recursive proofs.

2 Preliminaries

Notations. The inner product on $\mathbb{R}^n$ is defined as $\langle \cdot, \cdot \rangle : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}, (x, y) \mapsto \sum_i x_i y_i$. The Euclidean ($\ell_2$) norm is denoted by $\| \cdot \|$, or when required by context, by $\| \cdot \|_2$. The ℓ_1, ℓ_∞ norms are denoted by

$\|\cdot\|_1, \|\cdot\|_\infty$ respectively.

For $n \in \mathbb{Z}, n \geq 1, [n] = \{1, \ldots, n\}$.For $m \in \mathbb{Z}, m \geq 1, [n]_m \in [-m/2, m/2)$ is the representative element of n modulo m in the given range, as an integer.

2.1 Fields, Rings and Ideals

In this work we will use number fields with no sub-fields other than $\mathbb{Q}$, as well as having large Galois group. One such convenient number field is suggested in [BCLv17], where $K = \mathbb{Q}[X]/(f(X))$, with $f(X)$ of the form $X^a - X - 1$ for prime a. Instead of computing its ring of integers, we will work in the polynomial ring $R = \mathbb{Z}[X]/(f(X))$ (since we do not care about the ring being Dedekind Domain). We will denote $R_q = R = \mathbb{Z}[X]/(q, f(X))$.

It would be preferable to choose a so that for the underlying field $\mathbb{F}_p$, the polynomial $f(X)$ has a large degree irreducible factor. Then by Chinese Remainder Theorem, R_p can be considered to have an extension field of $\mathbb{F}_p$ of the same degree. Using Berlekamp's algorithm it is easy to check that the choice of prime a is quite extensive even for $p = 2$. For example, for $a = 293$, one irreducible factor has degree 279, and for $a = 521$, there is one irreducible factor of degree 385 (both for $p = 2$). We will use φ to denote the isomorphism of R_p and $\mathbb{F}_{p^d} \times R'$, where $f(X)$ has an irreducible factor of multiplicity one of degree d, modulo p. We will abuse notation, and refer to extension field $\mathbb{F}_{p^d}$ as $\mathbb{F}_p^{(f)}$.

2.1.1 Lattices.

A lattice $L \subset \mathbb{R}^n$ is a discrete additive subgroup of $\mathbb{R}^n$. Every (non-trivial) lattice has a basis. A basis for a full-rank lattice is a set of n linearly independent vectors $\{b_1, \ldots, b_n\} \subset \mathbb{R}^n$ such that $L = \{\sum_{i=1}^n z_i b_i \,|\, z_i \in \mathbb{Z}\}$. Let all lattices we consider be full-rank. If we arrange the vectors b_i as the columns of a matrix $B \in \mathbb{R}^{n \times n}$ then we may write $L = \{Bz \mid z \in \mathbb{Z}^n\}$. If B is a basis for L then we say that B generates L.

2.1.2 Geometry and Ideal Lattices.

Embedding K into $\mathbb{R}^n$ defines a geometry on K. In this work we consider the (bijective) coefficent embedding $\iota \colon K \to \mathbb{Q}^n \subset \mathbb{R}^n, \alpha_0 + \alpha_1 X + \cdots + \alpha_{n-1} X^{n-1} \mapsto (\alpha_0, \alpha_1, \ldots, \alpha_{n-1})$. We may therefore talk of the norm of an element of K and write, for some $\alpha \in K$ and $i \in \{1, 2, \infty\}$, $\|\alpha\|_i$ for $\|\iota(\alpha)\|_i$. Note that this is not the number field norm, i.e. the determinant of the multiplication by α map, but is instead the geometric norm of α given the coefficient embedding. We may also refer to and think of an element of K as the vector it forms under ι and vice versa, i.e. of α as $\iota(\alpha)$ and of $\iota(\alpha)$ as α for any $\alpha \in K$. A useful inequality in the power of two cyclotomic case is for any $\alpha, \beta \in K$,

$$\|\alpha\beta\| \leq \sqrt{n}\|\alpha\|\|\beta\|.$$

When considering the norm of some $r' \in R_q$ we consider the norm of the unique $r = r_0 + \cdots + r_{n-1} X^{n-1} \in R$ such that $[r] = r'$ and each $r_i \in [-q/2, q/2)$.

Complex conjugation over K (and similarly R) is the map $\bar{\cdot} \colon K \to K, \alpha_0 + \alpha_1 X + \cdots + \alpha_{n-1} X^{n-1} \mapsto \alpha_0 - \alpha_1 X^{n-1} - \cdots - \alpha_{n-1} X$, that is, it maps $X \mapsto X^{-1} = -X^{n-1}$.

We also consider $K_\mathbb{R} = \mathbb{R}[X]/\langle X^n + 1\rangle$, the topological closure of K, and we extend $\bar{\cdot}$ and ι to $K_\mathbb{R}$ in the natural way. It has a subring $S = \{x \colon x = \bar{x}, x \in K_\mathbb{R}\}$, which has a subset $S^+ = \{x\bar{x} \colon x \in K_\mathbb{R}\}$. Elements in S^+ have exactly one square root in S^+, denoted by the function $\sqrt{\cdot} \colon S^+ \to S^+$. If $A \colon K_\mathbb{R} \to S^+, x \mapsto x\bar{x}$ then for $\Sigma \in S^+, A(\sqrt{\Sigma}) = \Sigma$. In short, the function A is the inverse of $\sqrt{\cdot}$ over S^+. We follow the convention [DP18] of denoting the inverse of $x \in R_q$ as x^{-1} and reserving $1/x$ or $\frac{1}{x}$ for the inverse of x in K or $K_\mathbb{R}$. For $x \in K, x^{-1}$ is shorthand for $[x]^{-1}$, should it exist.

Given $g \in R \setminus \{0\}$ consider the ideal lattice generated by $\mathcal{I} = \langle g \rangle$. To see this object as a lattice, consider the discrete additive subgroup of $\mathbb{R}^n$ it produces under ι. A basis of $\mathcal{I}$ is $B_g = \{g, gX, \ldots, gX^{n-1}\}$.

2.1.3 The Ideal and Field Norms.

Given K and R as defined above, two further norms may be defined. The first, ideal norm, considers ideals $\mathfrak{a} \in R$ and is defined as $N_I(\mathfrak{a}) = [R\colon \mathfrak{a}] = |R/\mathfrak{a}|$, i.e. the number of distinct cosets of the form $r + \mathfrak{a}$. The second, field norm, considers $\alpha \in K$ and can be defined as the determinant of the $\mathbb{Q}$-linear map given by multiplication by α, $\varphi_\alpha\colon K \to K, \beta \mapsto \alpha \cdot \beta$ and is denoted by $N_{K/\mathbb{Q}}(\alpha) = \det(\varphi_\alpha)$. For principal ideals, $\mathfrak{a} = \langle r \rangle, r \in R$, we have $N_I(\mathfrak{a}) = N_{K/\mathbb{Q}}(r)$. Note that the field norm restricted to the ring of integers, R, will always take integer values. Note also that in the case of power of two cyclotomic fields we have $\det(\varphi_\alpha) = \mathrm{res}(X^n + 1, \alpha)$, that is the resultant of the cyclotomic polynomial and α, by e.g. [PP19, Prop. 1]. This norm will have size roughly $\|X^n + 1\|^{n-1} \cdot \|\alpha\|^n$ [SV10].

2.1.4 Lattice Gaussians.

For real $\sigma > 0$, define the (centred) spherical Gaussian function on $\mathbb{R}^n$ with parameter σ as $\rho_\sigma\colon \mathbb{R}^n \to \mathbb{R}, x \mapsto \exp(-\pi\|x\|^2/\sigma^2)$. Given a lattice L define $\rho_\sigma(L) = \sum_{x \in L} \rho_\sigma(x)$ and the discrete spherical Gaussian distribution over L with parameter σ as $D_{L,\sigma} = \rho_\sigma(x)/\rho_\sigma(L)$. This generalises to $K_\mathbb{R}$; let $\Sigma \in S^+$ and define the (again, centred) Gaussian function on $K_\mathbb{R}$ with parameter $\sqrt{\Sigma}$ as

$$\rho_{\sqrt{\Sigma}}\colon K_\mathbb{R} \to \mathbb{R}, x \mapsto \exp\left(-\frac{1}{2}\left\|\frac{x}{\sqrt{\Sigma}}\right\|^2\right).$$

Note that this distribution is not necessarily spherical. For any (potentially fractional) ideal, $\mathcal{I} \lhd K$, and any shift $y \in K_\mathbb{R}$, define $\rho_{\sqrt{\Sigma}}(y + \mathcal{I}) = \sum_{x \in y+\mathcal{I}} \rho_{\sqrt{\Sigma}}(x)$ and the discrete Gaussian distribution over $y + \mathcal{I}$ with parameter $\sqrt{\Sigma}$ as

$$D_{y+\mathcal{I},\sqrt{\Sigma}}\colon y + \mathcal{I} \to \mathbb{R}, x \mapsto \frac{\rho_{\sqrt{\Sigma}}(x)}{\rho_{\sqrt{\Sigma}}(y + \mathcal{I})}.$$

2.2 Arithmetic Circuits and Quadratic Arithmetic Programs (QAPs).

An arithmetic circuit consists of wires that carry values from a field $\mathbb{F}$ and connect to addition, multiplication and multiplication by scalar gates. A Quadratic Arithmetic Program (QAP), an encoding of such a circuit, was defined in [GGPR13] as follows.

Definition 1 (Quadratic Arithmetic Program (QAP)) *A QAP Q over field $\mathbb{F}$ contains three sets of $\mu + 1$ polynomials, $\mathcal{U} = \{u_k(x)\}$, $\mathcal{V} = \{v_k(x)\}$, $\mathcal{W} = \{w_k(x)\}$, for $k \in \{0\} \cup [\mu]$, and a target polynomial $t(x)$, all over $\mathbb{F}$. Suppose F is a function (describing some circuit C) that takes as input n elements of $\mathbb{F}$ and outputs n' elements of $\mathbb{F}$, for a total of $N = n + n'$ I/O elements. We say that Q computes F when: $(c_1, \ldots, c_N) \in \mathbb{F}^N$ is a valid assignment of F's inputs and outputs iff there exist coefficients $c_{N+1}, \ldots, c_\mu$ such that $t(x)$ divides $p(x)$. The polynomial $p(x)$ is defined as*

$$p(x) = \left(u_0(x) + \sum_{k=1}^{\mu} c_k \cdot u_k(x)\right) \cdot \left(v_0(x) + \sum_{k=1}^{\mu} c_k \cdot v_k(x)\right) - \left(w_0(x) + \sum_{k=1}^{\mu} c_k \cdot w_k(x)\right).$$

In other words, there must exist a polynomial $h(x)$ such that $t(x)h(x) = p(x)$. The size of Q is μ and the degree of Q is d, the degree of $t(x)$.

A QAP, in essence, checks that the output of each multiplication gate in the arithmetic circuit is correct (addition and multiplication by scalar gates are compressed into their contributions to multiplication gates).

To do this the target polynomial $t(x) = \prod_{g \in C_{mul}} (x - f_g)$ is formed as the product of $|C_{mul}|$ (so $d = |C_{mul}|$, where C_{mul} are the multiplicative gates of C) linear factors with an arbitrary, pairwise distinct, root f_g for each multiplication gate. The polynomial sets $\mathcal{U}, \mathcal{V}, \mathcal{W}$ are then formed to encode the left inputs, right inputs, and outputs, respectively, of a given multiplication gate. In particular each $k \in [\mu]$ is assosciated to either an input wire of C or an output wire of some multiplication gate. For the root f_g associated to some $g \in C_{mul}$ and the wire assosciated to some $k \in [\mu]$, the polynomial $u_k(f_g) = 0$ when this wire is not a left input to g and $u_k(f_g) = 1$ when this wire is a left input to g. The same is true of the polynomials in $\mathcal{V}, \mathcal{W}$, but for right inputs and outputs, respectively. To form these polynomials consider e.g. Lagrangian interpolation over $\mathbb{F}$, so that each has degree $d - 1$. The form of the polynomial $p(x)$ then exactly encodes multiplication in C, and that $t(x)$ divides $p(x)$ decomposes into d checks – $p(f_g) = 0$ at each $g \in C_{mul}$ – that each multiplication is carried out correctly (see [PHGR13, Sec. 2.1.1] for a useful worked example and [GGPR13, Sec. 7] for omitted technicalities).

2.3 Verifiable Computation (VC).

Following [PHGR13], we define a (public) Verifiable Computation scheme (VC) and the definitions of correctness, security, and efficiency. Such a scheme allows a computationally bounded client to outsource the evaluation of some function F on input u to a more powerful worker. This worker, along with evaluating and returning $F(u)$, should provide a proof so a public party may verify the correctness of the returned value.

Definition 2 *A public verifiable computation scheme consists of three polynomial time algorithms,* (KeyGen, Compute, Verify).

- $(\mathsf{EK}_F, \mathsf{VK}_F) \leftarrow \textsc{KeyGen}(F, 1^\lambda)$. *A randomised algorithm that takes the function to be outsourced, F, and a security parameter λ. It outputs a public evaluation key, EK_F and a public verification key VK_F.*
- $(y, \pi_y) \leftarrow \textsc{Compute}(\mathsf{EK}_F, u)$. *A deterministic worker algorithm that computes and outputs $y = F(u)$ alongside a proof π_y of the correctness of this output.*
- $\{0, 1\} \leftarrow \textsc{Verify}(\mathsf{VK}_F, u, y, \pi_y)$. *A deterministic algorithm that outputs 1 if $y = F(u)$ and 0 otherwise.*

For a given function F the client runs $\textsc{KeyGen}(F, 1^\lambda)$ once and may then have the worker compute several inputs. The cost of creating $\mathsf{EK}_F, \mathsf{VK}_F$ is linear in the function size, i.e. the efficiency comes from the amortized cost of having the worker compute the function on many inputs. Intuitively such a VC scheme is correct if for an honest output $y = F(u)$ and correctly generated proof, the Verify algorithm always outputs 1. It is secure if the probability of any probabilistic polynomial time adversary generating a proof for an incorrect output that passes verification is negligible in λ, and it is efficient if (after the one time cost for the client of KeyGen) the cost of Verify is cheaper than evaluating F. Formally,

- (Correctness.) For any function F and input u, if $(\mathsf{EK}_F, \mathsf{VK}_F) \leftarrow \textsc{KeyGen}(F, 1^\lambda)$ and $(y, \pi_y) \leftarrow \textsc{Compute}(\mathsf{EK}_F, u)$, then $\textsc{Verify}(\mathsf{VK}_F, u, y, \pi_y) = 1$.
- (Security.) For any function F and probabilistic polynomial time adversary $\mathcal{A}$, $\Pr[(\hat{u}, \hat{y}, \hat{\pi}_{\hat{y}}) \leftarrow \mathcal{A}(\mathsf{EK}_F, \mathsf{VK}_F) : \hat{y} \neq F(\hat{u}) \wedge \textsc{Verify}(\mathsf{VK}_F, \hat{u}, \hat{y}, \hat{\pi}_{\hat{y}}) = 1] = \mathsf{negl}(\lambda)$.
- (Efficiency.) For a function F and input u, if $(\mathsf{EK}_F, \mathsf{VK}_F) \leftarrow \textsc{KeyGen}(F, 1^\lambda)$ and $(y, \pi_y) \leftarrow \textsc{Compute}(\mathsf{EK}_F, u)$, then computing $\textsc{Verify}(\mathsf{VK}_F, u, y, \pi_y)$ should be cheaper than evaluating $F(u)$.

In Groth16, QAPs were used to build a VC scheme via encoding the polynomials $v_k(x)$ using bilinear groups (and similarly for $\mathcal{W}, \mathcal{Y}$). In particular some secret $s \in \mathbb{F}$ is sampled uniformly and the group elements $g^{v_k(s)}$ are given out in either EK_F or VK_F (depending on k). Here g is a generator of a subgroup of the bilinear group such that it is of prime order and isomorphic to the additive group of $\mathbb{F}$. After the worker has evaluated $F(u)$ it has learnt all $c_k, k \in [\mu]$ (and in particular the hitherto unknown internal wire values). It can then form $g^{v(s)}$ for $v(s) = v_0(s) + \sum_{k \in [\mu]} c_k v_k(s)$, or similar sums, by standard techniques, e.g. repeated squaring to compute $g^{c_k v_k(s)}$ followed by group multiplication. Given $c_k, k \in [\mu]$ the worker

can also compute $p(x)$ and therefore also $h(x)$. If g^{s^i}, for sufficient powers of s, are also available to the worker, then π_y consists roughly of the elements $\{g^{v(s)}, g^{w(s)}, g^{y(s)}, g^{h(s)}\}$. The verifier uses the bilinear map to check that $p(s) = h(s)t(s)$ and several conditions regarding $v(s), w(s), y(s)$. Intuitively this, along with the "if and only if" property of QAPs in Definition 1, checks the correctness at all internal multiplication gates, and therefore the output $y = F(u)$.

In this work we will adapt Groth16 [Gro16] to use the κ-Graded Encoding Scheme (multilinear map) of [GGH13]. While the adaptation adds some technical barriers to be overcome, multilinear maps have all the basic functionality (addition of encodings, some bounded notion of multiplication, and a form of equality testing, see Section 3) required to emulate this style of VC scheme from QAPs.

3 The (Altered) GGH13 Multilinear Map

Here, we introduce the GGH13 scheme (in its symmetric setting), and its subsequent adaptations, that we use to encode QAPs. Some differences between what is introduced here and the original GGH13 scheme [GGH13] include different parameter choices, different public quantities, and a different method of generating encodings; see below. In particular, the plaintext space in our scheme will be modulo a different prime p than the integer q used for the encoding space. Another difference is our lack of the zero testing parameter h. Our rational for this is that since our scheme will give sufficient information for h to be recovered in quantum polynomial time we should instead discuss (Appendix B) what security we can hope to achieve and enjoy the (slightly) improved efficiency of a scheme without h.

3.1 Graded Encoding Schemes

More formally, a graded encoding scheme (which we call multilinear map) consists of a ring QR and a system of encoding sets $\mathcal{S} = \{S_i^{(\alpha)} \subset \{0,1\}^* : \alpha \in QR, i \in \{0\} \cup [\kappa]\}$, where κ is the multilinearity parameter, and has the following properties.

1. For every fixed index i, the sets $\{S_i^{(\alpha)}\}_{\alpha \in QR}$ are disjoint.
2. There is an associative binary operation, such that for every α_1, α_2, every index i, and every encoding $u_1 \in S_i^{(\alpha_1)}$ and $u_2 \in S_i^{(\alpha_2)}$, it holds that
$$u_1 + u_2 \in S_i^{(\alpha_1+\alpha_2)}.$$
3. There is an associative binary operation $\times$, such that for every α_1, α_2, each pair of indices i_1, i_2 such that $i_1 + i_2 \leq \kappa$, and all encodings $u_1 \in S_i^{(\alpha_1)}$ and $u_2 \in S_i^{(\alpha_2)}$, it holds that
$$u_1 \times u_2 \in S_{i_1+i_2}^{(\alpha_1 \cdot \alpha_2)}.$$

3.2 Altered GGH13 Graded Encoding Scheme

The scheme is parameterised by a security parameter λ and a multilinearity parameter κ. An instance of GGH13 encodes elements of a quotient ring $R/\mathcal{I}$, where $\mathcal{I} = \langle g \rangle \lhd R$ (i.e. principal ideal of R generated by $g \in R$). Thus, this forms the plaintext space in GGH13. The encodings are elements of R_q where q is a large enough prime to support functionality. This q depends on λ, and κ. These encodings can be publicly homomorphically added and multiplied, under some constraints, and with the above properties. It can also be checked whether certain encodings encode the zero of $R/\mathcal{I}$.

Now, we describe how our plaintext space is different from GGH13 (hint: it is similar to the difference between plaintext space and ciphertext space in FHE schemes [BGV12]). Note that the field $\mathbb{F}$, over which function F is defined, will be some prime order field $\mathbb{F}_p$. In Groth16, encodings are in a subgroup of the bilinear group which is isomorphic to the additive group of $\mathbb{F}$, i.e. this subgroup is $\mathbb{Z}_p$ and the field is $\mathbb{F}_p$. Since their bilinear pairing is $\mathbb{Z}_p$-linear, this allows them to have a straightforward way of enabling (one

level of) multiplication in $\mathbb{F}_p$. On the other hand, for security reasons in the lattice setting, and for known graded encoding schemes, the multilinear map is only $\mathbb{Z}_q$-linear where q is much larger than p. However, we make use of the indeterminate X in the polynomial ring X to enable a much larger plaintext space field $\mathbb{F}_p[X]/(f(X))$ for some irreducible polynomial $f(X)$ over $\mathbb{F}_p$. This is assured by $f(X)$ being an irreducible factor of the cyclotomic polynomial $X^n + 1$ modulo p. More details about the parameter settings follow in Section 3.3.

The scheme consists of six polynomial time algorithms, ($\mathsf{InstGen}, \mathsf{enc}, \mathsf{add}$, $\mathsf{neg}, \mathsf{mul}, \mathsf{isZero}$) given below. Parameter choices will be made explicit after these algorithms are introduced.

3.2.1 Instance Generation.

$(\mathsf{sk}, \mathsf{params}) \leftarrow \mathsf{InstGen}(1^\lambda, 1^\kappa, p)$. An integer $n = 2^u, u \in \mathbb{N}$, modulus q, and $g \in R$ are chosen, subject to some constraints. In particular $g \leftarrow D_{\mathbb{Z}^n,\sigma}$ for some σ, and $\mathcal{I} = \langle g \rangle$. A uniform invertible element $z \in R_q^\times$ is sampled and the Gaussian parameters $\{\Sigma_i\}_{i\in[\kappa]} \subset S^+$ for forming encodings are set. Finally, no element $h \leftarrow D_{\mathbb{Z}^n,\sqrt{q}}$ is sampled and instead the "zero tester" is formed as $p_{zt} = z^\kappa g^{-1}$. Note z^κ and g^{-1} are all elements of R_q, so $p_{zt} \in R_q$. The public parameters are $\mathsf{params} = \{n, q, \kappa, p_{zt}, \{\Sigma_i\}_{i\in[\kappa]}\}$ and the secret key is $\mathsf{sk} = \{g, z\}$.

3.2.2 Encoding.

$u \leftarrow \mathsf{enc}(\mathsf{sk}, \mathsf{params}, a, i)$. The scheme encodes cosets $a + \mathcal{I} \in R/\mathcal{I}$ at a level $i \in [\kappa]$. An encoding of $a + \mathcal{I}$ at level i is

$$u = [a + rg]z^{-i} \in R_q,$$

with $a + rg$ a short element in $a + \mathcal{I}$. Throughout we refer to an encoding of $a + \mathcal{I} \in R/\mathcal{I}$ as an encoding of $a \in R$. Such an encoding is formed by sampling from $D_{a+\mathcal{I},\sqrt{\Sigma_i}}$, for $\Sigma_i \in S^+$ determining the distribution to sample from at level i. These encodings are called fresh, as opposed to encodings formed from Add or Mul. In particular, unlike in GGH13, we do not give out level 1 encodings of zero nor add randomised combinations of these to a specific representative element of the coset $a + \mathcal{I}$.

The quantity $a + rg \in R$ will be referred to as the numerator of the encoding. It will be sampled such that its length is much less than q, this says that $\|a + rg\| = \left\|[a + rg]_q\right\|$ which will not be the case if any coefficient of $a + rg$ is larger than $q/2$. The parameter q will be chosen such that a slightly stronger property remains true for any encoding we consider, see Add, Neg, Mul.

The method of creating encodings a priori by sampling from a discrete Gaussian over a coset came about due to attacks on multilinear map constructions making use of low level encodings of zeros [HJ16], in particular the level 1 encodings of zero, x_i, that allowed public parties to create encodings in the original GGH13. The different methods of sampling from [GGH13, DGG+18] were codified in [DP18] in terms of sampling parameters $\Sigma_i \in S^+$, in order to analyse the (statistical) leakage released to an adversary after successful zero tests. A new method, the "compensation" method, was also introduced.

These sampling methods (except for the "simplistic" method of GGH13) all ensure that they are able to efficiently sample from a distribution negligibly close to $D_{y+\mathcal{I},\sqrt{\Sigma_i}}$ via the techniques of [GPV08], see [DP18, Thm. 2]. That is, the encodings themselves are independent of the secrets of the scheme.

3.2.3 Addition, Negation, and Multiplication.

$u \leftarrow \mathsf{add}(u_1, u_2)$. If $u_i = [a_i + r_i g]z^{-j}$ encodes a_i at level $j \in [\kappa]$, then $u_1 + u_2$ encodes $a_1 + a_2$ at level j.

Furthermore q is chosen such that

$$\Big[[a_1 + r_1 g]_q + [a_2 + r_2 g]_q\Big]_q = [(a_1 + a_2) + (r_1 + r_2)g]_q,$$

that is, there is no reduction modulo q. Therefore the lifts of $[a_1 + r_1 g]+[a_2 + r_2 g]$ and $[(a_1 + a_2) + (r_1 + r_2)g]$ back to R (coefficients in $[-q/2, q/2)$) have the same length. The length of the numerator of $u_1 + u_2$ is at most twice that of the longer numerator of u_1, u_2.

$-u \leftarrow \mathsf{neg}(u)$. If $u = [a + rg]z^{-j}$ encodes a at level $j \in [\kappa]$, then $-1_R \cdot u$ encodes $-a$ at level j. As q will be odd, $\left[[-1_R]_q [a + rg]_q\right]_q = [-1_R(a + rg)]_q$, and the length of the numerator does not change.

$u \leftarrow \mathsf{mul}(u_1, u_2)$. If $u_1 = [a_1+r_1 g]z^{-j}, u_2 = [a_2+r_2 g]z^{-k}$ encode a_1, a_2 at levels j, k, respectively, for $j, k \in [\kappa], j + k \leq \kappa$, then $u_1 u_2$ encodes $a_1 a_2$ at level jk. Again, q is chosen such that $\left[[a_1 + r_1 g]_q [a_2 + r_2 g]_q\right]_q = [(a_1 + r_1 g)(a_2 + r_2 g)]_q$ and in this case the length of the numerator of $u_1 u_2$ is at most $\sqrt{n}$ times the square of the length of the longer numerator of u_1, u_2.

Note that NEG multiplies an encoding at level j by an element of R. In general, multiplying u which encodes a at level j by $b \in R$ produces an encoding of ab at level j with the numerator multiplied by b. We make use of this in our construction, multiplying by elements of R with coefficients of size up to p. We also want this operation to have the property that the numerator does not reduce modulo q, hence the dependence q will have on p in our construction.

3.2.4 Zero Testing.

$\{0, 1\} \leftarrow \mathsf{isZero}(\mathsf{params}, u)$. The algorithm ISZERO tests whether an encoding u at level κ is an encoding of 0. This is decided as

$$\mathsf{isZero}(\mathsf{params}, u) = \begin{cases} 1 & \text{if } \|[p_{zt} u]_q\|_\infty < q^{3/4}, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

If u encodes zero then $a \in \mathcal{I}$ and therefore $a = r'g$, so $u = [a + rg]z^{-\kappa} = [(r' + r)g]z^{-\kappa}$. In this case, letting $r'' = r + r'$, $p_{zt} u = [r'']$ will have length $\Theta(\mathsf{poly}(n, p))$ and we set q a little larger than that which ensures there are no false negatives. If a does not encode zero, it can be shown (via an adaptation of [GGH13, Lemma 4]) that our choice of q also prevents false positives – this requires that $\mathcal{I}$ be a prime ideal. Note that we can check whether two level κ encodings encode the same coset using ADD, NEG, and ISZERO.

3.3 Parameters Selection

In this section we consider choosing a number field with no sub-fields and very large extensions that become Galois...e.g. as suggested by [BCLv17]: $f(x) = x^a - x - 1$, where a is a prime, in which case the Galois group is known to be of size $p!$. Further, we choose a so that modulo prime p, $f(x)$ has an irreducible factor of larger enough degree as described below.

Throughout M is the maximum length of an encoding under ℓ_∞ for the given use case (see Section 7 for our use case and M).

- Dimension $n \in \mathbb{N}$ is chosen $n = \Omega(\kappa\lambda \log q)$ to achieve λ bit security against lattice attacks. Note [CDPR16], that does the short PIP, only works for power-of prime cyclotomics. Further, [DP18] shows attacks on NTRU-style encodings, which also works on fields with sub-fields, which our number field does not have. For this work κ will be constant, so we omit it. The choice of e is given below.
- Multilinearity parameter $\kappa > 0$. We set $\kappa = 2$.
- Gaussian parameter $\sigma \in R_{\geq 0}$. This parameter for a spherical Gaussian over $\mathbb{Z}^n$ is set as $\sigma = O(\sqrt{n})$.
- Ideal $\mathcal{I}$ generator $g \in R$. Sampled as $g \leftarrow D_{\mathbb{Z}^n, \sigma}$, so $\|g\| = \Theta(n)$ with overwhelming probability [DGG+16, Lemma. D.5]. Resampled until g is prime , $\|1/g\| = O(n^2)$ and $g^{-1} \in R_q$ exists.
- Gaussian parameter $\Sigma \in S^+$. This parameter is sampled following the "compensation" method of [DP18]. For $\kappa = 2$ it gives numerators, $c = y + rg$ drawn from $D_{y+\mathcal{I}, \sqrt{\Sigma}}$, of length $\|c\| = \Theta(n^{2+\varepsilon})$, $\varepsilon > 0$.[2]

[2] We note that, in the absence of h, the distortion analysis [DP18, Sec. 5] suggests we can achieve $\|c\| = \Theta(n^{7/4+\epsilon})$.

- Prime modulus $p \geq 2$. The modulus of the field over which we consider the circuit. Recall the worker, while more powerful than the client, is still efficient, so $|C| = \mathsf{poly}(\lambda)$. The extension field $\mathbb{F}_p^{(f)}$ should have size large enough so that it is larger than circuit size so as to form the QAP and select our required secret values.
- Modulus q is chosen to be a power of two (for boot-strapping the verification circuit, so that during recursion $p = 2$). Chosen so that isZero remains correct for the longest numerator of a top level zero. It is enough for q to be some polynomial in n, M, see Lemma 3 below. We prove in Section 7 that M is polynomial in n, p, both of which are polynomial in λ, hence q is also polynomial in λ.

Lemma 3 *Let M be the largest numerator, in terms of norm, of level-two encoding of zero considered during verification. If $\mathcal{I}$ is a prime ideal, $q = \Omega(n^8)$, and $q = \Omega(n^{10/3}M^{4/3})$, then* isZero *does not produce false positives or false negatives.*

Proof: First, let u encode zero at level two, so $u = rg * z^{-2} \bmod q$. We see that $\|[p_{zt}u]_q\|_\infty \leq \|[p_{zt}u]_q\| \leq \|g^{-1}rg\| \leq \sqrt{n}\|g^{-1}\|\|rg\| = O(n^{5/2}M)$[3]. To ensure this quantity passes IsZero set $q = \Omega((n^{5/2}M)^{4/3}) = \Omega(n^{10/3}M^{4/3})$. Second, let u encode a non-zero coset at level two, so $u = (a+rg)z^{-2}, a \notin \mathcal{I}$ and assume that $\|p_{zt}u\|_\infty < q^{3/4}$. To appeal to [GGH13, Lemma 3] let $w = p_{zt}u$ and $c = a + rg$. We require both $\|gw\|$ and $\|c\|$ to be less than $q/2$. That is, $\|gw\| \leq \sqrt{n}\|g\|\|w\| \leq n\|g\|\|w\|_\infty \leq n^2q^{3/4}$ must be less than $q/2$, as must $\|c\| \leq M$. This is satisfied when $q = \Omega(n^8)$ and $q = \Omega(M)$, and by the lemma we have $c \in \mathcal{I}$, i.e. $a \in \mathcal{I}$, a contradiction. The $q = \Omega(M)$ condition is redundant, hence the statement. □

For our VC scheme, from end of Section 7, $M = \Omega(\lambda^{1/2}|C|^2n^{7/2+\varepsilon}p^2)$, and therefore the relevant condition on q is really $q = \Omega(n^{10/3}M^{4/3})$. Let $|C| = \lambda^c$, and also assume q is polynomial in λ, say λ^e. Then With $n = \lambda \log q = e\lambda \log \lambda$, and assuming $p < \lambda$, this amounts to $q = \Omega(n^{10/3}M^{4/3})$, which is $q = \Omega(n^{8+\varepsilon}\lambda^{10/3+8c/3}$, which implies $q = \Omega(\lambda^{10/3+8c/3+8+\varepsilon} \cdot (e \log \lambda)^8$. Thus, conservatively, $q = \lambda^{12+8c/3}$. If the verification circuit is boot-strapped, so that $p = 2$, then q drops to $\lambda^{9+8c'/3}$, where $\lambda^{c'}$ is the size of the verification circuit. This has implications for the proof size as detailed towards the end of Section 7.

3.4 Zeroizing and Annihilation Attacks on [GGH13]

Proving the security of constructions based on [GGH13] (and multilinear maps, in general) has been a subtle art. In particular, three main kinds of attacks have been considered: (a) lattice-reduction attacks, (b) short principal ideal generator problem, and (3) zeroizing and annihilation attacks that can lead to discovery of short principal ideal generators. While the attacks (a) and (b) are of a more general nature and dictate parameter selection of the rings and ideals used in the scheme, the zeroizing and annihilation attacks can be scheme dependent, i.e. they can be more effective based on what additional information an Adversary can garner from the protocol (e.g. the CRS or public key). Since, our scheme only uses 2-linear maps, the low level zeroes are then just level-1 zeroes. While it will be easy to show that our scheme only gives out encodings of level-1 zeroes with negligible probability, the annihilation attacks that employ encodings of level-2 zeroes needs a comprehensive analysis. Thus, the focus of our analysis will be on annihilation attacks. We refer the reader to Appendix B for more details about the general attacks (a) and (b), as well as zeroizing attacks and "Statistical" zeroizing attacks [DP18].

In [MSZ16], it was shown that one can obtain annihilators of various top level encodings of zero for the [GGH13] multi-linear map based obfuscation schemes, to obtain elements in the ideal $\langle g \rangle$. We briefly describe the idea of annihilation attack, before we formalize a model for annihilation attack(as in [MSZ16]) and then further restrict it to a (new) generic model of annihilation attacks. Our brief description will focus only on attacks that target level-1 and level-2 encodings.

As described in Section 3, each level-1 encoding (of m) is of the form $(m + rg)z^{-1}$ in the ring R_q, where r is a randomizer. In particular, all the level-1 encodings, say $N1$ in number, given to the Adversary (e.g. as part of a public key) can be written as $e1_i = (m_i + r_ig)z^{-1}$ (for $i \in [1..N1]$). Thus, each encoding has

[3] $\|rg\| \leq M$ and $\|g^{-1}\| \leq n^2$

its own independent variable r_i. The aim of the annihilation attack is to use 2-linear pairings of level-1 encodings to get level-2 encodings of zero. Such level-2 encodings of zero (say, of the form $(r'g + r''g^2)z^{-2}$) can then be multiplied by $p_{zt} = z^2/g$ to cancel out g/z^2, and obtain $r' + r''g$, which if both r' and r'' are small, is obtained in the base polynomial ring R. Now, if further linear combination of such quantities can even annihilate r', we are then left with quantities of the form $r'''g$. This would be a small multiple of the secret g, which could potentially be used to find g. Hence, the goal of the Adversary in annihilation attacks is to take linear combinations as above to annihilate r' terms.

3.4.1 Miles-Sahai-Zhandry Attack Model

We now describe the annihilation attack model from in [MSZ16].

There are "hidden" variables $s_1, ..., s_n$ (for some integer n, and collectively denoted by $\vec{s}$) and $r_1, ..., r_m$ (for another integer m and collectively denoted by $\vec{r}$), and g. Then there are "public" variables $y_1, ..., y_m$ (denoted $\vec{y}$), which are set to $y_i = q_i(\vec{s}) + gr_i$ for some fixed but public polynomials $q_i(\vec{s})$ defined over a ring R. Thus, y_i is set to polynomials in $R[\vec{s}, \vec{r}, g]$. The adversary is allowed to make two types of queries:

- In a **Type 1** query, the adversary submits a "valid" polynomial p_k in $R[\vec{y}]$. Here "valid" polynomials come from some restricted set of polynomials. These restrictions are those that are enforceable using graded encodings, e.g. some asymmetry requirements. Next, we consider p_k as a polynomial of the formal variables $\vec{s}, \vec{r}, g$, given the definition of $\vec{y}$ above. Write $p_k = p_k^{(0)}(\vec{s}, \vec{r}) + gp_k^{(1)}(\vec{s}, \vec{r}) + \langle g^2 \rangle$, where $\langle g^2 \rangle$ is the ideal of $R[\vec{s}, \vec{r}, g]$ generated by g^2. If p_k is identically 0, then the adversary receives $\perp$ in return. If $p_k^{(0)}$ is not identically 0, then the adversary receives $\perp$ in return. If p_k is not 0, but $p_k^{(0)}$ is identically 0, then the adversary receives a handle to a new variable w_k, which is set to be $p_k/g \bmod \langle g \rangle = p_k^{(1)}(\vec{s}, \vec{r})$ (a polynomial in $R[\vec{s}, \vec{r}]$).

- In a **Type 2** query, the adversary is allowed to submit arbitrary polynomials a as a small algebraic circuits on the w_k that it has seen so far. Consider $a(\vec{w})$ as a polynomial of the variables $\vec{s}, \vec{r}$ by using the definition of w_k above, and write a as $a^{(0)}(\vec{s}, \vec{r})$. If $a^{(0)}$ is identically zero, then the model responds with 0, i.e. the adversary wins. Otherwise, the model responds with 1.

In other words, the adversary wins if it can find an annihilator polynomial $a(\vec{w})$ of the set of polynomials $\{p_k^{(1)}(\vec{s}, \vec{r})\}$. The annihilator polynomial is defined formally in the next sub-section.

4 A New Generic Model for Computing Annihilators

While the previous section modeled generic annihilation attacks in graded encoding schemes, the question arises as to whether we can restrict the adversary further to be of a generic type when computing an annihilator (as in Type 2 queries above). Instead of a complexity lower bounds, which are normally not known so far, we can try to model the various algorithms known to compute annihilators, and restrict the adversary to such algorithms. It turns out that all known algorithms, at least for set of polynomials that do not show a particular structure, follow the same principal of Grobner Basis computation. This in turn implies a generic complexity bound that can be given in terms of number of multi-variate polynomials, the number of variables and the degree of the polynomials [CKPS00]. This is akin to the generic group model [**?**], in the sense that the adversary is not allowed to look at the coefficients defining the polynomials. In other words, the adversary literally only looks at the polynomials as handles w_k instead of $p_k^{(1)}(\vec{s}, \vec{r})$ (see previous section). The actual generic model we consider below is slightly stronger in the sense that the adversary is allowed to consider the monomials that occur in the given polynomials (i.e. the ones with non-zero coefficients). To this end, we first start with a lemma that characterizes annihilators as a projection of an ideal.

Consider a set $\mathcal{F}$ of polynomials in $R[\vec{x}]$, $\mathcal{F} = \{f_1(\vec{x}), ...f_m(\vec{x})\}$. A polynomial $a(y_1, ...y_m)$ over R is called an annihilator of $\mathcal{F}$, if $a(f_1(\vec{x}), ..., f_m(\vec{x}))$ is identically zero.

Denoting the set of variables $y_1, ..., y_m$ by $\vec{y}$, consider the ideal I of the ring $R[\vec{x}, \vec{y}]$ generated by $\mathcal{F}^* = \{y_1 - f_1(\vec{x}), ..., y_m - f_m(\vec{x})\}$. The following lemma is well-known (e.g. Prop. 15.30 of [**?**]), but we give a simple proof here for completeness.

Lemma 4 *The ideal J of $R[\vec{y}]$, defined as $J = I \cap R[\vec{y}]$, is exactly the set of annihilators of $\mathcal{F}$.*

Proof: It is easy to check that J is an ideal of $R[\vec{y}]$.

We first show that if $a(\vec{y})$ is in J, then $a(\vec{y})$ is an annihilator of $\mathcal{F}$. By virtue of being in $R[\vec{y}]$, $a(\vec{y})$ is by definition a polynomial in $\vec{y}$ over R. Since $a(\vec{y})$ is also in I, by definition of an ideal, $a(\vec{y})$ is a linear combination of $\mathcal{F}^*$, with coefficients from the ring $R[\vec{x}, \vec{y}]$. But each of the elements of $\mathcal{F}^*$ is zero with $f_i(\vec{x})$ substituted for y_i. Hence, $a(\vec{y})$ with $f_i(\vec{x})$ substituted for y_i is also zero.

Next, we show that if some $a(\vec{y})$ in $R[\vec{y}]$ is an annihilator of $\mathcal{F}$, then $a(\vec{y})$ is in the ideal J. Since, $a(f_1(\vec{x}), ..., f_m(\vec{x}))$ is identically zero, we will just show that $b(\vec{x}, \vec{y})$ defined as $a(f_1(\vec{x}), ..., f_m(\vec{x})) - a(\vec{y})$ is in I (and, since it is in $R[\vec{y}]$ it is also in J).

View a as an arithmetic circuit. We show by induction that if $a_1(\vec{y}) - a_1(\vec{f}(\vec{x}))$ and $a_2(\vec{y}) - a_2(\vec{f}(\vec{x}))$ are both in I, then so are $(ua_1 + va_2)(\vec{y}) - (ua_1 + va_2)(\vec{f}(\vec{x}))$ and $a_1a_2(\vec{y}) - a_1a_2(\vec{f}(\vec{x}))$. The addition case is trivial. The multiplication case is also straightforward as follows:

$$a_1a_2(\vec{y}) - a_1a_2(\vec{f}(\vec{x})) = a_1(\vec{y})(a_2(\vec{y}) - a_2(\vec{f}(\vec{x}))) + a_2(\vec{f}(\vec{x}))(a_1(\vec{y}) - a_1(\vec{f}(\vec{x})))$$

This concludes the proof. □

The next question is whether there are efficient algorithms to compute an element of J. We will now review [**?**] the Grobner-basis approach to finding such an element; in fact we will compute the (Grobner-) basis of J. The aim is to highlight the generic nature of such a computation, in absence of any special structure of the given generators of the ideal. As the above lemma shows, we are interested in eliminating the $\vec{x}$ variables, and since the polynomial ring is not a field, we cannot resort to Gaussian elimination. But, could something similar work?

For an ideal F of a polynomial ring S, the monomials of F are all possible monomials that occur in any element of F. A **monomial ideal** is an ideal of S generated only by monomials. For example, the ideal generated by $(x^2 + y)$ is not a monomial order, where as, the ideal generated by (x^2, y) is a monomial ideal. While monomial ideals are easy to analyze, e.g. the quotient S/J for a monomial ideal J is easy to characterize, it is remarkable that general ideals can also be handled in a similar fashion with some machinery.

Definition 5 *Let F be an ideal of a polynomial ring S over a field k. A* **monomial order** *of F is a total order $>$ on the monomials of F such that if m_1, m_2 are monomials of F and $n \neq 1$ is a monomial of S, then*

$$m_1 > m_2 \text{ implies } nm_1 > nm_2 > m_2$$

Having defined a monomial order, we can define an initial term of every polynomial f in F. Then, it would be interesting if the ideal generated by the initial terms of F, which would be a monomial ideal, would allow similar characterization of S/F, i.e. as if F itself was a monomial ideal. With the above requirements on the monomial order $>$, for any $f \in F$, we define the **initial term** of f, written $in_>(f)$, to be the greatest monomial in f with respect to $>$ (here, we ignore the scalar multiples from field k). If M is any sub-ideal of F, we define $in_>(M)$ to be the monomial sub-ideal generated by the elements in $in_>(f)$ for all $f \in M$.

Definition 6 *A* **Grobner basis** *with respect to an order $>$ on an ideal F is a set of elements $g_1, ..., g_t \in F$ such that if M is the sub-ideal of F generated by these elements, then $in_>(g_1), ..., in_>(g_t)$ generate $in_>(M)$. We then say that $g_1, ..., g_t$ is a* **Grobner basis for** *M.*

There is a Grobner basis for any sub-ideal of M of F, for one can start with a finite set of generators of M, and then keep adding finitely many more elements of M to the set till their initial terms generate $in_>(M)$, the latter being finitely generated as well.

Now we state a lemma from [**?**] that embodies variable elimination. We first define a monomial order which allows elimination. An order on $T = S[\vec{x}] = k[\vec{y}, \vec{x}]$ is called an **elimination order** (w.r.t. $\vec{x}$) if it satisfies: for every f in T such that the initial term of f, $in_>(f)$, is in S, it is the case that f is itself in S. In other words, monomials purely in $\vec{y}$ are less than every monomial which has some degree from $\vec{x}$. A lexicographic monomial order where $\vec{x}$ variables are given higher precedence than $\vec{y}$ variables is an example of elimination order.

Lemma 7 *Let $>$ be a monomial order on $T = S[\vec{x}] = k[\vec{y}, \vec{x}]$, and suppose that $>$ is an elimination order with respect to the variables $\vec{x}$. If $I \subset T$ is an ideal, then with respect to the monomial order on S obtained by restricting $>$, we have*

$$in_>(I \cap S) \;=\; in_>(I) \cap S$$

Further, if $g_1, ..., g_t$ is a Grobner basis for I, and $g_1, ..., g_u$ are those g_i that do not involve the variables x_i, then $g_1, ..., g_u$ form a Grobner basis in S for $J = I \cap S$.

Thus, by Lemma 4, it follows that a basis for the ideal of annihilators of $\mathcal{F}$ is a Grobner basis B of I, with elimination-order w.r.t. $\vec{x}$, restricted to the subset of generators in B that are purely in $\vec{y}$.

There exists an algorithm to compute Grobner Basis due to Rechberger [**?**]. Further, many variants of this algorithm are known which can be potentially more efficient depending on the given ideal and the monomial order. The main idea of the algorithm is to start with any basis of the ideal, and considering any two generators in this basis, say g_1 and g_2, compute a new generator

$$g = g_2 * in_>(g_1)/\text{gcd}(in_>(g_1), in_>(g_2)) - g_1 * in_>(g_2)/\text{gcd}(in_>(g_1), in_>(g_2)).$$

By definition of $in_>$, the initial term of g is strictly less than the initial term of both g_1 and g_2. If g can be written as sum of many $f_i g_i$, with each f_i in S, and $in_>(g) \geq in_>(f_i g_i)$ for all i, then we stop, otherwise we add g to the basis and repeat. Note, termination is guaranteed as the process is well-founded.

Since we seek to eliminate $\vec{x}$, and hence use an elimination order where $\vec{x}$ are given higher precedence than $\vec{y}$, the above elimination process would start with generators having initial terms that are purely composed of $\vec{x}$, and hence the new generator "g" would be obtained by multiplication of g_1 and g_2 by monomials in $\vec{x}$ (see above). Subsequently, however, the monomials being multiplied in to get "g" may contain both $\vec{x}$ and $\vec{y}$, and eventually possibly just $\vec{y}$. Moreover, if the starting basis of ideal M is not a monomial basis, and is of a more generic nature, i.e. has generators with many monomials whose structure can not be exploited, the above process is captured by the following Macaulay matrix. Assume that the given basis of M has generators with maximum degree d. We can homogenize the generators so that all monomials in all the generators are of degree d by introducing a new variable, say z, and assigning z the lowest precedence in the lexicographic elimination order. The Macaulay matrix of the given basis of M at level $d+t$ is then obtained by multiplying each of the generators with every possible monomial of degree t (in the variables $\vec{x}, \vec{y}, z$), and then writing each such newly obtained polynomial's coefficients as a row in the Macaulay matrix (coefficients of monomials of degree $d+t$).

If the monomials are arranged according to the elimination order, with the highest precedence monomials on top, i.e. all monomials containing at least one degree from $\vec{x}$ on top, then the k-vector space J which is the ideal M restricted to $\vec{y}, z$ and degrees restricted to $d+t$, is generated by linear combination of rows of the Macaulay matrix with the coefficients of all $\vec{x}$ involving monomials zero. This basis of J is easily obtained by Gaussian elimination. The assumption that there is only a generic way to obtain a non-tivial element in J then amounts to there being more rows in the Macaulay matrix than there are $\vec{x}$ involving monomials of total degree $d+t$.

To analyze this further, to start with, if there are m polynomials in n variables $\vec{x}$ of maximum degree d in the set F, the ideal I is then generated by m homogeneous polynomials in $\vec{x}, \vec{y}, z$ of degree d. Then, a generic elimination of $\vec{x}$ variables requires a Macaulay matrix at level at least $d+t$ such that m times the number of monomials of degree t is more than the number of monomials of degree $d+t$ (involving at least one degree from $\vec{x}$). This leads to the following inequality

$$m * \binom{m+n+1+t-1}{t} \;\geq\; \binom{m+n+1+d+t-1}{d+t} - \binom{m+1+d+t-1}{d+t} \tag{2}$$

Since, the generators $\mathcal{F}^*$ of I only have simple degree one $\vec{y}$ monomials, an efficient annihilation computation may not necessarily be generic w.r.t. $\vec{y}$, and may only be generic w.r.t. $\vec{x}$ variables. However, even for any fixed monomial m' in $\vec{y}, z$, the number of equations obtained by multiplying I by all monomials m such that $\mathsf{m} = \mathsf{m}' * \mathsf{m}''$, with m'' a monomial in $\vec{x}$ (of degree t), must exceed the total number of monomials m of the form $\mathsf{m} = \mathsf{m}' * \mathsf{m}'''$ with m''' a monomial in $\vec{x}$ of degree $t + d$. This is so because otherwise these monomials containing $\vec{x}$ degrees cannot be eliminated. Thus, the following inequality must still be satisfied

$$m * \binom{n+t-1}{t} \geq \binom{n+d+t-1}{d+t} \tag{3}$$

As already shown by [CKPS00], this is satisfied only if $d + t \geq \frac{n}{m^{1/d}}$. However, this model is weak since even though the coefficients of polynomials of $\mathcal{F}$ may be generic, the monomials in $\mathcal{F}$ may not be generic. For example, all the monomials could be high degree in a single variable, and low total degree in all other variables. We strengthen the generic model in the next sub-section.

4.1 Direct Generic Method to Compute an Annihilator

Since, we seek only one annihilator and not the whole set of generators of the ideal of annihilators, the above methodology based on Macaulay matrix suggests a direct generic method (instead of defining ideal I generated by $\mathcal{F}^*$ and then computing its Grobner basis w.r.t. an elimination order). Treat the monomials in $\mathcal{F} = \{f_1(\vec{x}), ..., f_m(\vec{x})\}$ as new variables $W = \{w_1, ..., w_m\}$. The number of monomials in W of degree t is $\binom{m+t-1}{t}$. Let M_t be number of monomials in $\vec{x}$ in any such degree t polynomial viewed as a polynomial over $\vec{x}$ (by substituting $f_i(\vec{x})$ for w_i). Thus, this leads to an annihilator if $\binom{m+t-1}{t} > M_t$. The generic model then assumes that this inequality must also hold to find an annihilator.

Generic Annihilation Assumption (Informal) Given a set of polynomials $\mathcal{F} = \{f_1(\vec{x}), ..., f_m(\vec{x})\}$ in n variable $\vec{x}$, let t^* be the smallest t such that M_t (defined above) is less than $\binom{m+t-1}{t}$. Then, the time to compute an annihilator of $\mathcal{F}$ is at least $\Omega(M_{t^*})$.

A more formal statement in terms of ensemble of polynomials $\{\mathcal{F}\}$ will be given in the full version of the paper.

5 The Groth16 SNARK

We now describe the pairing-based NIZK argument for quadratic arithmetic programs given by Groth [Gro16]. Consider a relation R over the field $\mathbb{F}_p$, defined by the QAP $(\ell, \{u_i(X), v_i(X), w_i(X)\}_{i=0}^m, t(X))$, with $log(p) = poly(\lambda)$. The relation defines a language of statements $(a_1, \cdots, a_\ell) \in \mathbb{F}_p^\ell$ and witnesses $(a_{\ell+1}, ..., a_m) \in \mathbb{F}_p^{m-\ell}$ such that with $a_0 = 1$:

$$\sum_{i=0}^{m} a_i u_i(X) \cdot \sum_{i=0}^{m} a_i v_i(X) = \sum_{i=0}^{m} a_i w_i(X) + h(X)t(X),$$

for some degree $n-2$ quotient polynomial $h(X)$.

A bilinear group system $(\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_T, e, g, h)$ is generated with an $\mathbb{F}_p$-bilinear operator $e : \mathcal{G}_1 \times \mathcal{G}_2 \to \mathcal{G}_T$ and generators g and h for $\mathcal{G}_1$ and $\mathcal{G}_2$ respectively. Next, the (Setup, Prove, Vfy, Sim) algorithms are described as below. The notations $[x]_1, [y]_2$ denote $g^x \in \mathcal{G}_1, h^y \in \mathcal{G}_2$ respectively.

Setup$(R) \to (\sigma, \tau)$: Pick $\alpha, \beta, \gamma, \delta, x \leftarrow \mathbb{Z}_p^*$. Define $\tau = (\alpha, \beta, \gamma, \delta, x)$ and compute $\sigma = ([\sigma_1]_1, [\sigma_2]_2)$, where:

$$\sigma_1 = \begin{pmatrix} \alpha, \beta, \delta, \{x^i\}_{i=0}^{n-1}, \left\{\frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\gamma}\right\}_{i=0}^{\ell}, \\ \left\{\frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\delta}\right\}_{i=\ell+1}^{m}, \left\{\frac{x^i t(x)}{\delta}\right\}_{i=0}^{n-2} \end{pmatrix}, \quad \sigma_2 = \left(\beta, \gamma, \delta, \{x^i\}_{i=0}^{n-1}\right).$$

$\mathbf{Prove}(R, \sigma, a_1, \cdots, a_m) \to \pi:$ Pick $r, s \leftarrow \mathbb{Z}_p$ and compute $\pi = ([A]_1, [C]_1, [B]_2)$, where

$$A = \alpha + \sum_{i=0}^{m} a_i u_i(x) + r\delta \qquad B = \beta + \sum_{i=0}^{m} a_i v_i(x) + s\delta$$

$$C = \frac{\sum_{i=\ell+1}^{m} a_i(\beta u_i(x) + \alpha v_i(x) + w_i(x)) + h(x)t(x)}{\delta} + As + Br - rs\delta$$

$\mathbf{Vfy}(R, \sigma, a_1, \cdots, a_\ell, \pi):$ Check if:

$$[A]_1 \cdot [B]_2 = [\alpha]_1 \cdot [\beta]_2 + \sum_{i=0}^{\ell} a_i \left[\frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\gamma}\right]_1 \cdot [\gamma]_2 + [C]_1 \cdot [\delta]_2$$

$\mathbf{Sim}(R, \tau, a_1, \cdots, a_\ell):$ Pick $A, B \leftarrow \mathbb{Z}_p$ and compute $\pi = ([A]_1, [C]_1, [B]_2)$ with:

$$C = \frac{AB - \alpha\beta - \sum_{i=0}^{\ell} a_i(\beta u_i(x) + \alpha v_i(x) + w_i(x))}{\delta}$$

5.1 Applicability (of Graded Encodings) to Groth16 SNARK

Once we encode elements with our multilinear map we have reduction modulo $f(X)$ and modulo q, but lose reduction modulo (the much smaller) p. That is, even if the numerators are equal modulo p, they can represent different cosets of R_q. We therefore want both to think of elements of our QAP modulo $f(X)$ and to account for this lost reduction modulo p.

5.1.1 QAPs over Extension Fields and R_p.

We build our QAP modulo $f(X)$ via the Chinese Remainder Theorem, as described in Section 2.1. The circuit takes wire values in $\mathbb{F}_p$ and the QAP is first built over $\mathbb{F}_p^{(f)}$. Note that in this case the wire values c'_k will be the constant polynomials of the extension field, and these are represented directly as c'_k in R_q.

5.1.2 Altering CRS.

To account for the lost reduction modulo p we note that any equivalence modulo p can be expressed as an equivalence over the integers with a correction factor divisible by p. Thus, let $\mathcal{U}(x) = u_0(x) + \sum_{k\in[\mu]} c_k u_k(x) \in R[x]$ and similarly for $\mathcal{V}(x), \mathcal{W}(x)$. Note that, from Definition 1 of a QAP

$$\mathcal{U}(x)\mathcal{V}(x) - \mathcal{W}(x) = h(x)t(x) \bmod p. \tag{4}$$

The worker can therefore calculate various $r_i(x) \in R[x]$ with coefficients in $[-(p-1), p-1]$ such that the following equation is exact over $R[x]$,

$$\mathcal{U}(x)\mathcal{V}(x) - \mathcal{W}(x) = h(x)t(x) + \sum_{i=1}^{\eta} p^i \cdot r_i(x), \tag{5}$$

for $\eta < \log_p \left(3p^4(1+\mu)^2 nd\right) = O(\lambda)$.

Hence for any $s \in R_p$ we have equality in the following over R,

$$\mathcal{U}(s)\mathcal{V}(s) - \mathcal{W}(s) = h(s)t(s) + \sum_{i=1}^{\eta} p^i \cdot r_i(s). \tag{6}$$

We therefore must give the worker the ability to communicate (as encodings) the $r_i(s)$ required to make (5) equal over R. The $r_i(x)$ are polynomials over R of degree $2d-2$. This results in more powers s^j being encoded and included in CRS.

Finally, based on these changes we must calculate the largest numerator required to pass isZero and set q accordingly. For the full description, see Section 7.

5.2 Pairings Based Annihilation Attack

We now show a annihilation attack on the above natural extension of Groth16. Moreover, this attack can be demonstrated in the generic annihilation attack model of [MSZ16] described in Section 3.4.1.

Each level-1 encoding (of s) is of the form $(s+rg)z^{-1}$ in the ring R_p, where r is known as the randomizer. In particular, all the level-1 encodings, say $N1$ in number, given as part of the public key can be written as $e_i^{(1)} = (s_i + r_i g)z^{-1}$ (for $i \in [1..N1]$). Thus, each encoding has its own independent variable r_i. The aim of the annihilation attack is to use 2-linear pairings of level-1 encodings (along with given level-2 encodings) to get level-2 encodings of zero. Such level-2 encodings of zero (say, of the form $(r'g + r''g^2)z^{-2}$) can then be multiplied by $h = z^2/g$ to cancel out g/z^2, and obtain $r' + r''g$, which if both r' and r'' are small, is obtained in the base polynomial ring R. Thus, the r' correspond to the handles w_k obtained by Type 1 queries in Section 3.4.1. Now, if further algebraic combination of such quantities can even annihilate r', we are then left with quantities of the form $r'''g$. This algebraic combination is the polynomial $a(\vec{w}$ of Section 3.4.1. This would be a small multiple of the secret g, which could potentially be used to find g. Hence, the goal of the Adversary in annihilation attacks is to take algebraic combinations to annihilate r' terms.

Now, looking at the form of r' terms, we note that these were obtained either by pairings of level-1 encodings or directly from given level-2 encodings. In the former case, r' has the form $s_i r_j + s_j r_i$ (when obtained from pairing of $e_i^{(1)}$ and $e_j^{(1)}$).

Let's say a level-2 encoding of zero is obtained by

$$\sum_{k \in [N2]} b_k e_k^{(2)} + \sum_{k1,k2 \in [N1]} a_{k1,k2} * e_{k1}^{(1)} e_{k2}^{(1)},$$

the first sum ranging over given level-2 encodings (of quantities S_k) and the second sum ranging over level-1 encodings (of s_{k1} and s_{k2} resp.). This requires that

$$\sum_k b_k S_k + \sum_{k1,k2} a_{k1,k2} s_{k1} s_{k2} = 0 \tag{7}$$

Suppose we get N linearly-independent level-2 encodings of zero. By linear-independence it is meant that the coefficients of g (i.e. r') in these level-2 encodings can be viewed as belonging to the number field $Q[X]/(X^n + 1)$. Note that r' in any of these N level-2 encodings of zero has the form

$$\sum_k b_k r_k + \sum_{k1,k2} a_{k1,k2}(m_{k1} r_{k2} + m_{k2} r_{k1})$$

At this point, it is best to characterize the Q-vector space of the coefficients of g (i.e. r') more rigorously. While, the encoding randomness r_{k1} etc. can be treated as independent variables, the expressions m_{k1} must be treated more carefully, as these are rational functions of many secret quantities. In fact, these are polynomial expressions in $\delta, \gamma, \alpha, \beta, \alpha', \beta', x, y$, except for terms δ^{-1} and γ^{-1}.

Going back to upper bounding N above, we first note that the quantities m_{k1} that are encoded in the CRS are $\delta, \gamma, \alpha, \beta, \alpha', \beta', u_i(x), u_i(y), \theta v_i(x), \theta' v_i(y)$, and various other expression in encodings $\mathbf{e}_c$, $\mathbf{e}_d$ and $\mathbf{e}_t$. We can view each encoded expression m_{k1} as a fresh variable, or treat the expressions as polynomials over x, y and other Greek symbols. In the latter case the resulting equations (in terms of x, y etc.) will have a high degree, and the annihilation heuristic would fail as the number of equations N is at most quadratic in the number of encodings. Thus, we can focus on the number of (quadratic) equations of the form (7) that can be obtained in variables that are of the form m_{k1}. A naive calculation would imply that the kernel would be of size zero as there are at most $\binom{m}{2} + m$ pairings of the m encodings, and there are m variables (one for each encoding), and hence $\binom{m}{2} + m$ quadratic monomials. However, some of these monomials can be same, e.g. $x^i x^j$ is same as $x^{i'} x^{j'}$ if $i + j = i' + j'$. Thus, if there are N less total monomials than $\binom{m}{2} + m$, then there is a possibility of N independent quadratic equations.

However, the way we choose the exponents of x (and also y) in the various encodings (i.e. as described in Section A), N is at most sub-quadratic. We remark that even though in the encodings $\mathbf{e}_t$, encodings for

all powers are given (and not just the subset as from Section A), any pairing of $(\theta x^i t(x) + \theta' y^i t(y))$ with $(\theta x^j t(x) + \theta' y^j t(y))$ would yield terms of the type $x^i y^j$, and thus do not yield a contribution to the kernel as there are precisely m^2 different terms $x^i y^j$ (which is the same as the number of such pairings). This is the main reason our construction had to introduce the additional variable y [4]. Hence, by the generic annihilation model, the Adversary only has a super-polynomial time complexity attack.

Referring to requirement in inequality (3), it was observed there that this inequality is satisfied only if

$$d + t \geq \frac{n}{m^{1/d}},$$

where m is the number of degree d polynomials in n variables, and t is the additional degrees required in the Macaulay matrix. In other words, just the total number of monomials required would be $\binom{n+d+t+1}{d+t}$.

6 New Candidate Multivariate PRGs

Consider a finite field $\mathbb{F}$ and integer parameters m, n. Define the function f from $\mathbb{F}^{m+n*m}$ to $\mathbb{F}^{n(n-1)/2}$ as

$$f(x_1, ..., x_m, s_{1,1}, ..., s_{n,m})_{(i1,i2)} = \sum_{j=1}^{m} x_j^{i1} * s_{i2,j} + x_j^{i2} * s_{i1,j},$$

where the degrees $i1, i2$ range from one to n and $i1 \neq i2$.

Now, let λ be a security parameter. Let $\mathbb{F}$ be a field with at least 2^λ elements, The candidate pseudo-random generator is given by function f with m at least $\sqrt{\lambda}$ and $n > m$.

A more efficient candidate version of the PRG requires the s variables to be in $\{0, 1\}$. In this case the PRF is a function from $m * \log|\mathbb{F}| + m * n$ bits to $n(n-1)/2 * \log|\mathbb{F}|$ bits. So, the field size can technically be chosen as small as λ elements. In this work, we will not use this more efficient candidate, and it is described here only for future applications.

6.1 Cryptanalysis

Since f is defined over a finite field $\mathbb{F}$, any non-negligible probability correlation (over the uniform random choice of the input) amongst the $n(n-1)/2 * \lambda$ output bits is most likely to be a generic, i.e. an annihilator of the output (multivariate) polynomials $T_{i1,i2}$ (for $i1 \neq i2$)) defined by

$$T_{i1,i2} = \sum_{j=1}^{m} x_j^{i1} * s_{i2,j} + x_j^{i2} * s_{i1,j}$$

We first consider a simpler set of polynomials, which are effectively built from symmetric power sum polynomials, i.e. symmetric power sums of variables $x_1, ..., x_m$ linearly combined using variables $s_1, ..., s_n$:

$$\begin{aligned} T^*_{i1,i2} &= \sum_{j=1}^{m} x_j^{i1} * s_{i2} + x_j^{i2} * s_{i1} \\ &= (s_{i2} * \sum_{j=1}^{m} x_j^{i1}) + (s_{i1} * \sum_{j=1}^{m} x_j^{i2}) \\ &= s_{i2} * p_{i1}(\vec{x}) + s_{i1} * p_{i2}(\vec{x}), \end{aligned}$$

where $p_k(\vec{x})$ is the usual power-sum symmetric polynomial of degree k in the m variables $\vec{x}$. Note that the number of s variables have been reduced to n, as opposed to $n * m$ in the definition of polynomials T.

[4] It is easy to check that if there was no y variable in the encodings $\mathbf{e}_t$, we would get $O(m^2)$ equations by such pairings of different components of $\mathbf{e}_t$.

Also, recall the elementary symmetric polynomials $e_k(\vec{x})$:

$$e_k(\vec{x}) = \sum_{1 \le j_1 < j_2 < \ldots < j_k \le m} x_{j_1} x_{j_2} \ldots x_{j_k}$$

Note that e_k for $k > m$ is defined to be zero.

We have the famous Newton's identity between the elementary symmetric polynomials and the power-sum symmetric polynomials, in terms of matrix determinant as follows:

$$e_k = \frac{1}{k!} \begin{vmatrix} p_1 & 1 & 0 & \ldots & \\ p_2 & p_1 & 2 & 0 & \ldots \\ \vdots & & \ddots & \ddots & \\ p_{k-1} & p_{k-2} & \ldots & p_1 & k-1 \\ p_k & p_{k-1} & \ldots & p_2 & p_1 \end{vmatrix}$$

Now, since $e_{m+1}(\vec{x})$ is zero, the above identity gives an annihilator of power-sum symmetric functions of degree up to and including $(m+1)$. However, if one requires homogeneous annihilation, i.e. an annihilation polynomial that has all monomials of the same degree, then the above identity is not useful. Indeed, the degree one s-variables in each monomial of the functions $T^*_{i1,i2}$ force any annihilation of $T^*_{i1,i2}$ to be a homogeneous annihilation of the power-sum polynomials and hence the above identity does not work on T^*. Since the subject of power-sum symmetric polynomials is well-studied, to the best of our knowledge this is the only possibility of getting an annihilation using known identities. In view of this, the only alternative to get an annihilation is the general annihilation model attack of Section 4.1.

6.2 Generic Annihilation of polynomials T^* and T

6.2.1 Easy Annihilation of T^*

We first investigate generic-annihilation of the simpler T^*, (with $i1 \neq i2$). In this case, an adversary may just introduce new variables p_i for power-sum of degree i, and we are then left with $n(n-1)/2$ functions T^* in $2 * n$ variables, i.e. n s-variables and n p-variables. Moreover, an adversary may restrict itself to consider only the first $d < n$ degrees, i.e. $i1, i2 \le d$, and this d can be much less than $m = \sqrt{\lambda}$.

Now, the number of monomials of degree t in $T^*_{i1,i2}$ (with $i1, i2 < d$, $i1 \neq i2$) is $\binom{d(d-1)/2+t-1}{t}$. The number M_t of monomials (in s and p variables) in any such degree t monomials is at most $\binom{d+t-1}{t}^2$. Thus, there will be an annihilator polynomial if

$$\binom{d(d-1)/2+t-1}{t} > \binom{d+t-1}{t}^2$$

It is not difficult to see that this is satisfied for $t = 4$ and a small constant d, leading to constant time annihilation of T^*.

6.2.2 Generic Annihilation of T is hard

As opposed to the polynomials in T^*, the polynomials in T involve more s variables, i.e. $n * m$ in number. Although, just as for T^*, an adversary may try to get annihilation for polynomials restricted to degrees d or less (for $d \le n$).

Case 1: First let $d \le 2 * m$. Recall, each polynomial in T, say $T_{i1,i2}$ with $i1, i2 \le d$, now involve all m x-variables, and they cannot be bundled together into a single p-variable as in the case of polynomials T^*. Thus, while the total number of monomials of degree t in the functions T remain $\binom{d(d-1)/2+t-1}{t}$, the total number of monomials M_t (in s and x variables) will be much larger than in the case of T^* above. To estimate a lower bound for M_t, in this Case 1, we ignore the x-variables and just count the number of monomials in

s-variables. The total number of s-variables is $d*m$. And hence, M_t is at least $\binom{d*m+t-1}{t}$. For, $\binom{d(d-1)/2+t-1}{t}$ to exceed this bound on M_t, it must hold that $d(d-1)/2 > d*m$, and thus, $d > 2*m$. Hence, there is no generic attack if $d \leq 2*m$.

Case 2: The case $d > 2*m > 2*\sqrt{\lambda}$ and $t < m^2 < d*m/2$. To get a lower bound on M_t, we restrict the monomials to a subset of all potential monomials in x and s variables in the count M_t as follows: each s variable may have degree at most one in any monomial, and for each such monomial (in the s-variables) each x variable, say x_i may have degree as high as $d*t$. Since, high degrees in a single variable such as x_i may force the s-variable monomials to be of a restricted form, we will only allow an x variable a degree of at most d. Thus, this restricted set of monomials (that contribute to M_t) is at least $\binom{d*m}{t} * d^t$. Thus, to get an annihilator, the adversary would need

$$\binom{d(d-1)/2+t-1}{t} > \binom{d*m}{t} * d^t$$

Since $d > 2*m > 2*t$, the above is satisfied approximately at

$$(d^2/t)^t > (d*m/t)^t * d^t,$$

or for no such value of t.

Case 3: $t > m^2 > \lambda$. At this point, just the number of monomials in the T variables is exponential in λ, and hence the annihilation is not expected in any lesser time.

7 Enhanced Groth16 zkSNARK

Let η be as in Section 5.1, and the VC scheme follow Definition 2.3. Throughout $\mathsf{add}, \mathsf{neg}, \mathsf{mul}$ on encodings are replaced with the conventional symbols, $+, -, \cdot$.

- $\text{CRS} \leftarrow \text{KEYGEN}(F, 1^\lambda)$. Let F be a function over $\mathbb{F}_p$ with ℓ I/O wires. Convert F into an arithmetic circuit C and build the corresponding QAP $Q = (t(x), \mathcal{U}, \mathcal{V}, \mathcal{W})$ of size m and degree d over R_p as described after Definition 1, except that the roots of $t(x)$ are now chosen as powers of a primitive element. Recall $d = |C_{mul}|$. Let $I_{\text{mid}} = \{\ell+1, \ldots, m\}$, i.e. the non I/O indices. Let $(\mathsf{InstGen}, \mathsf{enc}, \mathsf{isZero})$ be algorithms from an encoding scheme (see Section 3). We run $\mathsf{InstGen}(1^\lambda, 1^\kappa, p)$ to obtain $\mathsf{params} = (N, q, \kappa, p_{zt}, \{\Sigma_i\}_{i\in[\kappa]})$ and $\mathsf{sk} = (g, z)$. We will let $n = 2d$ (with no confusion with degree of the defining polynomial of ring R, which is denoted by N for this section).

  Let $T \subseteq [0..3n^2]$ be the "sum collision-free" subset as defined in Section A. Recall $|T| = n$, and let $\psi : [0..n-1] \to T$ be the sum-collision-free mapping. The polynomials $u_i(x), v_i(x), w_i(x)$ can still be defined with the same semantics as before but by utilizing only the monomials x^j, with $j \in T$ as follows: since each gate g now have values f_g associated with it that is power of a common value (see above), the resulting matrix that needs to be inverted to obtain each $u_i(x)$ is still a Vandermonde matrix, which is well known to be invertible.

  Let λ' be a parameter defined from the security parameter λ (e.g. $\lambda' = \sqrt{\lambda}$ as required in Section 6). Choose uniform δ', γ', and $\alpha'_j, \beta'_j, \theta'_j, x'_j \leftarrow \mathbb{F}_p^{(f)}$, for each $j \in [0..\lambda'-1]$, and define $\delta = \varphi^{-1}(\delta', 0), \gamma = \varphi^{-1}(\gamma', 0)$, and further define $\alpha_j = \varphi^{-1}(\alpha'_j, 0), \ldots, x_j = \varphi^{-1}(x'_j, 0)\}$. Recall φ is the isomorphism underlying the CRT decomposition of ring $\mathbb{Z}[X]/(f(X))$. In the below, all encodings are level 1 (or level-2, when stated) using params, so we suppress these in enc. Construct public CRS as:

  - $e_\delta = \mathsf{enc}(\delta)$, $e_\gamma = \mathsf{enc}(\gamma)$,
  - $\mathbf{e}_\alpha = \{(e_{\alpha,j} =) \mathsf{enc}(\alpha_j)\}_{j\in[0..\lambda'-1]}$, $\mathbf{e}_\beta = \{(e_{\beta,j} =) \mathsf{enc}(\beta_j)\}_{j\in[0..\lambda'-1]}$,

- $\mathbf{e}_x = \{(e_{x,i,j} =) \mathsf{enc}(x_j^{\psi(i)})\}_{i\in[0..n-1],j\in[0..\lambda'-1]}$,
- $\mathbf{e}_w = \{(e_{w,i,j} =) \mathsf{enc}(\theta_j x_j^{\psi(i)})\}_{i\in[0..n-1],j\in[0..\lambda'-1]}$,
- $\mathbf{e}_d = \left\{(e_{d,i} =) \mathsf{enc}\left(\frac{\sum_j \beta_j u_i(x_j)+\alpha_j\theta_j v_i(x_j)+\theta_j w_i(x_j)}{\gamma}\right)\right\}_{i\in[0,\ell]}$,
- $\mathbf{e}_c = \left\{(e_{c,i} =) \mathsf{enc}\left(\frac{\sum_j \beta_j u_i(x_j)+\alpha_j\theta_j v_i(x_j)+\theta_j w_i(x_j)}{\delta}\right)\right\}_{i\in[\ell+1,m]}$,
- $\mathbf{e}_t = \left\{(e_{t,i} =) \mathsf{enc}\left(\frac{\sum_j \theta_j x_j^i t(x_j)}{\delta}\right)\right\}_{i\in[0,2(n^2-1)]}$
- $\left\{e_{p,i,k} = \text{Level-2-}\mathsf{enc}\left(p^i(\sum_j \theta_j x_j^k)\right)\right\}_{i\in[0,\eta-1],k\in[0,2(n^2-1)]}$
- $\Phi = \text{Level-2-}\mathsf{enc}\left(\sum_j \alpha_j\beta_j\right)$

- $(y,\pi_y) \leftarrow$ COMPUTE(CRS, u). On input u the worker evaluates the circuit for F to obtain $y = F(u)$ and the wire values $\{c'_k\}_{k\in I_{\text{mid}}}$ in $\mathbb{F}_p \subset \mathbb{F}_p^{(f)}$. From this obtain $c_k = \varphi^{-1}(c'_k, c'_k) \in R_p$. Note that base field entries map via this isomorphism to constant polynomials in R_p. It solves for $h(x) \in R_p[x]$. It then solves for $r_i(x) \in R[x], i \in [\eta]$ (using Equation 5) and lets $r_i(x) = r_{i,0} + \cdots + r_{i,2(n^2-1)}x^{2(n^2-1)}$, $r_{i,j} \in R$ with coefficients in $[-(p-1), p-1]$.

  Let $\bar{e}_{x,i,j} = \sum_{i'=0}^{n-1} u_{i,i'} e_{x,i',j}$, where $u_{i,i'}$ is the coefficient of $x^{\psi(i')}$ in $u_i(x)$, and let $\bar{e}_{w,i,j} = \sum_{i'=0}^{n-1} v_{i,i'} e_{w,i',j}$, where $v_{i,i'}$ is the coefficient of $x^{\psi(i')}$ in $v_i(x)$, (note, $e_{w,i',j}$ has a θ factor). Let $\bar{e}_{y,i,j} = \sum_{i'=0}^{n-1} u_{i,i'} e_{y,i',j}$, and let $\bar{e}_{z,i,j} = \sum_{i'=0}^{n-1} v_{i,i'} e_{z,i',j}$ (note, $e_{z,i',j}$ has a θ' factor).

  Let $\mathbf{c}$ be the vector of coefficients $\{c_k\}$. Pick the quantities $\{r_j^*, s_j^*\}_j$ randomly and independently from $\mathbb{F}_p^{(f)}$.

  So, let the proof π_y be computed as:

  - $A_j = e_{\alpha,j} + r_j^* e_\delta + \mathbf{c}^\top \cdot \bar{e}_{x,\cdot,j}$, for $j \in [0..\lambda'-1]$,
  - $B_j = e_{\beta,j} + s_j^* e_\delta + \mathbf{c}^\top \cdot \bar{e}_{w,\cdot,j}$, for $j \in [0..\lambda'-1]$,
  - $C = \sum_j \left(s_j^* A_j + r_j^* B_j - (r_j^* s_j^*) e_\delta\right) + \sum_{i=\ell+1}^{m} c_i e_{c,i} + \sum_{i=0}^{2(n^2-1)} h_i e_{t,i}$
  - $P = \sum_{i=1}^{\eta} \sum_{k=0}^{2(n^2-1)} r_{i,k} e_{p,i-1,k}$

- $\{0,1\} \leftarrow$ VERIFY(CRS, u, y, π_y). The verification of an alleged proof with elements $\{A_j, B_j\}_{j\in[0..\lambda'-1]}, C, P$ uses the public CRS CRS. Using elements from CRS compute $V_{\text{io}} = \sum_{i=0}^{\ell} \tilde{c}_i e_{d,i}$ and check the following, which is effectively Equation 6.

$$\mathsf{isZero}\left(\mathsf{params}, \quad C \cdot e_\delta + V_{\text{io}} \cdot e_\gamma - \sum_j A_j \cdot B_j + \Phi - pP\right). \tag{8}$$

- $\pi_y \leftarrow Sim(\tau, u, y)$. Sample A_j, B_j, P' from the Gaussian distribution $\mathcal{D}_{p*\mathcal{I}_g, p*\sigma}$.
  Compute $v_{io} = \sum_{i=0}^{\ell} \tilde{c}_i \sum_j (\beta_j u_i(x_j) + \alpha_j\theta v_i(x_j) + \theta_j w_i(x_j))$ and $c = \delta^{-1}(-v_{io} + \sum_j A_j B_j - \alpha_j\beta_j + pP')$.
  Output simulated proof $\pi_y = (\mathsf{enc}(A_j), \mathsf{enc}(B_j), \mathsf{enc}(c), \mathsf{enc}(p'))$.

The correctness of the protocol is standard, but we need to bound the quantity M as required in lemma 3. This is the largest numerator obtained of a level-two zero above during verification. Since, the encodings due to the compensation method (see Section **??**) have size $n^{7/4+\epsilon}$, the multiplication by scalars of the encodings can cause M to go up. These come from $\{r_j^*, s_j^*\}_j$ chosen randomly from $\mathbb{F}_p^{(f)}$. Their quadratic contribution in C, on pairing, cancels out in correct verification, so the main contribution to M is from pairing-sum of A_j and B_j which has the term $\mathbf{c}^\top \cdot \bar{e}_{x,\cdot,j}$. Each component of $\mathbf{c}$ is itself in $\mathbb{F}_p$ (and not in the extension field) This leads to an upper bound of

$$M < \lambda' * \left(p * \max(n,d) * n^{7/4+\epsilon}\right)^2$$

The soundness proof is given in Section 7.1 and 7.2 and is based on the generic model given in Section 4 and the candidiate PRG assumption of Section 6. The statistical zero-knowledge proof follows from the smoothing lemma of [MR04].

Proof Size For a circuit C defined over $\mathbb{F}_p$, the proof size above is $2 \cdot \lambda'$ encodings in R_q. From the discussion after lemma 3, $q = \lambda^{12+8c/3}$, where $|C| = \lambda^c$. Thus the proof size is $\lambda^{1/2} \cdot n \cdot \log q \leq \lambda^{3/2} \log^2 q \leq \lambda^{3/2} * (12 + 8/3 * \log|C|/\log\lambda) \log\lambda \leq \lambda^{3/2} * (12 \log\lambda + 8/3 * \log|C|) = O(\log|C|)$. More concretely, since the verification circuit is linear in $\mathbb{F}_2$ (because q is power of two), one can require the prover to just prove that the verification equation holds for the above verification equation treated as a boolean circuit, i.e. with $p = 2$, and with witness the above proof components. Recall, from Section **??**, for $p = 2$, q drops to $\lambda^{9+8c'/3}$, where $\lambda^{c'}$ is the size of the verification circuit. Let us estimate the size of the above verification circuit treated as a boolean circuit. The above verification requires pairing, i.e. multiplication of two $\mathcal{R}_q$ encodings or polynomials, followed by a zero test, which itself requires multiplication by the zero-testing parameter, another element in $\mathcal{R}_q$. For polynomial multiplication we can use Number Theory Transforms, but since the underlying defining polynomial is not cyclotomic, this requires a factor four blowup in degree. Thus, we require $4 * n * \log n$ multiplications of integers of size at most $2 * \log q$ (possibly using residue-number-system). However, we can require the prover to provide the NTT version of the encodings. Thus the verification circuit size can be estimated to be $O(16 * \lambda' * n * \log q \log\log q$, which is about $16 * \lambda^{3/2} \log^2 \lambda)$. Thus, c' can be conservatively taken to be two. Thus, after one recursion, q can be taken to be λ^{15}, and hence the final proof size is $225 * \lambda^{3/2} \log^2 \lambda$. For $\lambda = 128$, this amounts to a proof size of $225 * (128)^{3/2} * 50 \approx 16$Mbits.

One can also employ Random Oracle based schemes recursively, e.g. Aurora [BCR+19], that have proof size $O(\log^2|C|)$ but with small constants in the big-O notation. Since our verification circuit has size λ^2, this can lead to a rather short sized proof, albeit in a Random-Oracle based scheme.

7.1 Proof Sketch of Soundness

In the next section we establish that generic annihilation attacks do not work against our construction. This is a property that just the CRS needs to provide, as the CRS setup phase is the only phase where the ideal g is used and short vectors are sampled. The prover and verifier algorithms just use these public parameters to derive other elements, but do not produce any "fresh" encodings. In other words, annihilation resistance is an intrinsic property of the CRS. What remains now is to prove the VC security property, namely soundness, which we proceed to do now. Soundness is proved in the generic bilinear group model, just as in Groth16, although the proof is much more intricate. We establish this with a sequence of lemmas.

Lemma 8 *The set of polynomials $\mathcal{W}$ consisting of all polynomials $\{w_i(x)\}_{i\in[l+1..m]}$ and $\{x^i t(x)\}_{i\in[0..3n^2-1]}$ are linearly independent over the field $\mathbb{F}_p^{(f)}$.*

Proof: Let $c_i, d_j \in \mathbb{F}_p^{(f)}$, $i \in [l+1..m], j \in [0..3n^2-1]$, be such that

$$\sum_{i=l+1}^{m} c_i * w_i(x) \; + \; \sum_{j=0}^{3n^2-1} d_j * x^j t(x) \; = \; 0.$$

We now show that all c_i, d_j must be zero. Note each $w_i(x)$ is defined to be a polynomial that evaluates to one at each (unique) β_i such that β_i is the root of $t(x)$ corresponding to the unique gate that has output wire numbered i, and evaluates to zero at all other β that are roots of $t(x)$. Then, since β_i is a root of $t(x)$, we have that the above sum evaluates to c_i at β_i, which forces c_i to be zero. Thus, the above sum is essentially $h(x)t(x)$, for some polynomial $h(x)$ of degree $3n^2 - 1$. But, for this polynomial to be zero over $\mathbb{F}_p^{(f)}$, $h(x)$ must be identically zero as $\mathbb{F}_p^{(f)}[x]$ is a unique factorization domain. This forces all d_j to be zero as well. □

Let $\boldsymbol{\zeta}$ stand for the vector of $2n$ variables $\{\alpha_j, \beta_j\}_{j\in[0..n-1]}$. Assume that the adversarially generated A_j (for each j) contains a linear term encoding $a_j(\boldsymbol{\zeta}) = \vec{a_j}^\top \cdot \boldsymbol{\zeta}$ (using $\mathbf{e}_\alpha$, $\mathbf{e}_\beta$ in the CRS) Similarly, suppose B_j contains a linear term encoding $b_j(\boldsymbol{\zeta}) = \vec{b_j}^\top \cdot \boldsymbol{\zeta}$, Of course, A_j is allowed to have other linear combinations of the level-one encodings in the CRS, but the contributions from encodings of $\boldsymbol{\zeta}$ is stipulated to be $a_j(\boldsymbol{\zeta})$. Now, arrange in a matrix $\mathbf{F}$ the columns $\{\vec{b_j}, \vec{a_j}\}_{j\in[0..n-1]}$.

Let I_n be the $(n \times n)$ identity matrix. Define a $2n \times 2n$ matrix σ as follows

$$\sigma = I_n \otimes \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{9}$$

Note that σ is orthogonal and symmetric. Thus $\sigma = \sigma^\top = \sigma^{-1}$.

Lemma 9 *If the verification equation holds, then with high probability the matrix $\mathbf{F}$ is invertible.*

Proof: If the verification holds, then the encoding Φ of $\sum_j \alpha_j\beta_j$ must be cancelled by other terms. A quick inspection of the remaining terms in the verification equations shows that this cancellation can only come from $\sum_j A_j \cdot B_j$. In particular, by Schwartz-Zippel theorem, it must hold that

$$\sum_j \alpha_j\beta_j = \sum_j a_j(\boldsymbol{\zeta}) * b_j(\boldsymbol{\zeta}).$$

Since $\sum_j \alpha_j\beta_j$ can be written as $\frac{1}{2} \cdot \boldsymbol{\zeta}^\top \cdot \sigma \cdot \boldsymbol{\zeta}$, the above is equivalent to the requirement that

$$\boldsymbol{\zeta}^\top \cdot \sigma \cdot \boldsymbol{\zeta} = \boldsymbol{\zeta}^\top \cdot \mathbf{F} \cdot \sigma \cdot \mathbf{F}^\top \cdot \boldsymbol{\zeta},$$

and again by Schwartz-Zippel, with high probability the following holds

$$\sigma = \mathbf{F} \cdot \sigma \cdot \mathbf{F}^\top. \tag{10}$$

The claim follows by taking determinant of both sides, and noting that determinant of σ is non-zero, and $\det(\mathbf{F}^\top) = \det \mathbf{F}$, and that determinant commutes with (square-) matrix product. □

Lemma 10 *In an affine proof that passes the verification equation, the elements A_j, B_j (for all $j \in [0..n-1]$) can have no linear contributions from $e_{c,i}$ and $e_{t,i}$.*

Proof:

Fix a $j^* \in [0..n-1]$.

Note that given the level-one encodings in the CRS, the only monomials in powers of x_{j^*} and θ_{j^*}/δ come from $\theta_{j^*} w_i(x_{j^*})/\delta$ (in encodings $e_{c,i}$) and $\theta_{j^*} x_{j^*}^i t(x_{j^*})/\delta$ (in encodings $e_{t,i}$). Let the linear contributions to A_j from e_{c_i} and $e_{t,i}$ be given by coefficients $A_{d,i,j}$ and $A_{t,i,j}$ respectively. Similarly, the linear contributions to B are given by coefficients $B_{d,i,j}$ and $B_{t,i,j}$. Then the net contribution to the verification equation with monomials in powers of x_{j^*} and (linear in) $\boldsymbol{\zeta} * \theta_{j^*}/\delta$ is

$$\sum_j \left(\sum A_{d,i,j}\theta_{j^*} w_i(x_{j^*})/\delta + \sum A_{t,i,j}\theta_{j^*} x_{j^*}^i t(x_{j^*})/\delta \right) * b_j(\boldsymbol{\zeta}) +$$
$$\sum_j \left(\sum B_{d,i,j}\theta_{j^*} w_i(x_{j^*})/\delta + \sum B_{t,i,j}\theta_{j^*} x_{j^*}^i t(x_{j^*})/\delta \right) * a_j(\boldsymbol{\zeta})$$

If the vector $\mathbf{x}_{j^*}$ denotes the list of monomials $\{x_{j^*}^{i+k}\}_{i\in I,k\in[0..n^2-1]}$, the set of polynomials $\mathcal{W}_{j^*}$ (consisting of all polynomials $\{w_i(x_{j^*})\}_{i\in[l+1..m]}$ and $\{x_{j^*}^i t(x_{j^*})\}_{i\in[0..n^2-1]}$) can be written as $\mathsf{W}\cdot\mathbf{x}_{j^*}$. Lemma 8 showed that W is invertible (it is easy to check that it is a square matrix). Next we write the vector of coefficients $A_{d,i,j}$ appended by coefficients $A_{t,i,j}$ as $\vec{A}_j$, and similarly the vector of coefficients $B_{d,i,j}$ appended by coefficients $B_{t,i,j}$ as $\vec{B}_j$, and so forth Then the above expression can be written as

$$\sum_j \vec{A}_j^\top \mathsf{W}\mathbf{x}_{j^*} * b_j(\boldsymbol{\zeta}) * \theta_{j^*}/\delta + \vec{B}_j^\top \mathsf{W}\mathbf{x}_{j^*} * a_j(\boldsymbol{\zeta}) * \theta_{j^*}/\delta$$

Note that "$*$" denotes polynomial multiplication. However, since the polynomials are over different set of variables, namely x_{j^*} and $\boldsymbol{\zeta} * \theta_{j^*}/\delta$, the above can be written as a tensor product as follows. We first write each of $a_j(\boldsymbol{\zeta}), b_j(\boldsymbol{\zeta})$ as $\vec{a}_j^\top \cdot \boldsymbol{\zeta}$, $\vec{b}_j^\top \cdot \boldsymbol{\zeta}$. Then the above becomes

$$\begin{aligned}
&\sum_j \vec{A}_j^\top \mathsf{W}\mathbf{x}_{j^*} * (\vec{b}_j^\top \cdot \boldsymbol{\zeta} * \theta_{j^*}/\delta) + \vec{B}_j^\top \mathsf{W}\mathbf{x}_{j^*} * (\vec{a}_j^\top \cdot \boldsymbol{\zeta} * \theta_{j^*}/\delta) \\
= \\
&\sum_j \vec{A}_j^\top (\vec{b}_j^\top \otimes \mathsf{W}) \cdot (\boldsymbol{\zeta} * \theta_{j^*}/\delta \otimes \mathbf{x}_{j^*}) + \vec{B}_j^\top (\vec{a}_j^\top \otimes \mathsf{W}) \cdot (\boldsymbol{\zeta} * \theta_{j^*}/\delta \otimes \mathbf{x}_{j^*})
\end{aligned}$$

If we now concatenate all the coefficients $\vec{A}_j, \vec{B}_j$ into a single vector $\vec{D}$, the above becomes

$$\vec{D}^\top (\mathsf{F}^\top \otimes \mathsf{W}) \cdot (\boldsymbol{\zeta} * \theta_{j^*}/\delta \otimes \mathbf{x}_{j^*})$$

Since F is invertible by lemma 9, and so is W, this then implies that if the above is zero then $\vec{D}$ is zero. □

Lemma 11 *In an affine proof that passes the verification equation, the elements A_j, B_j (for all $j \in [0..n-1]$) can have no linear contributions from $e_{d,i}$ or e_γ.*

Proof: The proof for no linear contributions from $e_{d,i}$ is same as the proof of previous lemma 10, but with monomials of powers of x_{j^*} and θ_{j^*}/δ replaced by monomials of powers of x_{j^*} and θ_{j^*}/γ. For e_γ, note that as in the proof of lemma 10, the linear terms $a_j(\boldsymbol{\zeta}), b_j(\boldsymbol{\zeta})$ in A_j, B_j are linearly independent by lemma **??**. Since, there is no other way to generate $\gamma \cdot \boldsymbol{\zeta}$ in the verification equation, the contributions from e_γ must be zero in all of A_j, B_j. □

Lemma 12 *In an affine proof that passes the verification equation, the element C can have no linear contributions from $e_{d,i}$ and e_γ.*

Proof: Since there is no linear contribution from e_γ in any of A_j, B_j by lemma 11, any linear contribution from e_γ to C would yield a encoding of (linear in) $\delta\gamma$ in the verification equation. Since it cannot be canceled by any other term in the verification equation, this contribution must be zero. Similar consideration with δ/γ shows that there can be no non-trivial linear contribution from $e_{d,i}$ to C. Further, since $w_i(x_j)$ are linearly independent by lemma 8, there can in fact be no linear contribution from $e_{d,i}$. □

Proof: (**Soundness**) By above lemmas, it follows that A_j, B_j are of the form

$$\begin{aligned}
A_j &= \left(\vec{A}_{x,j}^\top \cdot \mathbf{e}_x + \vec{A}_{w,j}^\top \cdot \mathbf{e}_w\right) + \vec{a}_j^\top \cdot \boldsymbol{\zeta} + r_{a,j} e_\delta \\
B_j &= \left(\vec{B}_{x,j}^\top \cdot \mathbf{e}_x + \vec{B}_{w,j}^\top \cdot \mathbf{e}_w\right) + \vec{b}_j^\top \cdot \boldsymbol{\zeta} + r_{b,j} e_\delta
\end{aligned}$$

Now, write A as the $2 * n^2 \times 2 * n$ matrix with columns comprising of $\{\vec{A}_{x,j}, \vec{B}_{x,j}\}$. Similarly, let E be the vector $(\mathbf{e}_x, \mathbf{e}_w)$. Let F be the $2n \times 2n$ matrix with columns $\{\vec{b}_j, \vec{a}_j\}_j$ (note the permuted order). Finally, let the vector $\{r_{a,j}, r_{b,j}\}_j$ be denoted by R.

Then, the above is written more succinctly as

$$[...A_j\ B_j...]^\top = \mathbf{A}^\top \cdot \mathsf{E} + \mathbf{F}^\top \cdot \boldsymbol{\zeta} + e_\delta \mathsf{R}$$

Further, by lemma 12, C is of the form

$$C = \sum \tilde{c}_i e_{c,i} + \sum h_i e_{t,i} + \left(\sum C_{x,i,j} e_{x,i,j} + ... + \sum C_{w,i,j} e_{w,i,j}\right) + \vec{c}^\top \cdot \boldsymbol{\zeta} + r_c e_\delta,$$

for some $\tilde{c}_i, h_i, C_{x,i,j}, ..., C_{w,i,j}$ and $\vec{c}, r_c$. The vector of $\tilde{c}_i$ values (including those for input and output wires) will be written as $\tilde{\mathbf{c}}$. Now, given that the verification equation must pass, and by matching coefficients of various monomials it follows that

$$\begin{aligned} &\sum C_{x,i,j} e_{x,i,j} + ... + \sum C_{w,i,j} e_{w,i,j} \\ &= \mathsf{R}^\top \cdot \mathbf{A} \cdot \mathsf{E}, \end{aligned} \tag{11}$$

$$\vec{c}^\top \cdot \boldsymbol{\zeta} \ = \ \mathsf{R}^\top \mathbf{F}^\top \cdot \boldsymbol{\zeta}, \tag{12}$$

$$r_c \ = \ \sum_j r_{a_j} * r_{b,j}, \tag{13}$$

and finally (modulo p)

$$\begin{aligned} &\Phi + \delta * \sum_{i=l+1}^{m} \tilde{c}_i e_{c,i} + \delta * \sum_{i=0}^{n-2} h_i e_{t,i} + \gamma * \sum_{i=0}^{l} \tilde{c}_i e_{d,i} \\ &= \sum_j \left(\vec{A}^\top_{x,j} \cdot \mathbf{e}_x + \vec{A}^\top_{w,j} \cdot \mathbf{e}_w + \vec{a}_j^\top \cdot \boldsymbol{\zeta}\right) * \left(\vec{B}^\top_{x,j} \cdot \mathbf{e}_x + \vec{B}^\top_{w,j} \cdot \mathbf{e}_w + \vec{b}_j^\top \cdot \boldsymbol{\zeta}\right) \end{aligned} \tag{14}$$

Let $\mathbf{u}(x)$ stand for the vector of polynomials $u_i(x)$. Similarly, define $\mathbf{v}(x)$ and $\mathbf{w}(x)$. Further, instead of E, define X as the vector with $2n$ components $\{\mathbf{u}(x_j), \theta_j \mathbf{v}(x_j)\}$ (i.e. $2n^2$ expanded components). Also, redefine the matrix $\mathbf{A}$ as the $2*n^2 \times 2*n$ matrix with columns comprising of $\{\vec{A'}_{x,j}, \vec{B'}_{x,j}\}$, where the primed vectors are now coefficients in terms of X instead of E. Also, let $h(x)$ be the polynomial with coefficients h_i. Since the encodings are additively homomorphic, the above equation (14) then imply the requirement (mod p)

$$\begin{aligned} &\mathsf{X}^\top \cdot (\sigma \otimes \tilde{\mathbf{c}}) \cdot \boldsymbol{\zeta} + \sum_j \theta_j \mathbf{w}(x_j)^\top \cdot \tilde{\mathbf{c}} \ + \ h(x_j)\theta_j t(x_j) \ + \ \sum_j \alpha_j \beta_j \\ &= \sum_j \left(\mathsf{X}^\top \cdot \mathbf{A}_{2j} + \vec{a}_j^\top \cdot \boldsymbol{\zeta}\right) * \left(\mathsf{X}^\top \cdot \mathbf{A}_{2j+1} + \vec{b}_j^\top \cdot \boldsymbol{\zeta}\right) \quad \pmod p \end{aligned} \tag{15}$$

By Schwartz-Zippel[5], all the variables $\{x_j, \alpha_j, \beta_j, \theta_j\}$, can be treated as independent variables. Thus, the above equation can be seen as polynomial equation in these variables over the underlying ring Z_p. Thus, the above can be broken down into separate requirements

$$\sum_j \alpha_j \beta_j = \sum_j (\boldsymbol{\zeta}^\top \cdot \vec{a}_j) * (\vec{b}_j^\top \cdot \boldsymbol{\zeta}) \tag{16}$$

$$\sum_j \theta_j \cdot (\mathbf{w}(x_j)^\top \cdot \tilde{\mathbf{c}} + h(x_j)t(x_j)) = \sum_j \left(\mathsf{X}^\top \cdot \mathbf{A}_{2j}\right) * \left(\mathsf{X}^\top \cdot \mathbf{A}_{2j+1}\right) \tag{17}$$

$$(\mathsf{X}^\top \cdot (\sigma \otimes \tilde{\mathbf{c}}) \cdot \boldsymbol{\zeta}) = \sum_j \left(\mathsf{X}^\top \cdot \mathbf{A}_{2j} * \vec{b}_j^\top \cdot \boldsymbol{\zeta}\right) + \left(\mathsf{X}^\top \cdot \mathbf{A}_{2j+1} * \vec{a}_j^\top \cdot \boldsymbol{\zeta}\right) \tag{18}$$

[5]slight variant for rings Z_p, where p is power of prime

We have already encountered and used equation (16) in lemma 9, which was used to show that F is invertible. Now, the last equation (18) implies

$$\sigma \otimes \tilde{\mathbf{c}} \;=\; \mathbf{A} \cdot \mathbf{F}^{\top}. \tag{19}$$

which is equivalent to

$$\mathbf{A} \;=\; (\sigma \otimes \tilde{\mathbf{c}}) \cdot \mathbf{F}^{-\top} \;=\; (\sigma \cdot \mathbf{F}^{-\top}) \otimes \tilde{\mathbf{c}}. \tag{20}$$

Now, since the right hand side of equation (17) is a symmetric bilinear form in X, it can be re-written as

$$\begin{aligned}
&\sum_j \theta_j \cdot (\mathbf{w}(x_j)^{\top} \cdot \tilde{\mathbf{c}} + h(x_j)t(x_j)) \\
&= \left(\mathsf{X}^{\top} \cdot \mathbf{A} \cdot \sigma \cdot \mathbf{A}^{\top} \cdot \mathsf{X}\right) \\
&= \left(\mathsf{X}^{\top} \cdot (\sigma \cdot \mathbf{F}^{-\top}) \otimes \tilde{\mathbf{c}} \cdot \sigma \cdot (\mathbf{F}^{-1} \cdot \sigma) \otimes \tilde{\mathbf{c}}^{\top} \cdot \mathsf{X}\right) \\
&= \left(\mathsf{X}^{\top} \cdot (\sigma \cdot \mathbf{F}^{-\top}) \otimes \tilde{\mathbf{c}} \cdot (\sigma \otimes 1) \cdot (\mathbf{F}^{-1} \cdot \sigma) \otimes \tilde{\mathbf{c}}^{\top} \cdot \mathsf{X}\right) \\
&= \left(\mathsf{X}^{\top} \cdot ((\sigma \cdot \mathbf{F}^{-\top}) \cdot \sigma \cdot (\mathbf{F}^{-1} \cdot \sigma)) \otimes (\tilde{\mathbf{c}} \cdot \tilde{\mathbf{c}}^{\top}) \cdot \mathsf{X}\right),
\end{aligned}$$

where we used the fact that for arbitrary matrices A, B, C, D (of compatible sizes) it is the case that $(A \otimes B)(C \otimes D) = (AC \otimes BD)$. Further, using (10), which impies that $\mathbf{F}^{-\top} = \sigma^{-1} \cdot \mathbf{F} \cdot \sigma$, and using the fact that $\sigma^{-1} = \sigma$, the above simplifies to

$$\sum_j \theta_j \cdot (\mathbf{w}(x_j)^{\top} \cdot \tilde{\mathbf{c}} + h(x_j)t(x_j)) = \left(\mathsf{X}^{\top} \cdot (\sigma) \otimes (\tilde{\mathbf{c}} \cdot \tilde{\mathbf{c}}^{\top}) \cdot \mathsf{X}\right).$$

Now, we will write the polynomial $\mathbf{u}(x)^{\top} \cdot \tilde{\mathbf{c}}$ as $\mathcal{U}(x)$ and $\mathbf{v}(x)^{\top} \cdot \tilde{c}$ as $\mathcal{V}(x)$. Then, $\mathsf{X}^{\top} \cdot \tilde{\mathbf{c}}$ is the vector $\{\mathcal{U}(x_j), \theta_j \mathcal{V}(x_j)\}_j$. Further, equating monomials involving powers of x_1 and (linear in) θ_1, we get from above that

$$\begin{aligned}
\mathbf{w}(x_1)^{\top} \cdot \tilde{\mathbf{c}} + h(x_1)t(x_1) \;&=\; \mathcal{U}(x_1)\mathcal{V}(x_1) \bmod p \\
&=\; (\mathbf{u}(x_1)^{\top}\tilde{\mathbf{c}}) * (\mathbf{v}(x_1)^{\top}\tilde{\mathbf{c}}) \bmod p.
\end{aligned}$$

Noting that $\tilde{\mathbf{c}}$ is consistent with input and output wires, the proof of soundness follows from property of QAP. □

7.2 Proof of Security against Annihilation Attacks

7.2.1 Level Two Zeroes obtained from the CRS.

In this section, we study the (encodings of) level-two zeroes that can be obtained identically from the CRS elements. Since these would be encodings of zero, many such encodings can be used to annihilate the coefficient of g in the encoding, and hence obtain an element in the ideal $\langle g \rangle$ (after multiplying by the provided zero-testing element).

Since the verification test itself provides instances of encodings of zero, we will now characterize all possible encodings of zero that can be obtained identically from the CRS elements. Note, we restrict ourselves to obtaining level-two zeroes, as the zero-testing element is required to cancel out the secret z, and the provided zero-tester can only be used gainfully on level-two zero encodings.

This restriction on how only level-two zeroes can be used for annihilation, and that these level two zeroes are only obtained by pairings, fits with type I queries of the Miles-Sahai-Zhandry weak multilinear map model.

First point of observation is that the encodings of powers of x_j, i.e. $\mathbf{e}_x$, cannot by itself be used to obtain encodings of zero as ψ is a sum-collision-free mapping. So, the first thing to analyze is the encodings that can

be paired so as to be useful in obtaining a level-two zero. In particular, each such pairing will result in terms that employ the auxiliary variables $\alpha, \beta, \gamma, \delta, \theta$ or their reciprocal or none of these. So, for all the potential monomials that can be obtained from these (including one), the zeroes must be obtained individually for each such monomial. We will refer to these various monomials, such as α_j/γ, as auxiliary monomials.

1. The pairing of $\mathbf{e}_x$ with itself will have auxiliary monomial one. As we will see, no other pairing has auxiliary monomial one. Similarly, pairing of $\mathbf{e}_x$ with any of $\mathbf{e}_c$, $\mathbf{e}_d$ or $\mathbf{e}_t$, obtains auxiliraly monomials that cannot be obtained by any other way of pairing.
2. The pairing of $\mathbf{e}_w$ with itself will have auxiliary monomials $\theta_j\theta_{j'}$. As we will see, no other pairing has such auxiliary monomials. Similarly, pairing of $\mathbf{e}_w$ with any of $\mathbf{e}_c$, $\mathbf{e}_d$ or $\mathbf{e}_t$, obtains auxiliraly monomials that cannot be obtained by any other way of pairing.
3. The pairing of $\mathbf{e}_x$ and $\mathbf{e}_w$ will have auxiliary monomials θ_j. These can also be obtained by pairing $\mathbf{e}_\gamma$ with $\mathbf{e}_c$ or pairing $\mathbf{e}_\delta$ with $\mathbf{e}_t$ or $\mathbf{e}_d$. Moreover, the level two encodings $\mathbf{e}_p$ also have the same auxiliary monomials,
4. Pairing of $\mathbf{e}_\alpha$ with $\mathbf{e}_w$ and pairing of $\mathbf{e}_\beta$ with $\mathbf{e}_x$ also obtain auxiliary monomials that are obtained in the previous item.
5. The Pairing of $\mathbf{e}_{\alpha_j}$ and $\mathbf{e}_{\beta_j}$ obtain auxiliary monomials that are the same in the given level two encoding Φ.

From the above it is clear that any level-two zero that can be obtained will actually have a structure similar to the verification test. More precisely, it will be of the form

$$\sum_j A_j \cdot B_j + \Phi - pP + C \cdot \mathbf{e}_\delta + D \cdot \mathbf{e}_\gamma,$$

where

- A_j and B_j are a linear combination of $\mathbf{e}_\alpha$, $\mathbf{e}_\beta$, e_δ and $\bar{\mathbf{e}}_x$. Here j may range beyond $[n]$, but it is worth proving that it suffices to only consider $j \in [n]$.
- C is a linear combination of $\mathbf{e}_\alpha$, $\mathbf{e}_\beta$, e_δ, $\mathbf{e}_c$, and $\mathbf{e}_t$, $\bar{\mathbf{e}}_x$ and $\bar{\mathbf{e}}_w$.
- P is a linear combination of $e_{p,i,k}$,
- D is a linear combination of $\mathbf{e}_d$.

From the above consideration, as in Section 7.1, let A_j, B_j be of the form

$$A_j = \left(\vec{A}_{x,j}^\top \cdot \mathbf{e}_x + \vec{A}_{w,j}^\top \cdot \mathbf{e}_w\right) + \vec{a}_j^\top \cdot \boldsymbol{\zeta} + r_{a,j} e_\delta$$
$$B_j = \left(\vec{B}_{x,j}^\top \cdot \mathbf{e}_x + \vec{B}_{w,j}^\top \cdot \mathbf{e}_w\right) + \vec{b}_j^\top \cdot \boldsymbol{\zeta} + r_{b,j} e_\delta$$

Now, write $\mathbf{A}$ as the $2n^2 \times 2n$ matrix with columns comprising of $\{\vec{A}_{x,j}, \vec{B}_{x,j}\}$. Similarly, let E be the vector $(\mathbf{e}_x, \mathbf{e}_w)$. Let $\mathbf{F}$ be the $2n \times 2n$ matrix with columns $\{\vec{b}_j, \vec{a}_j\}_j$ (note the permuted order). Finally, let the vector $\{r_{a,j}, r_{b,j}\}_j$ be denoted by R.

Then, the above is written more succinctly as

$$[...A_j\ B_j...]^\top = \mathbf{A}^\top \cdot \mathsf{E} + \mathbf{F}^\top \cdot \boldsymbol{\zeta} + e_\delta \mathsf{R}$$

Again by above considerations, let C be of the form

$$C = \sum \tilde{c}_i e_{c,i} + \sum h_i e_{t,i} + \left(\sum C_{x,i,j} e_{x,i,j} + ... + \sum C_{w,i,j} e_{w,i,j}\right) + \vec{c}^\top \cdot \boldsymbol{\zeta} + r_c e_\delta,$$

for some $\tilde{c}_i, h_i, C_{x,i,j}, ..., C_{w,i,j}$ and $\vec{c}, r_c$. The vector of $\tilde{c}_i$ values (including those for coefficients of $\mathbf{e}_d$ coming from D) will be written as $\tilde{\mathbf{c}}$. Now, given that we get a zero, and by matching coefficients of various monomials it follows that

$$\begin{aligned}&\sum C_{x,i,j} e_{x,i,j} + ... + \sum C_{w,i,j} e_{w,i,j} \\ &= \mathsf{R}^\top \cdot \mathbf{A} \cdot \mathsf{E}, \end{aligned} \tag{21}$$

$$\vec{c}^\top \cdot \boldsymbol{\zeta} \;=\; \mathsf{R}^\top \mathbf{F}^\top \cdot \boldsymbol{\zeta}, \tag{22}$$

$$r_c \;=\; \sum_j r_{a_j} * r_{b,j}, \tag{23}$$

and finally (modulo p)

$$\begin{aligned}&\Phi + \delta * \sum_{i=l+1}^{m} \tilde{c}_i e_{c,i} + \delta * \sum_{i=0}^{n-2} h_i e_{t,i} + \gamma * \sum_{i=0}^{l} \tilde{c}_i e_{d,i} \\ &= \sum_j \left(\vec{A}^\top_{x,j} \cdot \mathbf{e}_x + \vec{A}^\top_{w,j} \cdot \mathbf{e}_w + \vec{a}^\top_j \cdot \boldsymbol{\zeta} \right) * \left(\vec{B}^\top_{x,j} \cdot \mathbf{e}_x + \vec{B}^\top_{w,j} \cdot \mathbf{e}_w + \vec{b}^\top_j \cdot \boldsymbol{\zeta} \right) \end{aligned} \tag{24}$$

Let $\mathbf{u}(x)$ stand for the vector of polynomials $u_i(x)$. Similarly, define $\mathbf{v}(x)$ and $\mathbf{w}(x)$. Further, instead of E, define X as the vector with $2n$ components $\{\mathbf{u}(x_j), \theta_j \mathbf{v}(x_j)\}$ (i.e. $2n^2$ expanded components). Also, redefine the matrix $\mathbf{A}$ as the $2*n^2 \times 2*n$ matrix with columns comprising of $\{\vec{A'}_{x,j}, \vec{B'}_{x,j}\}$, where the primed vectors are now coefficients in terms of X instead of E. Also, let $h(x)$ be the polynomial with coefficients h_i. Then, as in the proof of soundness, it follows that

$$\sigma \otimes \tilde{\mathbf{c}} \;=\; \mathbf{A} \cdot \mathbf{F}^\top. \tag{25}$$

which is equivalent to

$$\mathbf{A} \;=\; (\sigma \otimes \tilde{\mathbf{c}}) \cdot \mathbf{F}^{-\top} \;=\; (\sigma \cdot \mathbf{F}^{-\top}) \otimes \tilde{\mathbf{c}}. \tag{26}$$

7.2.2 Formal expressions of coefficients of g in level-two zeroes

Since, in this section we will be looking for coefficients of g in the encoding of zeroes obtained above, we introduce terminology to represent the randomness used in the encodings. Recall, an encoding of a is $(a + s \cdot g)/z$. Thus, the variables representing the randomness s, are conveniently represented by boldface $\mathbf{s}_x$ to represent the whole set $s_{x,i,j}$, similar to how $\mathbf{e}_x$ stands for the set $e_{x,i,j}$. We extend this notation to all encodings such as $\mathbf{s}_c$ etc. Also, for purpose of analysis in this section, we will ignore the term z^{-1} in the encoding. Thus,

$$\begin{aligned}\mathbf{e}_x &= \{(e_{x,i,j} \;=)\; \mathsf{enc}(x_j^{\psi(i)})\}_{i,j} \\ &= \{x_j^{\psi(i)} + s_{x,i,j} \cdot g\}_{i,j}\end{aligned}$$

Thus, while in Section 7.1, $\sum_j (\boldsymbol{\zeta}^\top \cdot \vec{a}_j) * (\vec{b}^\top_j \cdot \boldsymbol{\zeta})$ was equivalently written as half of $\boldsymbol{\zeta}^\top \cdot \mathbf{F} \cdot \sigma \cdot \mathbf{F}^\top \cdot \boldsymbol{\zeta}$, we now include the "$g$" term in the encoding to this equation and then we obtain the expression

$$(\boldsymbol{\zeta} + \mathbf{s}_\zeta g)^\top \cdot \mathbf{F} \cdot \sigma \cdot \mathbf{F}^\top \cdot (\boldsymbol{\zeta} + \mathbf{s}_\zeta g)$$

For convenience, we will denote the payload in encodings $\mathbf{e}_c$ as $\mathbf{y}_c$, and similarly for other encodings. In other words, $\mathbf{e}_c = \mathbf{y}_c + \mathbf{s}_c g$. So, while the equation 24 above must hold to obtain an encoding for zero, the

coefficient of g in this encoding of zero will be

$$
\begin{aligned}
& s_\Phi + s_\delta * \sum_{i=l+1}^{m} \tilde{c}_i y_{c,i} + \delta * \sum_{i=l+1}^{m} \tilde{c}_i s_{c,i} + \\
& s_\delta * \sum_{i=0}^{n-2} h_i y_{t,i} + \delta * \sum_{i=0}^{n-2} h_i s_{t,i} + \\
& s_\gamma * \sum_{i=0}^{l} \tilde{c}_i y_{d,i} + \gamma * \sum_{i=0}^{l} \tilde{c}_i s_{d,i} - \\
& \sum_j \left(\vec{A}_{x,j}^\top \cdot \mathbf{s}_x + \vec{A}_{w,j}^\top \cdot \mathbf{s}_w + \vec{a}_j^\top \cdot s_{\boldsymbol{\zeta}} \right) * \left(\vec{B}_{x,j}^\top \cdot \mathbf{y}_x + \vec{B}_{w,j}^\top \cdot \mathbf{y}_w + \vec{b}_j^\top \cdot \boldsymbol{\zeta} \right) - \\
& \sum_j \left(\vec{A}_{x,j}^\top \cdot \mathbf{y}_x + \vec{A}_{w,j}^\top \cdot \mathbf{y}_w + \vec{a}_j^\top \cdot \boldsymbol{\zeta} \right) * \left(\vec{B}_{x,j}^\top \cdot \mathbf{s}_x + \vec{B}_{w,j}^\top \cdot \mathbf{s}_w + \vec{b}_j^\top \cdot s_{\boldsymbol{\zeta}} \right)
\end{aligned} \tag{27}
$$

We make the task of the Adversary easier, by focusing only on the terms involving $\mathbf{s}_x, \mathbf{s}_w, s_{\boldsymbol{\zeta}}$, as these must be annihilated as well. Then the above expression simplifies to

$$
\begin{aligned}
& \sum_j \left(\vec{A}_{x,j}^\top \cdot \mathbf{s}_x + \vec{A}_{w,j}^\top \cdot \mathbf{s}_w + \vec{a}_j^\top \cdot s_{\boldsymbol{\zeta}} \right) * \left(\vec{B}_{x,j}^\top \cdot \mathbf{y}_x + \vec{B}_{w,j}^\top \cdot \mathbf{y}_w + \vec{b}_j^\top \cdot \boldsymbol{\zeta} \right) + \\
& \sum_j \left(\vec{A}_{x,j}^\top \cdot \mathbf{y}_x + \vec{A}_{w,j}^\top \cdot \mathbf{y}_w + \vec{a}_j^\top \cdot \boldsymbol{\zeta} \right) * \left(\vec{B}_{x,j}^\top \cdot \mathbf{s}_x + \vec{B}_{w,j}^\top \cdot \mathbf{s}_w + \vec{b}_j^\top \cdot s_{\boldsymbol{\zeta}} \right)
\end{aligned} \tag{28}
$$

Now, writing the encoding for X as $\mathsf{X} + \mathsf{S}g$, the above can be written more succinctly as

$$
\begin{aligned}
& \mathsf{X}^\top \cdot (\mathbf{A} \sigma \mathbf{A}^\top) \cdot \mathsf{S} + \\
& \boldsymbol{\zeta}^\top \cdot (\mathbf{F} \mathbf{A}^\top) \cdot \mathsf{S} + \\
& s_{\boldsymbol{\zeta}}^\top \cdot (\mathbf{F} \mathbf{A}^\top) \cdot \mathsf{X} + \\
& s_{\boldsymbol{\zeta}}^\top \cdot (\mathbf{F} \sigma \mathbf{F}^\top) \cdot \boldsymbol{\zeta}
\end{aligned} \tag{29}
$$

Since, from the requirements imposed for obtaining zero encodings, $\mathbf{A} \sigma \mathbf{A}^\top = \sigma \otimes (\tilde{\mathbf{c}} \tilde{\mathbf{c}}^\top)$ and $\mathbf{F} \mathbf{A}^\top = \sigma \otimes \tilde{\mathbf{c}}^\top$, and also $\mathbf{F} \sigma \mathbf{F}^\top = \sigma$, the above simplifies to

$$
\mathsf{X}^\top \cdot (\sigma \otimes (\tilde{\mathbf{c}}\, \tilde{\mathbf{c}}^\top)) \cdot \mathsf{S} + \boldsymbol{\zeta}^\top \cdot (\sigma \otimes \tilde{\mathbf{c}}^\top) \cdot \mathsf{S} + s_{\boldsymbol{\zeta}}^\top \cdot (\sigma \otimes \tilde{\mathbf{c}}^\top) \cdot \mathsf{X} + s_{\boldsymbol{\zeta}}^\top \cdot \sigma \cdot \boldsymbol{\zeta} \tag{30}
$$

Now, note that the adversary may choose to take $\tilde{\mathbf{c}} = 0$, in which case the Adversary obtains only a single expression $s_{\boldsymbol{\zeta}}^\top \cdot \sigma \cdot \boldsymbol{\zeta}$, i.e. independent of $\mathbf{A}$. On the other hand, with many different $\tilde{\mathbf{c}} \neq 0$, he may take linear combinations of the above expressions to obtain

$$
\mathsf{X}^\top \cdot (\sigma \otimes \tilde{C}) \cdot \mathsf{S},
$$

where $\tilde{C}$ is any $n \times n$ symmetric matrix, and in fact all $n \times n$ elementary symmetric matrices. In the above, we have ignored the terms involving $s_{\boldsymbol{\zeta}}$, as it only makes the task of the Adversary easier.

For example, if $\tilde{C}$ is the elementary diagonal matrix (hence, symmetric) with only the i-th diagonal entry on and equal to one, the above simplifies to

$$
\sum_j (u_i(x_j) + \theta_j v_i(x_j)) s_{x,i,j}
$$

Further if the elementary symmetric matrix $\tilde{C}$ has only the $(i1, i2)$-th and $(i2, i1)$-th entry on and one, we get the expression

$$\sum_j (u_{i1}(x_j) + \theta_j v_{i1}(x_j)) s_{x,i2,j} \ + \ (u_{i2}(x_j) + \theta_j v_{i2}(x_j)) s_{x,i1,j}$$

We can now ignore θ_j terms, to make the task easier for the adversary, and also take $u_i(x_j) = x_j^{\psi(i)}$ or even simpler, $u_i(x_j) = x_j^i$. Then the above simplifies to (call these terms $T_{i1,i2}$)

$$T_{i1,i2} \ = \ \sum_j x_j^{i1} \ s_{x,i2,j} \ + \ x_j^{i2} \ s_{x,i1,j} \tag{31}$$

Employing the PRG Assumption to rule out Annihilation Since the multivariate polynomials above in equations 31 are the same as the defining polynomials of the candidate PRG from section 6, the assumption that the polynomials form a PRG implies that there is no efficient annihilator of equations 31.

Acknowledgements

The authors would like to thank Craig Gentry and Shai Halevi for initial discussions. In particular, they broached the idea of bootstrapping the verification circuit, which is a rather simple circuit for lattice based schemes.

References

[ABD16] Martin R. Albrecht, Shi Bai, and Léo Ducas, A subfield lattice attack on overstretched NTRU assumptions - cryptanalysis of some FHE and graded encoding schemes, CRYPTO 2016, Part I (Matthew Robshaw and Jonathan Katz, eds.), LNCS, vol. 9814, Springer, Heidelberg, August 2016, pp. 153–178. B.0.1

[ACL+22] Martin R. Albrecht, Valerio Cini, Russell W. F. Lai, Giulio Malavolta, and Sri Aravinda Krishnan Thyagarajan, Lattice-based SNARKs: Publicly verifiable, preprocessing, and recursively composable - (extended abstract), CRYPTO 2022, Part II, LNCS, Springer, Heidelberg, August 2022, pp. 102–132. 1, 3

[BCI+13] Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth, Succinct non-interactive arguments via linear interactive proofs, TCC 2013 (Amit Sahai, ed.), LNCS, vol. 7785, Springer, Heidelberg, March 2013, pp. 315–333. 1

[BCLv17] Daniel J. Bernstein, Chitchanok Chuengsatiansup, Tanja Lange, and Christine van Vredendaal, NTRU prime: Reducing attack surface at low cost, SAC 2017 (Carlisle Adams and Jan Camenisch, eds.), LNCS, vol. 10719, Springer, Heidelberg, August 2017, pp. 235–260. 2.1, 3.3

[BCR+19] Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward, Aurora: Transparent succinct arguments for R1CS, EUROCRYPT 2019, Part I (Yuval Ishai and Vincent Rijmen, eds.), LNCS, vol. 11476, Springer, Heidelberg, May 2019, pp. 103–128. 1, 7

[BCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner, Interactive oracle proofs, TCC 2016-B, Part II (Martin Hirt and Adam D. Smith, eds.), LNCS, vol. 9986, Springer, Heidelberg, October / November 2016, pp. 31–60. 1

[BEF+17] Jean-François Biasse, Thomas Espitau, Pierre-Alain Fouque, Alexandre Gélin, and Paul Kirchner, Computing generator in cyclotomic integer rings - A subfield algorithm for the principal ideal problem in $(1_{\overline{2}})$ and application to the cryptanalysis of a FHE scheme, EUROCRYPT 2017,

Part I (Jean-Sébastien Coron and Jesper Buus Nielsen, eds.), LNCS, vol. 10210, Springer, Heidelberg, April / May 2017, pp. 60–88. B.0.2

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan, (Leveled) fully homomorphic encryption without bootstrapping , ITCS 2012 (Shafi Goldwasser, ed.), ACM, January 2012, pp. 309–325. 3.2

[BISW17] Dan Boneh, Yuval Ishai, Amit Sahai, and David J. Wu, Lattice-based SNARGs and their application to more efficient obfuscation , EUROCRYPT 2017, Part III (Jean-Sébastien Coron and Jesper Buus Nielsen, eds.), LNCS, vol. 10212, Springer, Heidelberg, April / May 2017, pp. 247–277. 1

[BISW18] ———, Quasi-optimal SNARGs via linear multi-prover interactive proofs , EUROCRYPT 2018, Part III (Jesper Buus Nielsen and Vincent Rijmen, eds.), LNCS, vol. 10822, Springer, Heidelberg, April / May 2018, pp. 222–255. 1

[BS16] Jean-François Biasse and Fang Song, Efficient quantum algorithms for computing class groups and solving the principal ideal problem in arbitrary degree number fields , 27th SODA (Robert Krauthgamer, ed.), ACM-SIAM, January 2016, pp. 893–902. B.0.2

[CDPR16] Ronald Cramer, Léo Ducas, Chris Peikert, and Oded Regev, Recovering short generators of principal ideals in cyclotomic rings , EUROCRYPT 2016, Part II (Marc Fischlin and Jean-Sébastien Coron, eds.), LNCS, vol. 9666, Springer, Heidelberg, May 2016, pp. 559–585. 3.3, B.0.2

[CGS14] Peter Campbell, Michael Groves, and Dan Shepherd, Soliloquy: A cautionary tale , `https://docbox.etsi.org/Workshop/2014/201410_CRYPTO/S07_Systems_and_Attacks/S07_Groves_Annex.pdf`, 2014. B.0.2

[CJL16] Jung Hee Cheon, Jinhyuck Jeong, and Changmin Lee, An algorithm for ntru problems and cryptanalysis of the ggh multilinear map without a low-level encoding of zero , LMS Journal of Computation and Mathematics **19** (2016), no. A, 255–266. B.0.1

[CKPS00] Nicolas Courtois, Alexander Klimov, Jacques Patarin, and Adi Shamir, Efficient algorithms for solving overdefined systems of multivariate polynomial equations , EUROCRYPT 2000 (Bart Preneel, ed.), LNCS, vol. 1807, Springer, Heidelberg, May 2000, pp. 392–407. 4, 4

[CLM23] Valerio Cini, Russell W. F. Lai, and Giulio Malavolta, Lattice-based succinct arguments from vanishing polynomials - (extended abstract) , CRYPTO 2023, Part II, LNCS, Springer, Heidelberg, August 2023, pp. 72–105. 3

[DGG+16] Nico Döttling, Sanjam Garg, Divya Gupta, Peihan Miao, and Pratyay Mukherjee, Obfuscation from low noise multilinear maps , Cryptology ePrint Archive, Report 2016/599, 2016, `https://eprint.iacr.org/2016/599`. 3.3

[DGG+18] ———, Obfuscation from low noise multilinear maps , INDOCRYPT 2018 (Debrup Chakraborty and Tetsu Iwata, eds.), LNCS, vol. 11356, Springer, Heidelberg, December 2018, pp. 329–352. 3.2.2

[DP18] Léo Ducas and Alice Pellet-Mary, On the statistical leak of the GGH13 multilinear map and some variants , ASIACRYPT 2018, Part I (Thomas Peyrin and Steven Galbraith, eds.), LNCS, vol. 11272, Springer, Heidelberg, December 2018, pp. 465–493. 2.1.2, 3.2.2, 3.3, 2, 3.4, B.0.3, B.3

[EHKS14] Kirsten Eisenträger, Sean Hallgren, Alexei Kitaev, and Fang Song, A quantum algorithm for computing the unit group of an arbitrary degree number field , 46th ACM STOC (David B. Shmoys, ed.), ACM Press, May / June 2014, pp. 293–302. B.0.2

[GGH13] Sanjam Garg, Craig Gentry, and Shai Halevi, Candidate multilinear maps from ideal lattices , EUROCRYPT 2013 (Thomas Johansson and Phong Q. Nguyen, eds.), LNCS, vol. 7881, Springer, Heidelberg, May 2013, pp. 1–17. (document), 1, 2.3, 3, 3.2.2, 3.2.4, 3.3, 3.4, B.0.3, B.2, B.3

[GGPR12] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova, Quadratic span programs and succinct NIZKs without PCPs , Cryptology ePrint Archive, Report 2012/215, 2012, `https://eprint.iacr.org/2012/215`. 1

[GGPR13] ———, Quadratic span programs and succinct NIZKs without PCPs , EUROCRYPT 2013 (Thomas Johansson and Phong Q. Nguyen, eds.), LNCS, vol. 7881, Springer, Heidelberg, May 2013, pp. 626–645. 2.2, 2.2

[GMM+16] Sanjam Garg, Eric Miles, Pratyay Mukherjee, Amit Sahai, Akshayaram Srinivasan, and Mark Zhandry, Secure obfuscation in a weak multilinear map model , TCC 2016-B, Part II (Martin Hirt and Adam D. Smith, eds.), LNCS, vol. 9986, Springer, Heidelberg, October / November 2016, pp. 241–268. 1

[GMNO18] Rosario Gennaro, Michele Minelli, Anca Nitulescu, and Michele Orrù, Lattice-based zk-SNARKs from square span programs , ACM CCS 2018 (David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, eds.), ACM Press, October 2018, pp. 556–573. 1

[Gol11] Oded Goldreich, Candidate one-way functions based on expander graphs , Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation: In Collaboration with Lidor Avigad, Mihir Bellare, Zvika Brakerski, Shafi Goldwasser, Shai Halevi, Tali Kaufman, Leonid Levin, Noam Nisan, Dana Ron, Madhu Sudan, Luca Trevisan, Salil Vadhan, Avi Wigderson, David Zuckerman (2011), 76–87. 1

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan, Trapdoors for hard lattices and new cryptographic constructions , 40th ACM STOC (Richard E. Ladner and Cynthia Dwork, eds.), ACM Press, May 2008, pp. 197–206. 3.2.2

[Gro16] Jens Groth, On the size of pairing-based non-interactive arguments , EUROCRYPT 2016, Part II (Marc Fischlin and Jean-Sébastien Coron, eds.), LNCS, vol. 9666, Springer, Heidelberg, May 2016, pp. 305–326. 1, 2.3, 5

[GW11] Craig Gentry and Daniel Wichs, Separating succinct non-interactive arguments from all falsifiable assumptions , 43rd ACM STOC (Lance Fortnow and Salil P. Vadhan, eds.), ACM Press, June 2011, pp. 99–108. 1

[HJ16] Yupu Hu and Huiwen Jia, Cryptanalysis of GGH map , EUROCRYPT 2016, Part I (Marc Fischlin and Jean-Sébastien Coron, eds.), LNCS, vol. 9665, Springer, Heidelberg, May 2016, pp. 537–565. 3.2.2, B.0.2

[JLS21] Aayush Jain, Huijia Lin, and Amit Sahai, Indistinguishability obfuscation from well-founded assumptions , ACM Press, 2021, pp. 60–73. 1

[KF17] Paul Kirchner and Pierre-Alain Fouque, Revisiting lattice attacks on overstretched NTRU parameters , EUROCRYPT 2017, Part I (Jean-Sébastien Coron and Jesper Buus Nielsen, eds.), LNCS, vol. 10210, Springer, Heidelberg, April / May 2017, pp. 3–26. B.0.1

[Kil92] Joe Kilian, A note on efficient zero-knowledge proofs and arguments (extended abstract) , 24th ACM STOC, ACM Press, May 1992, pp. 723–732. 1

[KPY19] Yael Tauman Kalai, Omer Paneth, and Lisa Yang, How to delegate computations publicly , 51st ACM STOC (Moses Charikar and Edith Cohen, eds.), ACM Press, June 2019, pp. 1115–1124. 1

[Mic00] Silvio Micali, Computationally sound proofs , SIAM J. Comput. **30** (2000), no. 4, 1253–1298. 1

[MR04] Daniele Micciancio and Oded Regev, Worst-case to average-case reductions based on Gaussian measures , 45th FOCS, IEEE Computer Society Press, October 2004, pp. 372–381. 7

[MSZ16] Eric Miles, Amit Sahai, and Mark Zhandry, Annihilation attacks for multilinear maps: Cryptanalysis of indistinguishability obfuscation over GGH13 , CRYPTO 2016, Part II (Matthew Robshaw and Jonathan Katz, eds.), LNCS, vol. 9815, Springer, Heidelberg, August 2016, pp. 629–658. 1, 3.4, 3.4.1, 5.2, B.0.3

[Pel18] Alice Pellet-Mary, Quantum attacks against indistinguishablility obfuscators proved secure in the weak multilinear map model , CRYPTO 2018, Part III (Hovav Shacham and Alexandra Boldyreva, eds.), LNCS, vol. 10993, Springer, Heidelberg, August 2018, pp. 153–183. B.0.2, B.2, 6

[PHGR13] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova, Pinocchio: Nearly practical verifiable computation , 2013 IEEE Symposium on Security and Privacy, IEEE Computer Society Press, May 2013, pp. 238–252. 2.2, 2.3

[PP19] Thomas Pornin and Thomas Prest, More efficient algorithms for the NTRU key generation using the field norm , PKC 2019, Part II (Dongdai Lin and Kazue Sako, eds.), LNCS, vol. 11443, Springer, Heidelberg, April 2019, pp. 504–533. 2.1.3

[SV10] Nigel P. Smart and Frederik Vercauteren, Fully homomorphic encryption with relatively small key and ciphertext sizes , PKC 2010 (Phong Q. Nguyen and David Pointcheval, eds.), LNCS, vol. 6056, Springer, Heidelberg, May 2010, pp. 420–443. 2.1.3

A Sum collision-free sets

In this section we construct a set that satisfies the combinatorial constraints needed to defy generic annihilation attacks. In particular, for a given n, q, we need a set with n elements such that there are no "addition collisions" modulo q: two pairs of elements which add up to the same quantity. One such natural set is powers of 2. However, we want a set whose elements are upper bounded in value by a small polynomial in n. We show how to construct such a set with values in $O(n^2)$.

Theorem 13 *Let $n, q \in \mathbb{Z}$ with n prime, $2n$ coprime to q and $q > 3n^2$. There is a set S of size n with entries in $[1, 3n^2]$, such that for all $x, y, z, w \in S$, with $\{x, y\} \neq \{z, w\}$ we have $x + y \neq z + w \ (mod\ q)$.*

Proof: We construct S as the set of the following elements $s_i \in [1, 3n^2]$ with $i \in [1, n]$:

$$s_i = 2ni + [i^2]_n,$$

where $[.]_n$ means the remainder when dividing by n.

For the sake of contradiction, consider indices (i, j, i', j'), such that: $s_i + s_j = s_{i'} + s_{j'}$. Therefore, $2n(i + j) + [i^2]_n + [j^2]_n = 2n(i' + j') + [i'^2]_n + [j'^2]_n$. Both the quantities $i + j$ and $[i^2]_n + [j^2]_n$ (and their primed versions) are less than $2n$. So we can interpret the above equation as a $2n$-radix system. This means:

$$i + j = i' + j' \ (mod\ n)$$

$$i^2 + j^2 = i'^2 + j'^2 \ (mod\ n).$$

For any fixed i, j, there are at most two solutions for i', j' mod n. This is because, using the first linear equation we can eliminate j' from second equation and we are left with a non-trivial quadratic equation in i'. Therefore, $(i', j') = (i, j)$ or (j, i), as these are definitely the two solutions (and using the fact that all indices i, j etc. are less than n). □

Generalization. We note that this technique generalizes to settings where we consider k-tuples of sums, instead of just pairs. We briefly describe the $k = 3$ case at an intuitive level.

Now we take $q > 10n^3$. Construct set S with elements:

$$s_i = i * 9n^2 + [i^2]_n * 3n + [i^3]_n,$$

for $i \in [1, n]$.

Now the condition $s_i + s_j + s_k = s'_i + s'_j + s'_k$ translates to:

$$(i + j + k) * 9n^2 + ([i^2]_n + [j^2]_n + [k^2]_n) * 3n + ([i^3]_n + [j^3]_n + [k^3]_n) =$$

$$(i' + j' + k') * 9n^2 + ([i'^2]_n + [j'^2]_n + [k'^2]_n) * 3n + ([i'^3]_n + [j'^3]_n + [k'^3]_n)$$

Interpret the quantities in a $3n$-radix system, with each co-efficient being $< 3n$. Therefore, they are individually equal. So we have:

$$i + j + k = i' + j' + k' \ (mod\ n)$$

$$i^2 + j^2 + k^2 = i'^2 + j'^2 + k'^2 \ (mod\ n)$$

$$i^3 + j^3 + k^3 = i'^3 + j'^3 + k'^3 \ (mod\ n)$$

By Bezout's theorem, there are at most $1 * 2 * 3 = 6$ solutions. We know that all the 6 permutations of (i, j, k) are solutions, so that exhausts all solutions. Hence there are no collisions.

B Additional Cryptanalysis

Proving the security of constructions based on GGH13 (and multilinear maps in general) has been a subtle art. Showing any of the hardness assumptions in Section **??** to be false would undermine the security of our QAP, and therefore here we consider the attacks on the underlying multilinear map. Given the nature of our hardness assumptions, and our desire for our VC scheme to be plausibly post quantum, we are most concerned with lattice reduction attacks (which recover z) and combinations of zeroizing attacks and short principal ideal problem solvers (which would plausibly recover g), see below. There are three broad classes of attacks that one can hope to avoid.

B.0.1 Lattice Reduction.

The first, and simplest to avoid, consists of "overstretched NTRU" style attacks, which see the inherent NTRU like structure in GGH13, in particular that the quotient of two encodings, e_1, e_2, i.e. e_1/e_2, is a fraction of short elements in R_q. The works of [ABD16, CJL16] and [KF17] consider the problem in subfields and subrings respectively, and conclude that, roughly, when q is of the order $2^{\sqrt{n}}$ efficient lattice reduction attacks can be mounted to recover short multiples of $e_1, e_2 \in K$ from $e_1/e_2 \in R_q$. This results in the recovery of the secret parameter z, see [ABD16, Sec. 4.2]. To avoid these attacks it is enough to increase n by a polynomial factor and ensure $n = \Omega(\lambda(\log q)^2)$, as we do in Section 3.3.

B.0.2 The (Short) Principal Ideal Problem.

The problem of being given some principal ideal $\mathfrak{a} = \langle r \rangle, r \in R$ and recovering some generator $ur, u \in R^\times$ (possibly also with ur short) has received a lot of attention both classically and quantumly. More precisely, if r is sampled from some distribution D over R the principal ideal problem (PIP) is to recover, given any $\mathbb{Z}$ basis of $\langle r \rangle$ (seen as a sublattice of R), any generator ur. In the prime power cyclotomic case, the short principal ideal problem (sPIP) is to recover $\pm rX^i$ for any i [CDPR16, Pel18]. The short principal ideal problem is pertinent to the study of multilinear maps as, given the ability to zero test, various quantities relating to $\langle g \rangle, \langle hg \rangle$, and $\langle h \rangle$ are potentially available to an adversary (in roughly decreasing order of severity). While much work has considered how to restrict this information, see below, this represents one of the fundamental obstacles hindering the development of secure multilinear maps, namely that the zero test leaks more than a single bit of information – more than just whether an encoding is zero or not.

Classically, the algorithm of [BEF+17] can recover a short generator with complexity approximately $2^{\sqrt{n}+o(1)}$, but such attacks are exponential time in our parameter setting as $n = \Omega(\lambda^{2+\epsilon})$. Quantumly, such attacks can be more devastating. The works [EHKS14, BS16, CGS14, CDPR16] ultimately result in a polynomial time quantum algorithm for sPIP in prime power cyclotomics, under certain conditions. In particular [BS16] solves PIP in quantum polynomial time in an arbitrary number field, building on the work of [EHKS14]. Then [CDPR16, Thm. 4.1], building on an observation of [CGS14] about power of two cyclotomics, show how to recover an sPIP solution from the output of [BS16] when r is sampled from a discrete Gaussian over any prime power cyclotomic.

How exactly this quantum polynomial time algorithm affects security depends on what is available to an adversary, which is closely related to whether different forms of encodings of zero can be exploited, see "zeroizing attacks" below. For example an attack against GGH13 made use of knowledge of $\langle g \rangle$ to classically attack multipartite key exchange [HJ16]. Given a quantum adversary, knowledge of $\langle g \rangle$ may instead plausibly result in the recovery of g and therefore a total break of the scheme (we note there is a subtlety here, in that the Gaussian parameter of the distribution g is sampled from is slightly too small for the algorithm of [CDPR16]).

More pertinent to us is the fact that the worker can form many level κ zeros by evaluating the outsourced function F on inputs of its choosing, forming an honest proof and performing the isZero tests. There are also level 2 zeros available from combinations of elements in $\mathsf{EK}_F \cup \mathsf{VK}_F$ that are more efficient for the worker to form, but we note that even if none such existed, it can always resort to honest evaluation of F on multiple inputs.

From this, and following [Pel18], we see that, if we included it, we could recover h in quantum polynomial time and form (quantities similar to) zero tests at levels $\hat{\kappa} = \nu\kappa$, for ν an integer greater than zero. As we do not include it, we may also form these new zero tests. We discuss these zero testers further, and the forms of these $\hat{\kappa}$ zeros in Section B.2.

B.0.3 Zeroizing Attacks.

This style of attack makes use of the extra information given to an adversary after zero testing encodings of zero. In a simple case, we see that zero testing a level κ encoding of zero, $[rg]z^{-\kappa}$ reveals the element $[r]$, which will equal r as it is short and therefore not reduced modulo q. When level $\kappa' < \kappa$ encodings of zero are given, in our construction level 1 encodings, products can be taken to form $[rg^2]z^{-\kappa}$ so that zero testing returns $[rg] = rg$ over R. Ultimately [GGH13, Sec. 6.3.1] this leads to knowledge of $\langle g \rangle$, which we must avoid to claim any notion of post quantum security.

This style of attack can be extended in two ways; firstly it may be the case that elements of $\mathsf{EK}_F \cup \mathsf{VK}_F$ encode zero, and secondly via the "annihilation" style attacks, first introduced by [MSZ16], which recover elements of $\langle g \rangle$ without requiring level κ' encodings of zero. The first extension is considered in Section B.1. The second extension is considered in Section 3.4.

To capture this class of annihilation attacks, the weak multilinear map model was introduced [MSZ16] which assumes an ideal multilinear map and allows an adversary to make certain manipulations of encodings and the values returned from zero tests. This model is important because it provides concrete theoretical targets for constructions, however it is not without weaknesses. For example, it classically captures polynomial evaluations on zero tested values, but does not capture averaging attacks that are available due to statistical leaks from zero testing [DP18]. We discuss such "statistical" zeroizing attacks in Section B.3.

B.1 Avoiding Zeros

Given the discussion above it is important to avoid elements of $\mathsf{EK}_F, \mathsf{VK}_F$ being low level (henceforth level 1, given our scheme) zeros. Given the construction of Pinocchio, however, many will naturally occur. Indeed, a wire in a circuit encoded as a QAP can possibly be the output wire of a multiplication gate and the left (exclusive) or right input to a single further multiplication gate. Thus, for this wire, one of $v'_k(x), w'_k(x), y'_k(x)$ will be identically zero. To account for this, during the setup the client can choose a symbol, say $\perp$, to use in place of any level 1 encoding of zero. The worker or verifier, when forming the proof elements or verifing the proof respectively, do not include contributions from any $\perp$ present in EK_F or VK_F respectively.

It could also be the case that some values formed during the execution of the protocol are level 1 encodings of zero. We show below that this is unlikely. Every element $\tilde{e}$ encoded in $\mathsf{EK}_F \cup \mathsf{VK}_F$ is such that $\varphi(e) = (e', 0)$, while g is chosen such that $(\varphi \circ \xi)(g) = (g_1, g_2), g_2 \neq 0$. This is not enough to say that sums and products of encoded elements will never land in the ideal generated by g, but we can show that the index of relevant ideals is superexponential in λ. More precisely we have two ideals of R, $\mathcal{I} = \langle g \rangle$ and $\mathcal{J} = \langle \tilde{\ell}(X), p \rangle$. The ideal $\mathcal{J}$ represents all $r \in R$ such that $(\varphi \circ \xi)(r) = (r', 0)$. When considering the underlying additive groups, we will show $[(\mathcal{J}, +) : (\mathcal{I} \cap \mathcal{J}, +)] = N_{K/\mathbb{Q}}(g) \approx n^n = \lambda^{\Omega(\lambda^2)}$. This says that superexponentially few elements of $\mathcal{J}$, i.e. elements of R that can be formed from our encodings, are also elements of $\mathcal{I}$, and therefore encodings of zero.

Lemma 14 *Let $g \in R$ be a prime element. Let $\mathcal{I} = \langle g \rangle, \mathcal{J} = \langle \tilde{\ell}(X), p \rangle$. Then $\mathcal{I}, \mathcal{J}$ are maximal ideals.*

Proof: All rings of integers of number fields are Dedekind domains, where prime ideals are maximal. As g is a prime element, $\mathcal{I} = \langle g \rangle$ is a prime ideal, and therefore maximal. Consider the natural surjective ring morphism $\mathbb{Z}[X] \to R$. Surjective ring morphisms map maximal ideals to maximal ideals. The maximal ideals of $\mathbb{Z}[X]$ are of the form $\langle f(X), p \rangle$ with p prime and $f(X)$ irreducible modulo p. By construction $\ell(X)$ is irreducible in $\mathbb{F}_p[X]$, so $\langle \tilde{\ell}(X), p \rangle$ is maximal in $\mathbb{Z}[X]$ and therefore also in R. Hence $\mathcal{J}$ is also maximal. □

Given a group G and subgroups $K \leq H \leq G$ we have $[G : K] = [G : H] \cdot [H : K]$. Setting $G = R, H = \mathcal{J}, K = \mathcal{I} \cap \mathcal{J}$, where we consider the additive group structure of rings or ideals as required, we see $[\mathcal{J} : \mathcal{I} \cap \mathcal{J}] =$

$[R\colon \mathcal{I}\cap\mathcal{J}]/[R\colon \mathcal{J}]$. Similarly we have $[R\colon \mathcal{I}] = [R\colon \mathcal{I}\cap\mathcal{J}]/[\mathcal{I}\colon \mathcal{I}\cap\mathcal{J}]$. The ideal norm $N_I(\mathcal{I}) = [R\colon \mathcal{I}] = N_{K/\mathbb{Q}}(g)$ as $\mathcal{I}$ is principal and generated by g, so if $[R\colon \mathcal{J}] = [\mathcal{I}\colon \mathcal{I}\cap\mathcal{J}]$ then $[\mathcal{J}\colon \mathcal{I}\cap\mathcal{J}] = N_{K/\mathbb{Q}}(g)$.

We prove this with the following fact, $[H\colon H\cap K] \leq [G\colon K]$ with equality if $HK = G$. Note here that $HK = \{hk\colon h\in H, k\in K\}$ is a product of group subsets. Our group operation is addition, explicitly if $G = (R,+), H = (\mathcal{I},+), K = (\mathcal{J},+)$ then HK is not the product of ideals $\mathcal{I}\mathcal{J}$ but rather $\{i+j\colon i\in\mathcal{I}, j\in\mathcal{J}\}$.

Ideals $\mathcal{I},\mathcal{J}$ such that there exist $i\in\mathcal{I}, j\in\mathcal{J}, i+j=1$ are called coprime. Coprime ideals are such that $\{i+j\colon i\in\mathcal{I}, j\in\mathcal{J}\} = R$. Distinct maximal ideals are coprime; our ideals $\mathcal{I},\mathcal{J}$ are distinct as $g\notin\mathcal{J}$, and are therefore coprime. We therefore satisfy the condition on equality above, so $[R\colon \mathcal{J}] = [\mathcal{I}\colon \mathcal{I}\cap\mathcal{J}]$ and $[\mathcal{J}\colon \mathcal{I}\cap\mathcal{J}] \approx \lambda^{\Omega(\lambda^2)}$.

B.2 Alternative Zero Testing and Level $\hat{\kappa}$ Zeros

In [GGH13, Sec. 6.3.3] a method is given for creating zero tests at levels $\hat{\kappa} = \nu\kappa$ using low level zeros. In the absence of low level zeros [Pel18] creates new zero testers from one of our form, i.e. $p_{zt} = z^{\kappa}g^{-1}$, as $p_{zt}^{(\hat{\kappa})} = p_{zt}^{\nu} = z^{\nu\kappa}g^{-\nu}$.[6] This new zero tester can only test level $\hat{\kappa}$ zeros of particular forms. Indeed, $p_{zt}^{(\hat{\kappa})}$ can only test level $\hat{\kappa}$ zeros of the form $[rg^{\nu}]z^{-\nu\kappa}$.

We make the assumption that zeros of this form can only be created from the products of level 2 zeros with other encodings. We show that under our parameters, for some small class of obtainable level 2 zeros, we cannot create any N such level 2 zeros without requiring at least N different encodings. Therefore each such set of relations has at least N unknown elements r, where encodings are of the form $[a+rg]z^{-1}$. We note that we do not consider level 2 zeros which are weighted sums of other level 2 zeros.

The class of level 2 zeros obtainable from $\mathsf{EK}_F \cup \mathsf{VK}_F$ we consider is the zeros of the form

$$p^j\left(\varphi_{i,0}\cdot P_i\right) - \left(\varphi_{i+j,0}\cdot P_{i+j}\right)$$

for $j\in[\eta-1], i\in[\eta-j]$. While for the i,j given above, there are $O(\eta^2)$ such zeros and only $O(\eta)$ different encodings, we note that from the analysis at the end of Section 7 and Lemma 3 we have $q\approx p^3$ and so may, in reality, only take $j\in[3]$ and expect the zero test to pass. This leads to fewer relations than encodings, and as such we conclude that this class of level 2 zeros does not aid annihilation attacks.

In general, we leave as a conjecture that the extra power of the various $p_{zt}^{(\hat{\kappa})}$ does not aid annihilation attacks 3.4.

B.3 "Statistical" Zeroizing

In the work of [DP18] the statistical leakage of zero tests allowed in a certain model are considered. They use statistical techniques to argue that all previously considered encoding techniques leak quantities related to either $A(h/g)$ or $A(hz^{\kappa}/g)$, and that the "simplistic" encoding of [GGH13, Sec. 4.1] allows for a full recovery of $A(h/g)$,[7] which represents a break in the weak multilinear map model. The model they use (adapted to the symmetric setting) considers uniformly chosen plaintexts $a_1, a_1', \ldots, a_m, a_m'$ such that $\sum_{i\in[m]} a_i\cdot a_i' = 0 + \mathcal{I}$ so that u_i encoding a_i at some level $\kappa' < \kappa$ and u_i' encoding a_i' at level $\kappa-\kappa'$ are such that $\sum_{i\in[m]} u_i\cdot u_i'$ is a level κ encoding of zero. They then average such top level zeros over $\kappa'\in[\kappa-1]$. Given their findings on the above the authors present a new encoding method, called the "compensation" method, which statistically leaks nothing in their model. While we do not match the model of [DP18] exactly, because our plaintexts are not sampled uniformly, we note that any top level zero created from $\mathsf{EK}_F\cup\mathsf{VK}_F$ will be of the form considered, with $\kappa = 2, \kappa' = 1$, and the low multilinearity parameter of our scheme gives few levels to average over. Furthermore, the compensation method gives the shortest encodings of any proposed encoding method, and we therefore choose to use it.

[6] When the element h is present and known, as in [Pel18], then one does $p_{zt}^{\nu}h^{-\nu}$.

[7] In our case, $A(1/g), A(z^{\kappa}/g)$.