Sublinear Pairing-based Arguments with Updatable CRS and Weaker Assumptions

Alonso González[1⋆] and Carla Ràfols[2⋆⋆]

[1] ENS de Lyon, Laboratoire LIP (U. Lyon, CNRS, ENSL, INRIA, UCBL), France.
[2] Universitat Pompeu Fabra and Cybercat, Barcelona, Spain.

Abstract. The need for structured and trusted common parameters is often cited as one of the major drawbacks of pairing-based SNARKs. Although multiparty computation techniques can be used to address this, the resulting parameters are circuit dependent and this costly process must be repeated for every circuit. Recent proposals to switch to a weaker updatable model for parameter generation are not yet sufficiently efficient. We propose a new model for updatability which generates the common reference string in two phases, each of them updatable: in the first phase, parameters are generated for a set of universal quadratic constraints and in the second phase specific circuit dependent parameters which impose some affine constraints can be derived from them non-interactively. We propose a concrete construction based on (but more efficient than) Pinocchio.
An additional contribution of the paper is to obtain a very efficient argument for verifiable computation using the same design principles which is based on weaker assumptions. The communication is $\approx 4d$ group elements and verifying a proof requires computing $\approx 4d$ pairings and $O(n+d)$ exponentiations, where n is the input size and d the circuit depth. While the argument for the quadratic constraints is based on standard falsifiable assumptions, the argument for the linear constraints is based on a very ad-hoc assumption about certain properties of arguments of membership in linear spaces.

1 Introduction

Zero-Knowledge proofs, and in particular, non-interactive ones have played a central role in both the theory and practice of cryptography. A long line of research [33,37,26,25,22,16,30] has led to efficient pairing-based *Succint Non-interactive ARguments of Knowledge* or SNARKs. These arguments are *succint*, in fact, they allow to prove that circuits of arbitrary size are satisfied with a constant-size proof. They are also extremely efficient concretely (3 group elements in the best construction for arithmetic circuits [23]).

Zero-Knowledge SNARKs have been famously used to provide privacy of transactions in the cryptocurrency setting, where they are now deployed both in ZCash and Ethereum. Despite of significant research in finding alternatives to zk-SNARKs in these settings (for example BulletProofs [9]), none of them can really compete with the extremely cheap verification procedure of SNARKs which is a fundamental requirement in this setting.

The use of SNARKs with cryptocurrencies has motivated a lot of research and optimizations in their implementation and also in theoretical aspects (e.g. [40,4,5,6]). Despite all the improvements, some aspects of current constructions of SNARKs are still unsatisfactory. Probably the most problematic unsolved issue is their reliance on long trusted structured parameters. Although this can be solved with multiparty computation techniques, as famously done in the ZCash ceremony [8], the fact that the parameters are circuit dependent means that this costly process must be repeated for every circuit. To deal with this issue, recent research [24] designs a SNARK in which the common reference string can be updated by any party non-interactively in a verifiable way, resulting in a properly generated common reference string (where the simulation trapdoor is unknown to all parties) if at least one party is honest. Further, the common reference string is universal and can be used for arbitrary circuits. However, the original construction proposed in [24],

⋆ This author was supported in part by the French ANR ALAMBIC project (ANR-16-CE39-0006).
⋆⋆ The research leading to this article was supported by a Marie Curie "UPF Fellows" Postdoctoral Grant and by the Catalan Agency for Management of University and Research Grant 2017 SGR 1739.

as well as more recent follow-up work [36], are still not sufficiently practical. The main contribution of this paper is to construct an efficiently updatable scheme for both arithmetic and boolean circuits, with very little overhead in performance in all aspects and with similar security properties as the state-of-the-art.

Our second contribution improves the security guarantees provided by SNARKs. One of the important motivations of SNARKs is their application to the problem of verifiable delegation of computation, in which a resource constrained verifier asks the prover to do a costly computation and requires a proof that it is done correctly. Although it is well known that succintness cannot be achieved under falsifiable assumptions [17], such impossibility results do not apply in this setting, where the input of the circuit is public, and it could be in principle possible to achieve very efficient constructions based on falsifiable assumptions. Using the same basic design principle as for our updatable SNARK, we makes a significant step towards this direction by giving a scheme which is quite efficient ($\approx 4d$ source group elements for the proof and $O(n+d)$ verification complexity, for d the depth of the circuit and n the size of the input) for arithmetic circuits. Our solution is still not satisfactory as to rule out a certain class of attacks, it makes a (falsifiable) but ad-hoc and circuit dependent assumption on some properties of QANIZK arguments of membership in linear spaces [29,35,30,34,20,13].

We now discuss in more detail our contributions, as well as the comparison with previous work and our techniques.

1.1 First Contribution: an Updatable and Universal zk-SNARK

Our Results. Arithmetic Circuit Satisfiability can be encoded as a system of quadratic and linear constraints. One the one hand, quadratic equations encode correct evaluation of the multiplication gates while the linear constraints encode the linear relations of the output of multiplication gates at level i with the input of gates at level $i+1$.

Quadratic Arithmetic Programs [16] are a particular way to encode circuit satisfiability following this approach but in which the set of linear equations is empty because the consistency checks and linear relations are absorbed into many quadratic equations. For instance, given a circuit $\phi : \mathbb{Z}_p^3 \to \mathbb{Z}_p$ such that the first multiplication gate computes $(2x_1 + x_2)x_3$ and the second gate multiplies the output of this gate by $x_1 + 3$, the system of constraints is:

$$(2c_1 + c_2)c_3 = c_4 \qquad c_4(c_1 + 3) = c_5.$$

The quadratic equations are then proven simultaneously in constant-size by writing them as a polynomial equation which is verified at a single point encoded in the common reference string. We refer to this as polynomial aggregation technique. This way of encoding the circuit has the advantage that additive gates are essentially "for free".

Our approach instead is to use the encoding considered in [7], which is also the natural encoding if one wants to prove linear and quadratic equations separately. Namely, if c_1, c_2, c_3 are the inputs and a_i, b_i, c_i, $i = 4, 5$ are respectively the left, right and output wires of the multiplication gates, we can encode the circuit as:

– $a_i b_i = c_i$, $i = 4, 5$ (quadratic constraints)
– $a_4 = 2c_1 + c_2, b_4 = c_3, a_5 = c_4, b_5 = c_1 + 3$ (affine constraints).

The quadratic constraints can be proven in constant size with the same polynomial aggregation technique. On the other hand, the affine constrains can be proven using a specially tailored proof system such as the very efficient quasi-adaptive NIZK proof systems for membership in linear spaces [29,30].

The set of quadratic equations is independent of the circuit. On the other hand, the crs associated to the linear constraints is circuit dependent, as the crs of the argument of membership in linear spaces depends on the generating matrix of the space. Therefore, it makes sense to define crs generation as a two step process in which first a general quadratic crs is defined, and then a specialized circuit dependent crs is derived.

This approach turns out to be quite practical. The "quadratic" universal crs can be trivially seen to be updatable (it essentially includes only all powers of a secret value, i.e. all the monomials in the terminology of [24]). On the other hand, the "affine part" of the crs is a linear transformation of the generating matrix of the linear space, and is easy to additively update.

An important point is that the argument of membership in linear spaces is used as an argument of knowledge (in fact, the linear space in question is all of the space, and the statement is trivially true). It is not hard to prove that in the generic group model these arguments have knowledge soundness, as it was already proven in [13] for a specific choice of distribution of a linear space.

Our resulting argument which combines the proof of quadratic and linear constraints ends up being similar to Pinocchio. However, the knowledge soundness of the linear subspace argument allows to extract a witness from it and eliminate some terms which are redundant, resulting in a proof size of $(5\mathbb{G}_1, \mathbb{G}_2)$ as opposed to the original scheme in asymmetric groups $(7\mathbb{G}_1, \mathbb{G}_2)$ [40,6].

Previous Work. Our arguments for arithmetic and boolean circuit satisfiability are a few elements above the state-of-the-art. A detailed performance comparison is given in Table 1. The most efficient schemes in terms of proof size are [23] for arithmetic circuits

Scheme	CRS size		Runtime		Proof size	Setup	Ass.
	Universal	Circ. spc.	Prover	Verifier			
[21]	$O(m^2)$	-	$O(m^2)$	$36P + mE$	42	↺	KOE
Pinocchio[6]	-	$8m + 7n + n_p$	$8m + 7n - n_p$	$12P + n_pE$	8	✗	KOE
[23]	-	$4m + n$	$5m + n - n_p$	$3P + n_pE$	3	✗	GG
Bulletproofs[9]	$m/2$	-	$8m$	$4mE$	$O(\log(m))$	✓	RO
[24]	$O(m^2)$	$O(m + n - n_p)$	$O(m + n - n_p)$	$5P + n_pE$	3	↺	GG
Sonic[36][a]	$4d$	$12m$	$60m(18m)$	$13P(9P)$	20(7)	↺	GG,RO
This work	$2(m + n)$	$n + m + 7$	$2n + 6m + 4$	$9P + n_pE$	6	↺	GG

Table 1. Comparison of our results with other SNARKS. 'Circ. spc.' indicates the size of the circuit dependent crs and 'Ass.' the assumption used for proving security of the corresponding scheme. Sizes are given in #group elements in both base groups and the prover's runtime in number of exponentiations in any of the base groups. m stands for #multiplication gates, n #inputs, n_p #public inputs. P is the cost of computing a pairing, E the cost of a exponentiation. A mark ✓ means that the crs can be fully adversarially chosen, ✗ means that the crs must be chosen by a trusted party, and ↺ means that any party can update the crs. KOE stands for a "knowledge of exponent" type of assumption, GG for generic group model, RO for random oracle model.

[a] The numbers in parenthesis are only achieved for batched proofs assuming the availability of *helpers* which do part of the computation.

Given that Pinocchio was implemented and deployed, and our argument improves its performance, and that updatability and universal crs seem great advantages in practice, there are reasons to argue that the approach is practical and might compensate for the overhead w.r.t state-of-the-art.

Previous approaches to deal with the problem of generating the long structured parameters needed for SNARK were using multiparty computation techniques [5]. Several authors have studied the effects on security of relaxing the trust on parameter generation. Bellare et al. [3] give security definitions for subversion resistant SNARKs, in which parameters are controlled by an adversary. They prove that subversion zero-knowledge and subversion soundness cannot be achieved simultaneously. [1,15] study how to prove that SNARKs are subversion zero-knowledge.

The notion of updatable common reference string was proposed by Groth et al. in [24]. Such a notion relaxes the trust in the common parameter generation step by allowing each party to non-interactively contribute a share of the final secret. They construct a scheme which is subvertible zero-knowledge and updatable sound and has a universal crs. The size of the crs is quadratic in the (maximal) size of any circuit to be proven, although for efficiency, a specific circuit dependent linear crs can be derived incurring in an additional cubic cost for each derivation.

In concurrent work independent of ours, Maller et al. [36] proposed a new updatable scheme. The starting point (separating affine and quadratic constraints, inspired by [7]) is similar to ours, except that their techniques are a combination of the interactive setting ([7,9]) and SNARKs. The resulting scheme is less efficient than ours specially in terms of prover runtime and proof size. They also give a batched version, i.e. the same statement is proved many times, in a stronger model where helpers are introduced — untrusted parties which are willing to give out computation power to help with some part of the computation — and the resulting scheme is slightly more efficient in terms of verification but is still less efficient in the other parameters.

1.2 Second Contribution: Towards Efficient Verifiable Computation based on Falsifiable Assumptions

Our Results. We construct an argument for proving that an arithmetic circuit $\phi : \mathbb{Z}_p^n \to \mathbb{Z}_p^{n'}$ is correctly evaluated, of size $(3d+2)\mathbb{G}_1 + (d+2)\mathbb{G}_2$ group elements and verification requires $4d+6$ parings and $O(n+d)$ exponentiations, where d is the depth of the circuit and n the size of the input. For boolean circuits, the argument can be made zero-knowledge and the resulting proof has size $O((n-n_{pub})+d)$, where n_{pub} is the number of public inputs, while verification remains the same.

Proving that a circuit is correctly evaluated under falsifiable assumptions is easy if we eliminate the requirement that the proof size and the verifier complexity are sublinear in the circuit size. On the other hand, the tautological assumption "the scheme is sound" is falsifiable, as one can efficiently check if an adversary breaks soundness by verifying if the computation is incorrect while the proof is accepted. In particular, one can use a SNARK to prove correctness of computation and make the (falsifiable) assumption that it is sound.

Obviously, this is not so interesting from a theoretical point of view. The challenge is to reduce the proof that the scheme is sound into falsifiable assumptions as standard as possible. We achieve this objective only partially: we reduce soundness to 3 types of attacks, an attack against a certain q-assumption in bilinear groups, a kernel MDDH Assumption [38] and finally an attack against an ad-hoc and circuit dependent (but falsifiable) assumption which relates to certain properties of the QANIZK argument for membership in linear spaces. To give more confidence in the latter, we show that it is generically equivalent to the discrete logarithm. We think it should be possible to eliminate this assumption in future work.

A first approach to prove correct computation under falsifiable assumptions is simply to encode circuit satisfiability as a set of quadratic and affine constraints as described before. Leaving affine constraints aside, if one aims for sublinear proofs some kind of aggregation mechanism for quadratic equations needs to be used. We are only aware of one work with such results (in the crs model), [20] and it requires perfectly binding commitments to the witness (a valid assignment to the equations). In particular, this means the resulting proof would be linear in the circuit size.

Our solution is to divide the set of constraints into d sets of quadratic and affine constraints, one per multiplicative level of the circuit. Namely, if $\phi : \mathbb{Z}_p^n \to \mathbb{Z}_p^{n'}$ is an arithmetic circuit of depth d, correct evaluation at level i is expressed by the following system:

– (quadratic constraints) $c_{i,j} = a_{i,j} b_{i,j}$ for $j = 1, \dots, n_i$.
– (affine constraints) $a_{i,j}, b_{i,j}$ are affine combinations of output wires of previous levels.

For quadratic constraints, our technical innovation is to eliminate the need for binding commitments. More specifically, given adversarially chosen commitments (the commitments need to be shrinking but not hiding, as the argument does not need to be zk and it is important that they are deterministic) L_i (resp. R_i, O_i) to all the left (resp. right, output) wires at level i, we give an argument with constant-size verification which proves:

$$\text{If } (\boldsymbol{a}_i, \boldsymbol{b}_i, L_i, R_i, O_i) \text{ is such that } L_i,\ R_i \text{ open to } \boldsymbol{a}_i, \boldsymbol{b}_i \text{ then } O_i \text{ opens to } \boldsymbol{c}_i = \boldsymbol{a}_i \circ \boldsymbol{b}_i.$$

We think of this building block as a "quadratic knowledge transfer argument", as it shows that if an adversary knows an opening for left and right wires, it also knows an opening of the output wires at next level. Formally, this property is formalized as a promise problem because the verifier of the argument never checks that L_i, R_i open to $\boldsymbol{a}_i, \boldsymbol{b}_i$ (otherwise the verification of the argument would be linear in the witness). Using a quadratic arithmetic program encoding of the quadratic constraints we construct a proof which can be proven secure under a certain q-assumption.

With this building block, the problem of constructing the argument is reduced to arguing that left and right wires are correctly assigned, i.e. proving that affine constraints are satisfied. We build a "linear knowledge transfer" argument with constant verification time showing that:

Given an opening of the commitments to the output wires $O_1, \ldots, O_i$ which is consistent with $L_1, \ldots, L_i$ and $R_1, \ldots, R_i$ then it is also consistent with L_{i+1} and R_{i+1}.

Correct evaluation of the circuit can be easily proven by combining these two building blocks. Since the input of the circuit is public, a consistent assignment $O_1, L_1, R_1, \ldots, O_d, L_d, R_d$ of the circuit wires is known by the reduction in the proof of soundness. Any adversary breaking soundness must output another assignment which disagrees with it starting from some level i. If the adversary may output as part of its proof $L_1, \ldots, L_i, R_1, \ldots, R_i, O_1, \ldots, O_{i-1}, O_i^*$, with $O_i^* \neq O_i$. Then the reduction knows openings of L_i, R_i and it can break the soundness of the quadratic knowledge transfer argument. On the other hand, if it sends $L_1, \ldots, L_i^*, R_1, \ldots, R_i^*, O_1, \ldots, O_{i-1}$, where $L_i^* \neq L_i$ or $R_i^* \neq R_i$, then it knows valid openings of O_j until level $i-1$ and it can break the soundness of the "linear knowledge transfer".

To construct the latter, we use again QANIZK arguments of membership in linear spaces. Although soundness of these arguments can be proven under standard assumptions, it turns out that this is not sufficient to rule out a certain kind of attack. The issue arises when one tries to prove that two shrinking commitments open to the same value. Let $\mathbf{A}, \mathbf{B}$ be the commitment keys. If $x = \mathbf{A}\boldsymbol{w}$ and $\boldsymbol{y} = \mathbf{B}\boldsymbol{w}$ are commitments to the same value, obviously

$$\begin{pmatrix} x \\ y \end{pmatrix} \in \mathbf{Im}\begin{pmatrix} \mathbf{A} \\ \mathbf{B} \end{pmatrix}. \tag{1}$$

Let π as a QANIZK proof for (1). In our linear knowledge transfer argument, π should convince the verifier that:

"If $x = \mathbf{A}\boldsymbol{w}$ for some known $\boldsymbol{w}$, and π verifies then $y = \mathbf{B}\boldsymbol{w}$."

The problem is that for any $\boldsymbol{w}'$ such that $x = \mathbf{A}\boldsymbol{w} = \mathbf{A}\boldsymbol{w}'$, an adversary can set $y = \mathbf{B}\boldsymbol{w}'$ and compute π honestly with $\boldsymbol{w}'$. In other words, the adversary could be "switching witnesses" at some point when we are trying to argue that it must always know an opening which is consistent with the input. Our reduction shows that, under fairly standard assumptions in bilinear groups, this is the only possible successful attack.

On the other hand, the "witness switching attack" is easy to rule out, as it requires the attacker to know two openings for x but this breaks the binding property of the first commitment. However, because the commitment is shrinking we do not know how to extract w' to get a reduction to the binding property. We show that such an attack is equivalent to proving with the QANIZK argument for linear spaces that vectors of a certain form are in the space. We prove that generically this is equivalent to breaking the binding property, (or computing discrete logarithms).

In all our subarguments the verification equations are pairing product equations, so they can be made zk with Groth-Sahai proofs [26]. However, our proof uses in a fundamental way that the input of the verification is public. Therefore, it only works when the commitment to the inputs is extractable. The resulting scheme is not practical as it requires bit-by-bit commitments. However, it can be easily extended to boolean circuits with a proof size of $O(n - n_{pub} + d)$, which is an interesting improvement over state-of-the-art, as all constructions in the crs model under falsifiable assumptions are linear in the circuit size (see [25] and concrete improvements thereof, mainly [20]).

Previous Work. As explained in [31], previous work on verifiable delegation on computation can be roughly classified into a) designated verifier schemes [32], b) schemes under very strong assumptions: "knowledge of exponent" type (e.g. [16,40]), generic or algebraic group model (e.g.[23,36]), assumptions related to obfuscation, or homomorphic encryption [39] or c) interactive arguments [19].

Kalai et al. [31], based on [19], constructed the first publicly verifiable non-interactive delegation scheme for boolean circuits from a simple constant size assumption in bilinear groups. Their crs is circuit dependent but they made it universal using a crs for the a universal circuit. [3]. The verifier's runtime is $O((n+d)\mathsf{polylog}(s))$, and the communication complexity is $O(d \cdot \mathsf{polylog(s)})$, where s is the size of the circuit, and in most other parameters it is far from being efficient (crs size, prover complexity).

2 Preliminaries

Given some distribution $\mathcal{D}$ we denote by $x \leftarrow \mathcal{D}$ the process of sampling x according to $\mathcal{D}$. For a finite set S, $x \leftarrow S$ denote an element sampled from the uniform distribution over S. For a probabilistic polynomial time turing machine (PPT) $\mathcal{A}$, we denote by $\mathsf{output} \leftarrow \mathcal{A}(\mathsf{input})$ to the process of sampling enough randomness $r \in \{0,1\}^*$ for running $\mathcal{A}(\mathsf{input}; r)$ and then assign the output to output. We sometimes write $\mathsf{output} \overset{r}{\leftarrow} \mathcal{A}(\mathsf{input})$ to make explicit reference of r, the random coins used by $\mathcal{A}$.

Bilinear Groups. Let $\mathcal{G}$ be some probabilistic polynomial time algorithm which on input 1^λ, where λ is the security parameter, returns the *group key* which is the description of an asymmetric bilinear group $gk := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \mathcal{P}_1, \mathcal{P}_2)$, where $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ are groups of prime order p, the elements $\mathcal{P}_1, \mathcal{P}_2$ are generators of $\mathbb{G}_1, \mathbb{G}_2$ respectively, $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an efficiently computable, non-degenerate bilinear map, and there is no efficiently computable isomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$.

Elements in $\mathbb{G}_\gamma$, are denoted implicitly as $[a]_\gamma := a\mathcal{P}_\gamma$, where $\gamma \in \{1, 2, T\}$ and $\mathcal{P}_T := e(\mathcal{P}_1, \mathcal{P}_2)$. For simplicity, we often write $[a]_{1,2}$ for the pair $[a]_1, [a]_2$. The pairing operation will be written as a product $\cdot$, that is $[a]_1 \cdot [b]_2 = [a]_1[b]_2 = e([a]_1, [b]_2) = [ab]_T$. Vectors and matrices are denoted in boldface. Given a matrix $\mathbf{T} = (t_{i,j})$, $[\mathbf{T}]_s$ is the natural embedding of $\mathbf{T}$ in $\mathbb{G}_s$, that is, the matrix whose (i, j)th entry is $t_{i,j}\mathcal{P}_s$. We denote by $|\mathbb{G}_s|$ the bit-size of the elements of $\mathbb{G}_s$.

$\mathbf{I}_n$ refers to the identity matrix in $\mathbb{Z}_p^{n\times n}$, $\mathbf{0}_{m\times n}$ and $\mathbf{1}_{m\times n}$ the all-zero and all-one matrices in $\mathbb{Z}_p^{m\times n}$, respectively, and $\boldsymbol{e}_i^n$ the ith element of the canonical basis of $\mathbb{Z}_p^n$ (simply $\mathbf{I}$, $\mathbf{0}$, $\mathbf{1}$, and $\boldsymbol{e}_i$, respectively, if n and m are clear from the context).

Lagrange Interpolation. Given an arbitrary set $\mathcal{R} = \{r_1, \dots, r_m\} \subset \mathbb{Z}_p$, we define the jth Lagrange interpolation polynomial as:

$$\lambda_j(X) := \prod_{\ell \neq j} \frac{(X - r_\ell)}{(r_j - r_\ell)}.$$

We define also $t(X) = \prod_{j=1}^n (X - r_j)$. It is a well known fact that given a set of values x_j, $j = 1, \dots, m$, $p(X) = \sum_{j=1}^m x_j \lambda_j(X)$ is the unique polynomial of degree at most $m-1$ such that $p(r_j) = x_j$.

Lagrangian Pedersen Commitments. Given a set of points $\mathcal{R}$ of cardinal m, the Lagrangian Pedersen commitment in $\mathbb{G}_\gamma$ for some $\gamma \in \{1, 2\}$ to a vector $x \in \mathbb{Z}_p^m$ is defined as

$$\mathsf{Com}_{ck}(\boldsymbol{x}; r) = \sum_{i=1}^m x_i[\lambda_i(s)]_\gamma + r[t(s)]_\gamma,$$

[3] There's the technicality that a verifier running in time sub-linear in the circuit size can not even read the circuit, which is part of the input of the universal circuit. For this reason, they restricted the circuits to be to log space uniform boolean cicuits

where $r \leftarrow \mathbb{Z}_p$ is the randomness of the commitment and the commitment key is $ck = ([\lambda_1(s)]_\gamma, \ldots, [\lambda_m(s)]_\gamma, [t(s)]_\gamma)$, for $s \leftarrow \mathbb{Z}_p$. The commitments are perfectly hiding and computationally binding under the discrete logarithm assumption. In some contexts we will not need the commitments to be hiding and we will take $r = 0$ and also eliminate $[t(s)]_\gamma$ from ck. Note that $\mathsf{Com}_{ck}(\boldsymbol{x}; r)$ is the polynomial $P(X) = \sum_{i=1}^m x_i\lambda_i(X) + rt(X)$ evaluated in $[s]_\gamma$, and we sometimes refer to it as $[P(s)]_\gamma$.

In Sect. 6 we use these commitments without randomness, that is $\mathsf{Com}_{ck}(\cdot, 0)$. Consequently, $P(X) = \sum_{i=1}^m x_i\lambda_i(X)$.

3 Security Definitions for SNARKs

Let $\mathcal{R}$ be a relation generator which on input 1^λ outputs a family of polynomial time decidable relations $R_\Phi = \{R_\phi\}$, $R_\phi = \{(\phi, u, w)\}$, with associated language $\mathcal{L}_\phi = \{(\phi, u) : \exists w \ (\phi, u, w) \in R_\phi\}$. For simplicity of notation, we assume that R_Φ implicitly or explicitly describes λ and R_ϕ implicitly or explicitly describes R_Φ.

For arithmetic circuits, $\mathcal{R}$ outputs $R_\Phi = \{R_\phi : \ \phi \in \Phi\}$ and a bilinear group gk, where

- for fixed values p, N, Φ is the set of all arithmetic circuits defined over $\mathbb{Z}_p$ and such that the total number of inputs plus multiplication gates is at most N,
- $R_\phi = \{(\phi, (\boldsymbol{x}_{pub}, \boldsymbol{x}_{sec}), \boldsymbol{y}) : \phi(\boldsymbol{x}_{pub}, \boldsymbol{x}_{sec}) = \boldsymbol{y}\}$, where we distinguish between public inputs $\boldsymbol{x}_{pub}$ and secret inputs $\boldsymbol{x}_{sec}$,
- gk is some bilinear group of order p.

The circuit ϕ computes a function $\mathbb{Z}_p^n \to \mathbb{Z}_p^{n'}$ for some $n, n' \in \mathbb{N}$, $n_{pub} + n_{sec} = n$. We define the language $\mathcal{L}_\phi = \{(\phi, \boldsymbol{x}_{pub}, \boldsymbol{y}) : \exists \boldsymbol{x}_{sec}, (\phi, (\boldsymbol{x}_{pub}, \boldsymbol{x}_{sec}), \boldsymbol{y}) \in R_\phi\}$. Without loss of generality, we assume that all outputs of ϕ are the output of some multiplication gate. The statement is $u = (\phi, \boldsymbol{x}_{pub}, \boldsymbol{y})$ and the witness is $w = (\boldsymbol{x}_{sec})$ or any information which can be efficiently computed from (u, w). For instance, we will often use as witness $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$ where $\boldsymbol{a}$ (resp. $\boldsymbol{b}$, resp. $\boldsymbol{c}$) is a valid assignment to all left (resp. right, resp. input and output of multiplication gates) wires.

As discussed in [23], relations in practice arise in many ways and there are different possibilities to generate them. Therefore we do not assume that the bilinear group (which determines p) is generated before the family of circuits is chosen or the other way around, but we leave this decision to the specification of the relation generator $\mathcal{R}$.

Below we give the following syntactical definition for our argument system. It is inspired by the definitions of Groth et al. [24], but with significant differences. The argument of Groth et al. generates a universal common reference string which is updatable. It allows to derive common reference strings for specific statements, but this is just for efficiency purposes (in the actual scheme, the global crs is quadratic, while the specific will be linear). That is, one could just use the global crs to create and verify proofs. In our case, there is a global, universal crs which is incomplete, and to prove statements about circuits one needs to derive a specific crs. Furthermore, both the global and the specific circuit dependent crs are updatable.

We note that our model of soundness assumes that the quadratic crs is updated before the linear crs is created and updated. In particular, we assume that a single query ($\mathsf{final}\ \Sigma$) and ($\mathsf{final}\ \sigma$) is done. We could extend this model to allow more queries of this form, but we need to make the restriction that for each updated quadratic crs, the linear crs is generated from scratch. For the scheme to be sound, there needs to be one honest update in the quadratic part and one in the linear part.

For simplicity of notation, we assume that the global crs is Σ_Φ and the specialized crs σ_ϕ include a description of the relations R_Φ, R_ϕ, respectively.

Definition 1. *An updatable non-interactive argument for a relation R_Φ is a tuple $P =$ (Setup, $\mathsf{Setup.Upd}$, $\mathsf{Setup.Vrfy}$, $\mathsf{Drv.Setup}$, $\mathsf{Drv.Upd}$, $\mathsf{Drv.Vrfy}$, Prove, Vrfy) such that:*

- $\mathsf{Setup}(R_\Phi)$ *is a probabilistic algorithm which produces a common reference string Σ_Φ together with a proof of correctness $\rho_{\Phi,0}$.*

– $\mathsf{Setup.Upd}(\Sigma_\Phi, \{\rho_{\Phi,i}\}_{i=0}^h)$ *is a probabilistic algorithm which on input a relation* R_Φ*, a common reference string* Σ_Φ *and proofs of correctness* $\{\rho_{\Phi,i}\}_{i=0}^h$ *derives an updated common reference string* Σ'_Φ *together with a new proof* $\rho_{\Phi,h+1}$ *and outputs* $(\Sigma'_\Phi, \{\rho_{\Phi,i}\}_{i=0}^{h+1})$.
– $\mathsf{Setup.Vrfy}(\Sigma_\Phi)$ *is a deterministic algorithm which on input* Σ_Φ *outputs a bit* $b \in \{0,1\}$.
– $\mathsf{Drv.Setup}(\Sigma_\Phi, R_\phi, \{\rho_{\Phi,i}\}_{i=0}^h)$ *is a probabilistic algorithm which on input a relation* $R_\phi \in R_\Phi$*, a common reference string* Σ_Φ *with a proof of correctness, outputs a common reference string* σ_ϕ *for* R_ϕ *together with a proof of correctness* $\rho_{\phi,0}$.
– $\mathsf{Drv.Upd}(\sigma_\phi, \{\rho_{\phi,i}\}_{i=0}^h)$ *is a probabilistic algorithm which on input a common reference string* σ_ϕ *and proofs of correctness* $\{\rho_{\phi,i}\}_{i=0}^h$ *derives an updated common reference string* σ'_ϕ *together with a new proof* $\rho_{\phi,h+1}$ *and outputs* $(\sigma'_\phi, \{\rho_{\phi,i}\}_{i=0}^{h+1})$.
– $\mathsf{Drv.Vrfy}(\sigma_\phi, \{\rho_{\phi,i}\}_{i=0}^h)$ *is a deterministic algorithm which on input a common reference string* σ_ϕ *and a sequence of proofs* $\{\rho_{\phi,i}\}_{i=0}^h$ *outputs* $b \in \{0,1\}$.
– $\mathsf{Prove}(\sigma_\phi, \phi, u, w)$ *is a probabilistic algorithm which on input* σ_ϕ, $(\phi, u, w) \in R_\phi$ *outputs a proof* π.
– $\mathsf{Vrfy}(\sigma_\phi, (\phi, u), \pi)$ *is a deterministic algorithm which on input* σ_ϕ, (ϕ, u) *outputs* $b \in \{0,1\}$.

We consider the following definitions for our argument. They are inspired by [24,36]. We define perfect completeness in the usual way. In Subversion Zero-Knowledge an adversary is allowed to request either real or simulated proofs for many specialized crs with associated arbitrary updates. Given an extractor which extracts the simulation trapdoor for all the queries, even an unbounded adversary should be unable to distinguish real from simulated proofs.

Definition 2. *A non-interactive argument for a relation* R *is*

– *Perfectly complete if for all PPT algorithms* $\mathcal{A}$ *the advantage* $|1 - \Pr[\mathsf{COMP}_\mathcal{A}(R_\Phi)]|$ *is negligible in* λ.
– P *is* S*-Zero-Knowledge (Subvertible Zero-Knowledge) if for all PPT algorithms* $\mathcal{A}$ *the probability* $|\Pr[\mathsf{S\text{-}ZK}_\mathcal{A}(R_\Phi)] - 1/2|$ *is negligible in* λ.
– P *is* U*-knowledge sound (updatable knowledge sound) if for all PPT algorithms* $\mathcal{A}$ *there exists a PPT extractor* $\chi_\mathcal{A}$ *such that the probability* $\Pr[\mathsf{U\text{-}KSND}_{\mathcal{A},\chi_\mathcal{A}}(R_\Phi)]$ *is negligible in* λ.

Definition 3. *An updatable argument is perfectly correct if the probability of each of the following events is 1.*

1. $\mathsf{Setup.Vrfy}(\Sigma_\Phi, \rho_{\Phi,0}) = 1$ *conditioned on* $(\Sigma_\Phi, \rho_{\Phi,0}) \leftarrow \mathsf{Setup}(R_\Phi)$.
2. $\mathsf{Setup.Vrfy}(\Sigma'_\Phi, \{\rho_{\Phi,i}\}_{i=0}^{h+1}) = 1$ *conditioned on* $(\Sigma'_\Phi, \{\rho_{\Phi,i}\}_{i=0}^{h+1}) \leftarrow \mathsf{Setup.Upd}(\Sigma_\Phi, \{\rho_{\Phi,i}\}_{i=0}^h)$ *and* $\mathsf{Setup.Vrfy}(\Sigma_\Phi, \{\rho_{\Phi,i}\}_{i=0}^h) = 1$
3. $\mathsf{Drv.Vrfy}(\sigma_\phi, \rho_{\phi,0}) = 1$ *conditioned on* $(\sigma_\phi, \rho_{\phi,0}) \leftarrow \mathsf{Drv.Setup}(\Sigma_\Phi, R_\phi, \{\rho_{\Phi,i}\}_{i=0}^h)$ *and* $\mathsf{Setup.Vrfy}(\Sigma_\Phi) = 1$.
4. $\mathsf{Drv.Vrfy}(\sigma'_\phi, \{\rho_{\phi,i}\}_{i=0}^{h+1}) = 1$ *conditioned on* $(\sigma'_\phi, \{\rho_{\phi,i}\}_{i=0}^{h+1}) \leftarrow \mathsf{Drv.Upd}(\sigma_\phi, \{\rho_{\phi,i}\}_{i=0}^h)$ *and* $\mathsf{Drv.Vrfy}(\sigma_\phi, \{\rho_{\phi,i}\}_{i=0}^h) = 1$

3.1 Cryptographic Assumptions

Definition 4. *Let* $k \in \mathbb{N}$. *We call* $\mathcal{D}_k$ *a matrix distribution if it outputs (in PPT time, with overwhelming probability) matrices in* $\mathbb{Z}_p^{k+1\times k}$.

The Kernel Diffie-Hellman Assumption [38] says one cannot find a non-zero vector in one of the groups which is in the co-kernel of $\mathbf{A}$. We use the split assumption (or $\mathcal{D}_k$-SKerMDH) which says one cannot find a pair of vectors in $\mathbb{G}_1^{k+1} \times \mathbb{G}_2^{k+1}$ such that the difference of the vector of their discrete logarithms is in the co-kernel of $\mathbf{A}$.

Assumption 1 (Split Kernel Diffie-Hellman Assumption [20]) *For all non-uniform PPT adversaries* $\mathcal{A}$*:*

$$\Pr\left[[\boldsymbol{r}]_1, [\boldsymbol{s}]_2 \leftarrow \mathcal{A}(gk, [\mathbf{A}]_{1,2}) : \boldsymbol{r} \neq \boldsymbol{s} \wedge \boldsymbol{x}^\top \mathbf{A} = \boldsymbol{s}^\top \mathbf{A}\right] = \mathsf{negl}(\lambda),$$

where the probability is taken over $gk \leftarrow \mathcal{G}(1^\lambda)$, $\mathbf{A} \leftarrow \mathcal{D}_k$ *and the coin tosses of adversary* $\mathcal{A}$.

```
MAIN $\mathsf{COMP}_{\mathcal{A}}\{R_\Phi\}$
    $(\sigma_\phi, (\rho_{\phi,i})_{i=1}^h, (\phi, u, w)) \leftarrow \mathcal{A}(R_\Phi)$
    $b \leftarrow \mathsf{Drv.Vrfy}(\sigma_\phi, (\rho_{\phi,i})_{i=1}^h)$
    If $b = 0$ or $(\phi, u, w) \notin R_\phi$ return 1
    $\pi \leftarrow \mathsf{Prove}(\sigma_\phi, (\phi, u, w))$
    Return $\mathsf{Verify}(\sigma_\phi, (\phi, u))$
```

```
MAIN $\mathsf{S\text{-}ZK}_{\mathcal{A},\chi_{\mathcal{A}}}(R_\Phi)$
    $b \leftarrow \{0, 1\}$
    $r \leftarrow \{0, 1\}^{\mathcal{A}.\mathsf{rl}(\lambda)}$
    $\mathsf{st} \leftarrow \mathcal{A}(R_\Phi; r)$
    $\{(\sigma_{\phi_j}, \tau_{\phi_j}) : R_{\phi_j} \in R_\Phi\}_{j=1}^q \leftarrow \chi_{\mathcal{A}}(R_\Phi; r)$
    $b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{pf}}}(\mathsf{st})$
    return $b' = b$
```

```
$\mathcal{O}_{\mathsf{pf}}(\sigma_{\phi_j}, \{\rho_{\phi_j,i}\}_{i=0}^h, u, w)$
    if $\mathsf{Drv.Vrfy}(\sigma_{\phi_j}, (\rho_{\phi_j,i})_{i=1}^h) = 0$
        return $\perp$
    if $b = 0$ return $\mathsf{SimProve}(\sigma_\phi, (\phi, u), \tau_{\phi_j})$
    else return $\mathsf{Prove}(\sigma_\phi, (\phi, u, w))$
```

```
MAIN $\mathsf{U\text{-}KSND}_{\mathcal{A},\chi_{\mathcal{A}}}(R_\Phi)$
    $Q_{\rho,\Phi}, Q_\phi, Q_{\rho,\phi} \leftarrow \perp;$
    $(\phi, u, \pi) \overset{r}{\leftarrow} \mathcal{A}^{\mathcal{O}_{\mathsf{upd}}}(R_\Phi)$
    $w \leftarrow \chi_{\mathcal{A}}(\sigma_\phi, r)$
    return $\big(\mathsf{Vrfy}(\sigma_\phi, u, \phi, \pi) \wedge$
        $(\phi, u, w) \notin R_\phi \wedge (\sigma_\phi, \phi) \in Q_\Phi\big)$
```

```
$\mathcal{O}_{\mathsf{upd}}(\mathsf{intent}, \phi, \sigma_{\phi,h}, \{\rho_{\phi,i}\}_{i=1}^h)$
    if $\mathsf{intent} = \mathsf{setup}\ \Sigma$
        if $Q_{\rho,\Phi} \neq \perp$ return $\perp$
        else $(\Sigma_{\Phi,0}, \rho_{\Phi,0}) \leftarrow \mathsf{Setup}(R_\Phi)$
            $Q_{\rho,\Phi} \leftarrow Q_{\rho,\Phi} \cup \{\rho_{\Phi,0}\}$

    if $\mathsf{intent} = \mathsf{update}\ \Sigma$
        if $\mathsf{Setup.Vrfy}(\Sigma_{\Phi,h}, \{\rho_{\Phi,i}\}_{i=0}^h) = 0$
        return $\perp$
        else $(\Sigma_{\Phi,h+1}, \{\rho_{\Phi,i}\}_{i=0}^{h+1})$
            $\leftarrow \mathsf{Setup.Upd}(\Sigma_\Phi, \{\rho_{\Phi,i}\}_{i=0}^h)$
            $Q_{\rho,\Phi} \leftarrow Q_{\rho,\Phi} \cup \{\rho_{\Phi,h+1}\}$

    if $\mathsf{intent} = \mathsf{final}\ \Sigma$
        if $\Big(\mathsf{Setup.Vrfy}(\Sigma_{\Phi,h}, \{\rho_{\Phi,i}\}_{i=0}^h) = 0$
            $\vee (\{\rho_{\Phi,i}\}_{i=0}^h \cap Q_{\rho,\Phi} = \emptyset)\Big)$ return $\perp$
        else $\Sigma_\Phi = \Sigma_{\Phi,h}$

    if $\mathsf{intent} = \mathsf{setup}\ \sigma$
        if $\phi \in Q_\phi \vee Q_{\rho,\Phi} = \emptyset$
            return $\perp$
        else
            $(\sigma_{\phi,0}, \rho_{\phi,0}) \leftarrow \mathsf{Drv.Setup}(\Sigma_\Phi, R_\phi, \{\rho_{\Phi,i}\}_{i=0}^h)$
            $Q_\phi \leftarrow Q_\phi \cup \{\phi\}$
            $Q_{\rho,\phi} \leftarrow Q_{\rho,\phi} \cup \{(\phi, \rho_{\phi,0})\}$

    if $\mathsf{intent} = \mathsf{update}\ \sigma$
        if $0 = \mathsf{Drv.Vrfy}(\phi, \sigma_{\phi,h}, \{\rho_{\phi,i}\}_{i=0}^h)$
            return $\perp$
        else
            $(\sigma_{\phi,h+1}, \{\rho_{\phi,i}\}_{i=0}^{h+1}) \leftarrow$
                $\mathsf{Drv.Upd}(\sigma_{\phi,h}, \{\rho_{\phi,i}\}_{i=0}^h)$
            $Q_{\rho,\phi} \leftarrow Q_{\rho,\phi} \cup \{(\phi, \rho_{\phi,h+1})\}$

    if $\mathsf{intent} = \mathsf{final}\ \sigma$
        if $\Big(\mathsf{Drv.Vrfy}(\phi, \sigma_{\phi,h}, \{\rho_{\phi,i}\}_{i=0}^h) = 0$
            $\vee\ \{(\phi, \rho_{\phi,i})\}_{i=0}^h \cap Q_{\rho,\phi}\Big) = \emptyset$ return $\perp$
        else $\sigma_\phi \leftarrow \sigma_{\phi,h}$
```

Fig. 1. Security Games.

This assumption is generically hard for all distributions for which $\mathcal{D}_k$-KerMDH_γ is hard, whenever $k > 2$.

For our construction based on falsifiable assumptions, we introduce an assumption which is similar to the q-SFrac Assumption considered in [18], but in the source group.

Assumption 2 ($(\mathcal{R}, \boldsymbol{q})$-RSDH Assumption) *Let $\mathcal{R}$ be an arbitrary set of integers. The $(\mathcal{R}, \boldsymbol{q}$-Rational Strong Diffie-Hellman Assumption holds in $\mathbb{G}_1$ if the following probability is negligible in λ:*

$$\Pr\left[\begin{array}{c} gk \leftarrow \mathcal{G}(1^\lambda); \\ ([z]_1, [w]_1) \leftarrow \mathcal{A}(gk, \left\{[s^i]_1\right\}_{i=1}^{q-1}, \left\{[s^i]_2\right\}_{i=1}^{q}); \end{array} \middle| \begin{array}{c} e([z]_1, [1]_2) = e([w]_1, [t(s)]_2) \\ z \neq 0 \end{array}\right],$$

where $t(s) = \prod_{r \in \mathcal{R}}(s - r)$, and the probability is taken over $gk \leftarrow \mathcal{G}(1^\lambda)$, $s \leftarrow \mathbb{Z}_p$ and the coin tosses of adversary $\mathcal{A}$.

It is important to note that it is possible to check if an adversary has succeeded in breaking the assumption, since given $\left\{[s^i]_1\right\}_{i=1}^{q-1}, \left\{[s^i]_2\right\}$ the value $[t(s)]_2$ can be constructed as a linear combination of $\{[s^i]_2\}_{i=1}^q$ given $\mathcal{R}$.

The intuition why the assumption is generically hard is as follows. Since $[z]_1, [w]_1$ are given in group $\mathbb{G}_1$, the adversary must construct them as a linear combinations of all elements it has received in $\mathbb{G}_1$, which are $([1]_1, [s]_1, \ldots, [s^{q-1}]_1)$. On the other hand, the adversary can only win if $z/t(s) = w$, but the adversary can only find a non-trivial solution generically if z is constructed as a (non-zero) multiple of $t(X) = \prod_{r \in \mathcal{R}}(X - r)$ evaluated at s. But this is not possible because in $\mathbb{G}_1$ it only receives powers of s of degree at most $q - 1$ and T is of degree q.

Finally, we will be using the following knowledge assumption.

Assumption 3 ($q(\lambda)$-PKE) *The $q(\lambda)$-power knowledge of exponent assumption holds for $gk \leftarrow \mathcal{G}(1^\lambda)$ if for all $\mathcal{A}$ there exists a non-uniform probabilistic polynomial-time extractor $\chi_\mathcal{A}$ such that:*

$$\Pr\left[\begin{array}{c} (([c]_1, [\hat{c}]_2); a_0, \ldots, a_q) \\ \leftarrow (\mathcal{A}||\chi_\mathcal{A})(gk, \{[s^i]_{1,2}\}_{i=0}^q) \end{array} \middle| \begin{array}{c} e([c]_1, [1]_2) = e([1]_1, [\hat{c}]_2); \\ c \neq \sum_{i=0}^q a_i s^i; \end{array}\right] = \mathsf{negl}(\lambda).$$

The following is a special case of the monomial computational assumption of [24], which is introduced for the proof of updatability.

Assumption 4 ($q(\lambda)$-MC,[24]) *The $q(\lambda)$-univariate monomial computational assumption holds for $gk \leftarrow \mathcal{G}(1^\lambda)$ for $\boldsymbol{a} = \{a_i(\boldsymbol{X})\}_{i=1}^{n_a}$ and $\boldsymbol{b} = \{b_i(\boldsymbol{X})\}_{i=1}^{n_b}$, two sets of n variate polynomials of degree at most m, with $m, n, n_a, n_b \leq q$ if for all PPT $\mathcal{A}$:*

$$\Pr\left[([A]_{1,2}, a(X)) \leftarrow \mathcal{A}(gk, \{[a_i(s)]_1\}_{i=1}^{n_a}, \{[b_j(s)]_2\}_{i=1}^{n_b}) \middle| \begin{array}{c} A = a(s); \\ a(X) \notin span(\{a_i(X)\}_{i=1}^{n_a}) \end{array}\right] = \mathsf{negl}(\lambda).$$

The following assumption is related to the monomial knowledge of exponent assumption of [24].

Assumption 5 ($q(\lambda)$-*BPKE*) *The $q(\lambda)$-bivariate power knowledge of exponent assumption holds for $gk \leftarrow \mathcal{G}(1^\lambda)$ if for all $\mathcal{A}$ there exists a non-uniform probabilistic polynomial-time extractor $\chi_\mathcal{A}$ such that:*

$$\Pr\left[\begin{array}{c} (([\boldsymbol{b}]_1, [\hat{\boldsymbol{b}}]_2, [\boldsymbol{c}]_2, [\hat{\boldsymbol{c}}]_2); \boldsymbol{a}_{b,0}, \ldots, \boldsymbol{a}_{b,q}, \boldsymbol{a}_{b,q+1}, \boldsymbol{a}_c) \\ \leftarrow (\mathcal{A}||\chi_\mathcal{A})(gk, \{[s^i]_{1,2}\}_{i=0}^q, [\omega]_{1,2}) \end{array} \middle| \begin{array}{c} e([\boldsymbol{b}]_1, [1]_2) = e([1]_1, [\hat{\boldsymbol{b}}]_2); \\ e([1]_1, [\boldsymbol{c}]_2) = e([\omega]_1, [\hat{\boldsymbol{c}}]_2); \\ \boldsymbol{b} \neq \sum_{i=0}^q \boldsymbol{a}_{b,i} s^i + \boldsymbol{a}_{b,q+1}\omega; \\ \boldsymbol{c} \neq \boldsymbol{a}_c \omega; \end{array}\right] = \mathsf{negl}(\lambda).$$

4 Arithmetic Circuits

Arithmetic circuits are acyclic directed graphs where the edges are called wires and the vertices are called gates. Gates with in-degree 0 are labeled by variables X_i, $i = 1, \ldots, n$ or with a constant field element, the rest of the gates are either labeled with $\times$ and are referred to as multiplication gates or with $+$ and are called addition gates. In this work we consider only fan-in 2 multiplication gates and the circuit is defined over a field $\mathbb{Z}_p$, where p is the order of some cryptographically useful bilinear group. Each circuit computes a function $\phi : \mathbb{Z}_p^n \to \mathbb{Z}_p^{n'}$.

Let $\mathcal{G}$ be the set of multiplicative gates of the circuit excluding multiplication-by-constant gates. We denote by m the cardinal of this set. For simplicity and without loss of generality, we may assume all outputs of the circuit to be the output of some multiplication gate.

We encode circuit satisfiability as a set of equations in $n + 3m$ variables, where variables are assigned to the left, right and output wires of each multiplication gate at level i, $i > 1$ and to the input wires. This encoding is standard and follows closely [7].

Lemma 1. *Let $\phi : \mathbb{Z}_p^n \to \mathbb{Z}_p^{n'}$ be a circuit with m multiplicative gates (excluding multiplication by constant gates). For $i = n+1, \ldots, n+m$, $j = 1, \ldots, n+m$, there exist*

a) variables A_i, B_i, C_j and
b) constants $f_i, g_i, f_{i,j}, g_{i,j} \in \mathbb{Z}_p$

such that, for every $(x_1, \ldots, x_n) \in \mathbb{Z}_p^n$, if we set $C_k = x_k$, for all $k = 1, \ldots, n$, then $C(x_1, \ldots, x_n) = (y_1, \ldots, y_{n'})$ and A_i, B_i, C_i are evaluated respectively to the left, the right and the output wires of the ith gate, if and only if the following equations are satisfied:

1. *(Quadratic constraints)* $C_i = A_i B_i$,
2. *(Affine Constraints)* $A_i = f_i + \sum_{j=1}^{n+m} f_{i,j} C_j$ *and* $B_i = g_i + \sum_{j=1}^{n+m} g_{i,j} C_j$.
3. *(Correct Output)* $C_{n+m-n'+k} = y_k$, *for all* $k = 1, \ldots, n'$.

Given an arithmetic circuit $\phi : \mathbb{Z}_p^n \to \mathbb{Z}_p^{n'}$, we define the witness for correct evaluation of $C(\boldsymbol{x}) = \boldsymbol{y}$ as a tuple $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}) \in \mathbb{Z}_p^m \times \mathbb{Z}_p^m \times \mathbb{Z}_p^{n+m}$, which is an assignment to A_i, B_i and C_j that satisfies the equations described in Lemma 1.

What is different of our encoding from writing circuit satisfiability as a "Quadratic Arithmetic Program" or QAP [16] is that we separate the set of linear and quadratic constraints more clearly by adding auxiliary variables which represent right and left wires. In particular, what is particularly interesting of this encoding is that the set of quadratic constraints does not depend on the circuit but just on the number of multiplication gates. By forcing some values to be 0, this will allow us to define a common reference string to prove correct evaluation of $\hat{m}$ multiplication gates, for any $\hat{m} < m$. To prove the quadratic constraints in a succint matter, we resort to QAPs, which we view as a technique for aggregation of quadratic equations.

Lemma 2. *(QAP for the Hadamard Product) Let $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}) \in (\mathbb{Z}_p^m)^3$, $m \in \mathbb{N}$. Let $\mathcal{R} = \{r_1, \ldots, r_m\} \subset \mathbb{Z}_p$ be a set of elements of $\mathbb{Z}_p$ and let $\lambda_i(X) = \prod_{j \neq i} \dfrac{X - r_j}{r_i - r_j}$. Define*

$$p(X) = \left(\sum_{i=1}^{m} a_i \lambda_i(X) \right) \left(\sum_{i=1}^{m} b_i \lambda_i(X) \right) - \left(\sum_{i=1}^{m} c_i \lambda_i(X) \right).$$

Then, $\boldsymbol{c} = \boldsymbol{a} \circ \boldsymbol{b}$ if and only if $p(X) = h(X)t(X)$, where $t(X) = \prod_{r \in \mathcal{R}} (X - r)$ and $h(X) \in \mathbb{Z}_p[X]$ is a polynomial of degree at most $m - 2$.

Proof. By definition, $p(r_i) = a_i b_i - c_i$, so $p(X)$ is divisible by $t(X)$ if and only if $a_i b_i - c_i = 0$ for all $i = 1, \ldots, m$.

Lemma 3. *(Polynomial characterization of affine constraints) Let* $\boldsymbol{a} = (a_{n+1}, \ldots, a_{n+m})$*,* $\boldsymbol{b} = (b_{n+1}, \ldots, b_{n+m})$ *and* $\boldsymbol{c} = (c_1, \ldots, c_{n+m})$ *be three sets of vectors of* $\mathbb{Z}_p$*. Then,*

$$a_i = f_i + \sum_{j=1}^{n+m} f_{i,j} c_j \text{ and } b_i = g_i + \sum_{j=1}^{n+m} g_{i,j} c_j,$$

for some constants $f_i, g_i, f_{i,j}, g_{i,j} \in \mathbb{Z}_p$ *if and only if there exist two sets of polynomials*

$$\mathcal{V} = \{v_i(X)\}_{i=0,\ldots,n+m}, \mathcal{W} = \{w_i(X)\}_{i=0,\ldots,n+m}$$

of degree at most $m-1$*, such that*

$$\sum_{i=n+1}^{i=n+m} a_i \lambda_i(X) = v_0(X) + \sum_{i=1}^{n+m} c_i v_i(X) \text{ and } \sum_{i=n+1}^{i=n+m} b_i \lambda_i(X) = w_0(X) + \sum_{i=1}^{n+m} c_i w_i(X).$$

The proof is obvious and follows from evaluating at the points of $\mathcal{R}$.

4.1 Circuit Slicing into Different Levels

In Sect. 6, we will need to partition the set $\mathcal{G}$ of multiplicative gates of the circuit into different levels. More precisely, we define $\{\mathcal{G}_i\}_{i=1}^{d'}$, where $\mathcal{G}_i$, for $i = 1, \ldots, d'$, is the set of gates $G \in \mathcal{G}$ such that the maximum of gates in $\mathcal{G}$ evaluated in any path from the input of the circuit to an input of G is $i-1$. The minimal such d' for which the partition exists is the multiplicative depth of the circuit, which we always denote by d. Further, we define $\mathcal{G}_0$ to be the set of n_0 variable inputs. If $G \in \mathcal{G}_i$, we say that G has multiplicative depth i. Let n_i be the cardinal of $\mathcal{G}_i$. With this notation, a circuit computes a function $\phi : \mathbb{Z}_p^{n_0} \to \mathbb{Z}_p^{n_d}$, i.e. $n = n_0$, $n' = n_d$ and the number of multiplication gates is $m = \sum_{i=1}^d n_i$ (recall that we may assume all outputs to be the output of some multiplication gate.)

We now consider an encoding circuit satisfiability where the variables are divided according to their multiplicative depth. For each gate in $\mathcal{G}_i$, $i \in \{1, \ldots, d\}$ the circuit is correctly evaluated if the output of the gate is the product of two multivariate polynomials of degree 1 where the variables are outputs of gates of less multiplicative depth, that is the output of gates in $\mathcal{G}_j$, for some j, $0 \geq j \geq i-1$. For clarity, we formalize this in the following lemma.

Lemma 4. *Let* $\phi : \mathbb{Z}_p^{n_0} \to \mathbb{Z}_p^{n_d}$*, be a circuit of multiplicative depth* d *and with* m *gates. For* $i \in \{1, \ldots, d\}$*, define* n_i *as the number multiplication gates at level* i*. There exist*

a) variables $C_{i,j}$*,* $i = 0, \ldots, d$*,* $j = 1, \ldots, n_i$*,*
b) variables $A_{i,j}$*,* $B_{i,j}$*,* $i = 1, \ldots, d$*,* $j = 1, \ldots, n_i$*,*
b) constants $f_{i,j}, g_{i,j}, f_{i,j,k,\ell}, g_{i,j,k,\ell} \in \mathbb{Z}_p$*,* $i = 1, \ldots, d$*,* $k = 0, \ldots, i-1$*,* $j = 1, \ldots, n_i$*,* $\ell = 1, \ldots, n_k$

such that, for every $(x_1, \ldots, x_{n_0}) \in \mathbb{Z}_p^{n_0}$*, if we set* $C_{0,j} = x_j$*, for all* $j = 1, \ldots, n_0$*, then* $C(x_1, \ldots, x_{n_0}) = (y_1, \ldots, y_{n_d})$ *and for each* $i \in \{1, \ldots, d\}$*,* $A_{i,j}, B_{i,j}, C_{i,j}$ *are evaluated respectively to the left, the right and the output wires of the* j*th gate at level* i*, if and only if the following equations are satisfied:*

1. *(Quadratic Constraints). For each* $i = 1, \ldots, d$*, if* $j = 1, \ldots, n_i$*:* $C_{i,j} = A_{i,j} B_{i,j}$*.*
2. *(Affine Constraints)* $A_{i,j} = f_{i,j} + \sum_{k=0}^{i-1} \sum_{\ell=1}^{n_k} f_{i,j,k,\ell} C_{k,\ell}$ *and* $B_{i,j} = g_{i,j} + \sum_{k=0}^{i-1} \sum_{\ell=1}^{n_k} g_{i,j,k,\ell} C_{k,\ell}$*.*
3. *(Correct Output)* $C_{d,j} = y_j$*,* $j = 1, \ldots, n_d$*.*

Given an arithmetic circuit $\phi : \mathbb{Z}_p^{n_0} \to \mathbb{Z}_p^{n_d}$, we can define the witness for correct evaluation of $\phi(\boldsymbol{x}) = \boldsymbol{y}$ as a tuple $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$, where $\boldsymbol{a} = (\boldsymbol{a}_1, \ldots, \boldsymbol{a}_d)$, $\boldsymbol{b} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_d)$, $\boldsymbol{c} = (\boldsymbol{c}_0, \ldots, \boldsymbol{c}_d)$, $\boldsymbol{s}_i = (s_{i1}, \ldots, s_{in_i})$ for any $s \in \{a, b, c\}$. The tuple should be an an assignment to $A_{i,j}, B_{i,j}$ and $C_{i,j}$ which satisfies the equations described in Lemma 4.

Our approach will be to apply Lemma 2 and Lemma 3 separately to each of the circuits levels. That is, for any fixed i we will aggregate all quadratic equations $C_{ij} = A_{ij}B_{ij}$ into a divisibility relation, while the linear constraints will translate into two polynomial identities.

We observe that affine constraints can be written in terms of families of polynomials $\mathcal{V} = \{v_{ik\ell}\}$, $\mathcal{W} = \{w_{ik\ell}\}$ such that $\sum_{j=1}^{n_i} a_{ij}\lambda_j(X) = v_{i0}(X) + \sum_{k=0}^{i-1}\sum_{\ell=1}^{n_k} c_{kl}v_{ik\ell}(X)$ and $\sum_{j=1}^{n_i} b_{ij}\lambda_j(X) = w_{i0}(X) + \sum_{k=0}^{i-1}\sum_{\ell=1}^{n_k} c_{kl}w_{ik\ell}(X)$.

5 A New NIZK Argument in the Generic Group Model

This section describes a new SNARK for proving arithmetic circuit satisfiability. The resulting proof size is $(5, 1)$ elements. For a comparison, state-of-the art in asymmetric bilinear groups (Groth [23]) is $(2, 1)$, and Pinocchio [40,6] is $(7, 1)$. This seems like a very acceptable tradeoff, as our scheme has several nice features: it is fully updatable, it can be specialized to any circuit (of a maximum size), has a linear crs and the cost of specializing the crs for every circuit or verifying its correctness is linear in the circuit size.

We omit some improvements which are described in Sect. 5.3. Among them, we think it is particularly interesting to observe that in fact the structured part of the crs needs to be only proportional to the number of multiplication gates plus secret inputs, and not to the total number of circuit wires.

Recall from Sect. 3 that R_Φ is a family of relations which specifies some maximal bound N of inputs plus multiplication gates and some bilinear group gk of order p.

The simulation trapdoor for our scheme are two values $s, \boldsymbol{k}$, the first one is the secret associated to the crs for proving the quadratic, universal constraints and $\boldsymbol{k}$ the secret associated to proving the affine and circuit dependent constraints. The final secret after a sequence of h_Φ updates for the quadratic part (resp. h updates for the linear part) is simply the multiplication (resp. the addition) of all the updates. That is, each update for the quadratic part contributes to the final trapdoor $\xi_i \in \mathbb{Z}_p^*$ and the final secret is $s_{h_\Phi} := \prod_{i=0}^{h_\Phi} \xi_i$. Each update for the linear part contributes some $\boldsymbol{\gamma}_i \in \mathbb{Z}_p^4$ and $\boldsymbol{k}_h = \sum_{i=0}^{h} \boldsymbol{\gamma}_i$. Multiplicative updates can be handled as in [24]. However, for the linear update, where the updates are additive, we add to the proof of correctness another element ψ which is updated multiplicatively in order to ensure that the contribution to the final value $\boldsymbol{k}_h$ can be extracted from any correct update. The party running the update algorithm must compute certain values involving the product of k_h and ψ_h, which generically it can only do when it has knowledge of its contribution to γ_h. The element ψ is updated multiplicatively and for this element it holds that $\psi_h := \prod_{i=0}^{h} \alpha_i$ for some α_i which is the update trapdoor.

Whenever it is clear from the context that h is the index of the latest update, we simply write $s, \boldsymbol{k}, \psi$ for $s_h, \boldsymbol{k}_h, \psi_h$, respectively.

$\mathsf{Setup}(R_\Phi)$**:** This algorithm samples $s \leftarrow \mathbb{Z}_p^*$ and publishes

$$\Sigma_\Phi = \left(gk, \{[\lambda_i(s)]_{1,2}\}_{i=1}^N, \{[s^i]_{1,2}\}_{i=1}^N, [t(s)]_{1,2}\right),$$

where $\lambda_i(X)$ are the Lagrangian polynomials associated to some set $\mathcal{R} = \{r_1, \ldots, r_N\} \subset \mathbb{Z}_p$. The simulation trapdoor is $\tau_\Sigma = s$. It also outputs a proof of correctness $\rho_{\Phi,0} := \{[\xi_0]_{1,2} = [s]_{1,2}, [s_0]_2 = [s]_2\}$ with associated update trapdoor $\tau_{upd,\Phi} = \xi_0$.

$\mathsf{Setup.Upd}(\Sigma_{\Phi,h}, \{\rho_{\Phi,i}\}_{i=0}^h)$**:** On input $\Sigma_{\Phi,h} = \left(gk, \{[\lambda_i(s_h)]_{1,2}\}_{i=1}^N, \{[s_h^i]_{1,2}\}_{i=1}^N, [t(s_h)]_{1,2}\right)$ and $\{\rho_{\Phi,i}\}_{i=0}^h = \{[\xi_i]_1, [s_i]_2\}_{i=1}^h$, pick $\xi_{h+1} \leftarrow \mathbb{Z}_p^*$ and define $s_{h+1} := \xi_{h+1}s_h$, $\rho_{\Phi,h+1} := ([\xi_{h+1}]_{1,2}, [s_{h+1}]_2)$. Define:

$$\Sigma_{\Phi,h+1} = \left(gk, \{[\lambda_i(s_{h+1})]_{1,2}\}_{i=1}^N, \{[s_{h+1}^i]_{1,2}\}_{i=1}^N, [t(s_{h+1})]_{1,2}\right),$$

which can be computed by writing all its terms as polynomials in s_h (whose coefficients depend on $\mathcal{R}$ and ξ_{h+1}) and by using the powers of s_h included in $\Sigma_{\Phi,h}$. Output $(\Sigma_{\Phi,h+1}, \{\rho_{\Phi,i}\}_{i=0}^{h+1})$ and the update trapdoor is $\tau_{upd,\Phi} = \xi_{h+1}$.

$\mathsf{Setup.Vrfy}(\Sigma_{\Phi,h}, \{\rho_{\Phi,i}\}_{i=0}^h)$**:** On input $\Sigma_{\Phi,h} = \left(gk, \{[\lambda_{i,1}]_1, [\lambda_{i,2}]_2\}_{i=1}^N, \{[s_{h,i,1}]_1, [s_{h,i,2}]_2\}_{i=1}^N, [t_1]_1, [t_2]_2\right)$ and $\{\rho_{\Phi,i}\}_{i=0}^h = \{[\xi_i]_{1,2}, [s_i]_2\}$, do the following checks:

1. $e([\xi_i]_1, [s_{i-1}]_2) = e([1]_1, [s_i]_2)$ and $e([\xi_i]_1, [1]_1) = e([1]_1, [\xi_i]_2)$ for all $i = 1, \dots, h$,
2. $e([s_{h,i-1,1}]_1, [s_h]_2) = e([s_{h,i,1}]_1, [1]_2) = e([1]_1, [s_{h,i,2}]_2)$, for all $i = 1, \dots, N$ and defining $s_{h,0,1} = 1$.
3. If $\hat{t}_i$ are the coefficients of $t(X)$, check a) $[t_1]_1 = \sum_{i=0}^{N} \hat{t}_i [s_{h,i,1}]_1$, b) $[t_2]_2 = \sum_{i=0}^{N} \hat{t}_i [s_{h,i,2}]_2$,
4. If $\hat{\lambda}_{i,j}$ are the coefficients of $\lambda_i(X)$, check a) $[\lambda_{i,1}]_1 = \sum_{j=0}^{N-1} \hat{\lambda}_{i,j} [s_{h,j,1}]_1$ and b) $[\lambda_{i,2}]_2 = \sum_{j=0}^{N-1} \hat{\lambda}_{i,j} [s_{h,j,2}]_2$.

$\mathsf{Drv.Setup}(\phi, n_{pub}, \Sigma_\Phi, \{\rho_{\Phi,i}\}_{i=0}^{h_\Phi})$**:** On input a circuit $\phi : \mathbb{Z}_p^n \to \mathbb{Z}_p^{n'}$ with m multiplication gates, $n_{pub} \leq n$ public inputs and such that $n+m \leq N$ and Σ_Φ, this algorithm computes $([\mathbf{Q}]_1, [\boldsymbol{q}_0]_1) \in \mathbb{G}_1^{4\times n+m+3} \times \mathbb{G}_1^4$, defined as:

$$\mathbf{Q} = \begin{pmatrix} \mathbf{\Lambda}_{pub} & \mathbf{0} & \mathbf{\Lambda}_{opt} & 0 & 0 & 0 \\ \mathbf{0} & \mathbf{\Lambda}_{mid} & \mathbf{\Lambda}_{opt} & t(s) & 0 & 0 \\ \mathbf{V}_{pub} & \mathbf{V}_{mid} & \mathbf{0} & 0 & t(s) & 0 \\ \mathbf{W}_{pub} & \mathbf{W}_{mid} & \mathbf{0} & 0 & 0 & t(s) \end{pmatrix}, \qquad \boldsymbol{q}_0 = \begin{pmatrix} 0 \\ 0 \\ v_0(s) \\ w_0(s) \end{pmatrix}, \tag{2}$$

where $\mathbf{\Lambda}_{pub} = (\lambda_1(s), \dots, \lambda_{n_{pub}}(s))$, $\mathbf{\Lambda}_{mid} = (0, \dots, 0, \lambda_{n+1}(s), \dots, \lambda_{n+m-n'}(s))$, $\mathbf{\Lambda}_{opt} = (\lambda_{n+m-n'+1}(s), \dots, \lambda_{n+m}(s))$, $\mathbf{V}_{pub} = (v_1(s), \dots, v_{n_{pub}}(s))$, $\mathbf{V}_{mid} = (v_{n_{pub}+1}(s), \dots, v_{n+m-n'}(s))$, $\mathbf{W}_{pub} = (w_1(s), \dots, w_{n_{pub}}(s))$, $\mathbf{W}_{mid} = (w_{n_{pub}+1}(s), \dots, w_{n+m-n'}(s))$, and the sets of polynomials $\mathcal{V} = \{v_i(X)\}_{i=0}^{n+m}$, $\mathcal{W} = \{w_i(X)\}_{i=0}^{n+m}$ are the ones associated to the circuit as specified in Lemma 1 and Lemma 3.
It samples $\boldsymbol{k} \leftarrow \mathbb{Z}_p^4$ and publishes $\sigma_{\phi,\mathsf{lin}} = ([\mathbf{Q}^\top \boldsymbol{k}]_1, [\boldsymbol{k}]_2)$. It outputs the final common reference string:

$$\sigma_\phi = (\Sigma_\Phi, [\mathbf{Q}]_1, [\boldsymbol{q}_0]_1, \phi, n_{pub}, \sigma_{\phi,\mathsf{lin}}).$$

For the proof of correctness, it samples $\psi_0 \leftarrow \mathbb{Z}_p^*$ and outputs $\rho_{\phi,0} = (\{\rho_{\Phi,i}\}_{i=0}^{h_\Phi}, [\boldsymbol{q}_1^\top \boldsymbol{\gamma}_0]_1, [\boldsymbol{\gamma}_0]_2, [\psi_{-1}\boldsymbol{\gamma}_0]_2, [\psi_0]_{1,2}, [\alpha_0]_1)$, where $\boldsymbol{q}_1$ is the first column of $\mathbf{Q}$, $\boldsymbol{\gamma}_0 = \boldsymbol{k}$, $\psi_{-1} = 1$, $\alpha_0 = \psi_0$. The simulation trapdoor is $\tau_\phi = (s, \boldsymbol{k})$ and the update trapdoor is $\tau_{upd,\phi} = (\boldsymbol{\gamma}_0, \alpha_0)$.

$\mathsf{Drv.Upd}(\phi, \sigma_\phi, \{\rho_{\phi,i}\}_{i=0}^{h})$**:** On input $\sigma_\phi = (\Sigma_\Phi, [\mathbf{Q}]_1, [\boldsymbol{q}_0]_1, \phi, n_{pub}, \sigma_{\phi,\mathsf{lin}})$, and $\{\rho_{\phi,i}\}_{i=0}^{h} = (\{\rho_{\Phi,i}\}_{i=0}^{h_\Phi}, \{[\boldsymbol{q}_1^\top \boldsymbol{\gamma}_i]_1, [\boldsymbol{\gamma}_i]_2, [\psi_{i-1}\boldsymbol{\gamma}_i]_2, [\psi_i]_{1,2}, [\alpha_i]_1\}_{i=0}^{h})$, for some $h \in \mathbb{N}$, this algorithm samples $\boldsymbol{\gamma}_{h+1} \leftarrow \mathbb{Z}_p^4, \alpha_{h+1} \leftarrow \mathbb{Z}_p^*$, sets $[\boldsymbol{k}_{h+1}]_2 = [\boldsymbol{\gamma}_{h+1}]_2 + [\boldsymbol{k}_h]_2$ and $[\psi_{h+1}]_1 = \alpha_{h+1}[\psi_h]_1$, and updates the common reference string as:

$$\sigma_{\phi,lin} \leftarrow ([\mathbf{Q}^\top \boldsymbol{\gamma}_{h+1}]_1 + [\mathbf{Q}^\top \boldsymbol{k}_h]_1, [\boldsymbol{k}_{h+1}])$$

and the proof of correctness as: $\left\{ \{\rho_{\Phi,i}\}_{i=0}^{h_\Phi}, \{[\boldsymbol{q}_1^\top \boldsymbol{\gamma}_i]_1, [\boldsymbol{\gamma}_i]_2, [\psi_{i-1}\boldsymbol{\gamma}_i]_2, [\psi_i]_{1,2}, [\alpha_i]_1\}_{i=0}^{h+1} \right\}$. The update trapdoor is $\tau_{upd,\phi} = (\boldsymbol{\gamma}_i, \alpha_i)$.

$\mathsf{Drv.Vrfy}(\phi, \sigma_\phi, \{\rho_{\phi,i}\}_{i=0}^{h})$**:** This algorithm parses the input as:

$$\sigma_\phi = (\Sigma_\phi, [\mathbf{Q}]_1, [\boldsymbol{q}_0]_1, \phi, n_{pub}, [\mathbf{Q}_{k,h}]_1, [\boldsymbol{k}_h]_2),$$

and $\{\rho_{\phi,i}\}_{i=0}^{h} = (\{\rho_{\Phi,i}\}_{i=0}^{h_\Phi}, \{[q_{\gamma,i}]_1, [\boldsymbol{\gamma}_i]_2, [\boldsymbol{\gamma}_{\psi,i}]_2, [\psi_{i,1}]_1, [\psi_{i,2}]_2, [\alpha_i]_1\}_{i=0}^{h})$, and does the following checks:
1. $\mathsf{Setup.Vrfy}(\Sigma_\Phi, \{\rho_{\Phi,i}\}_{i=0}^{h_\Phi}) = 1$.
2. $\sum_{i=0}^{h} [\boldsymbol{\gamma}_i]_2 = [\boldsymbol{k}_h]_2$.
3. $e([\boldsymbol{q}_{\gamma,i}]_1, [1]_2) = e([\boldsymbol{q}_1^\top]_1, [\boldsymbol{\gamma}_i]_2)$, for $i = 0, \dots, h-1$.
4. $e([\psi_{i,1}]_1, [1]_2) = e([1]_1, [\psi_{i,1}]_2) = e([\alpha_i]_1, [\psi_{i-1,2}]_2)$, for $i = 0, \dots, h$.
5. $e([\psi_{i-1,1}]_1, [\boldsymbol{\gamma}_i]_2) = e([1]_1, [\boldsymbol{\gamma}_{\psi,i}]_2)$ for $i = 1, \dots, h$.
6. $e([\mathbf{Q}_{k,h}]_1, [1]_2) = e([\mathbf{Q}^\top]_1, [\boldsymbol{k}_h]_2)$.

$\mathsf{Prove}(\phi, \sigma_\phi, (\boldsymbol{x}_{pub}, \boldsymbol{x}_{sec}, \boldsymbol{y}))$**:** From $(\boldsymbol{x}_{pub}, \boldsymbol{x}_{sec}, \boldsymbol{y})$ the prover generates a (redundant) satisfiability witness $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$ which satisfies the constraints of Lemma 1.
1. The prover samples $\delta_1, \delta_2, \delta_3 \leftarrow \mathbb{Z}_p$ and commits to the vector of output wires $\boldsymbol{c}$ starting from $n+1$ in $\mathbb{G}_1$, to the vector of left wires $\boldsymbol{a}$ in $\mathbb{G}_1$ and to the vector of right wires $\boldsymbol{b}$ in both groups $\mathbb{G}_1, \mathbb{G}_2$ as:

$$[O]_1 = \sum_{i=n+1}^{n+m} c_i [\lambda_i(s)]_1 + \delta_1 [t(s)]_1 \qquad [L]_1 = \sum_{i=n+1}^{n+m} a_i [\lambda_i(s)]_1 + \delta_2 [t(s)]_1$$

$$[R]_{1,2} = \sum_{i=n+1}^{n+m} b_i [\lambda_i(s)]_{1,2} + \delta_3 [t(s)]_{1,2}.$$

2. It defines

$$p(X) = \left(\sum_{i=n+1}^{n+m} a_i\lambda_i(X) + \delta_2 t(X)\right)\left(\sum_{i=n+1}^{n+m} b_i\lambda_i(X) + \delta_3 t(X)\right) - \left(\sum_{i=n+1}^{n+m} c_i\lambda_i(X) + \delta_1 t(X)\right).$$

.
   If $h(X)$ is such that $p(X) = h(X)t(X)$, compute $[H]_1 = [h(s)]_1$ with the powers $\{[s^i]_1\}_{i=0}^{N}$ given in Σ_Φ.
3. It defines $[O_{pub}]_1 = \sum_{i=1}^{n_{pub}} c_i[\lambda_i(s)]_1 + \sum_{i=n+m-n'+1}^{n+m} c_i[\lambda_i(s)]_1$ and sets $\boldsymbol{z}^\top = \big(O_{pub}, O, L, R\big) - \boldsymbol{q}_0^\top$, and it computes a proof that $[\boldsymbol{z}]_1$ is in the column span of $\mathbf{Q}$ as $[\pi]_1 = (\boldsymbol{c}^\top, \boldsymbol{\delta}^\top)[\mathbf{Q}^\top\boldsymbol{k}]_1$. The output of the algorithm is

$$\Pi = ([L]_1, [R]_1, [R]_2, [O]_1, [H]_1, [\pi]_1).$$

$\mathsf{Vrfy}((\phi, \boldsymbol{x}_{pub}, \boldsymbol{y}), \sigma_\phi, \Pi)$**:** On input the proof $\Pi = ([L]_1, [R]_1, [\hat{R}]_2, [O]_1, [H]_1, [\pi]_1)$ for some instance $(\phi, \boldsymbol{x}_{pub}, \boldsymbol{y})$, this algorithm outputs 1 if the following checks are successful and 0 otherwise:
1. $e([L]_1, [R]_2) - e([O]_1, [1]_2) = e([H]_1, [t(s)]_2)$.
2. Compute $[O_{pub}]_1 = \sum_{i=1}^{n_{pub}} c_i[\lambda_i(s)]_1 + \sum_{i=n+m-n'+1}^{n+m} c_i[\lambda_i(s)]_1$, $[\boldsymbol{z}^\top]_1 = [O_{pub}, O, L, R]_1 - [\boldsymbol{q}_0^\top]_1$ and verify if $e([\pi]_1, [1]_2) = e([\boldsymbol{z}^\top]_1, [\boldsymbol{k}]_2)$.
3. $e([R]_1, [1]_2) = e([1]_1, [\hat{R}]_2)$.

5.1 Proof of Updatability

The following lemmas are analogous to [24, Lemma 4, Lemma 5, Lemma 6], the only difference being that our single update adversary (Lemma 6) makes a single honest setup query to its oracle and possibly many non-honest updates, while [24, Lemma 5] makes a single honest setup and a single non-honest update. This is more consistent with the security model we have described.

Lemma 5 says that from any adversary producing a valid quadratic and linear crs "from scratch" it is possible to extract the corresponding simulation trapdoors, as well as the update trapdoor. This will be used in the proof of subvertible zero-knowledge and it will also be used for arguing that it is enough to prove soundness against single adversarial updates.

Lemma 5 (Trapdoor extraction for subvertible CRSs). . *For any PPT adversary $\mathcal{A}$ that outputs $(\Sigma_\Phi, \{\rho_{\Phi,i}\}_{i=0}^{h_\Phi})$ and $(\sigma_\phi, \{\rho_{\phi,i}\}_{i=1}^{h})$, under the BDH-KE Assumption and the KWKE Assumption of [2], there exists an extractor $\chi_\mathcal{A}$ which, on input R_Φ and the random coins of $\mathcal{A}$, outputs τ_Φ, τ_ϕ such that $(\Sigma_\Phi, \rho_{\Phi,0}) =$ $\mathsf{Setup}(R_\Phi; \tau_\Phi)$ and $(\sigma_\phi, \rho_{\phi,0}) = \mathsf{Drv.Setup}(\phi, n_{pub}, \Sigma_\Phi, \{\rho_{\Phi,i}\}_{i=0}^{m}; \tau_\phi)$.*

Proof. For trapdoor τ_Φ, note that the adversary outputs $[s]_1, [s']_2$ as part of Σ_Φ. Since such crs is valid, it must be that $e([s]_1, [1]_1) = e([1]_1, [s']_2)$ and, by the BDH-KE assumption, there is an extractor which on input $\mathcal{A}$'s random coins outputs $\tau_\Phi = s = s'$.

For the trapdoor τ_ϕ, as noted by Abdolmaleki et al. ([2, Thm. 1]), if a generic adversary produces $[\boldsymbol{k}]_2$ and $[\mathbf{Q}_{\boldsymbol{k}}]_1$ such that it is accepted by $\mathsf{Drv.Vrfy}$, that is, such that $e([\mathbf{Q}_{\boldsymbol{k}}]_1, [1]_2) = e([\mathbf{Q}^\top]_1, [\boldsymbol{k}]_2)$, one can efficiently extract such $\boldsymbol{k}$ (this is what they call the KWKE assumption). A full proof can be found in [2, Thm. 1], although our proof is actually simpler, as in our case $\boldsymbol{k}$ is unique (because $\mathbf{Q}$ has more columns than rows and $\overline{\mathbf{A}} = 1$).

Further, the adversary outputs $[\psi_{h,1}]_1, [\psi_{h,2}]_2$ such that $e([\psi_{h,1}]_1, [1]_2) = e([1]_1, [\psi_{h,2}]_2)$ from which we can extract $\psi_h = \psi_{h,1} = \psi_{h,2}$ by the BDH-KE Assumption. The rest of the checks of the crs imply that $\psi_h = \prod_{i=0}^{h} \alpha_i$.

Lemma 6 says that the update trapdoor can also be extracted when the adversary (maliciously) updates an honestly generated crs.

Lemma 6 (Trapdoor extraction for honest setup and single adversarial update). *Consider a PPT adversary $\mathcal{A}$ against* U-KSND *that calls its oracle $\mathcal{O}_{\mathsf{upd}}$ with the following sequence:* $\mathsf{setup}\ \Sigma$, $(\mathsf{final}\ \Sigma, \Sigma_{\Phi,1}, \{\rho_{\Phi,i}\}_{i=0,1})$ *and then* $(\mathsf{setup}\ \sigma, \phi_1), \ldots, (\mathsf{setup}\ \sigma, \phi_{q_c}), (\mathsf{final}\ \sigma, \phi_j, \sigma_{\phi_j,1}, \{\rho_{\phi_j,i}\}_{i=0,1})$*, where* $j \in \{1, \ldots, q_c\}$*,* $(\Sigma_{\Phi,0}, \rho_{\Phi,0})$ *and* $(\sigma_{\phi_j}, \rho_{\phi_j,0})$ *are the output of the oracle to the first and the* $(2+j)$*th call to* $\mathcal{O}_{\mathsf{upd}}$*.*

If $(\sigma_{\phi_j,1}, \{\rho_{\phi_j,i}\}_{i=0,1})$ *is accepted by the* Drv.Vrfy *algorithm, under N-BPKE Assumption and the $(N+2)$-MC Assumption, for any such adversary there exists an extractor $\chi_{\mathcal{A}}$ which, on input $\mathcal{A}$'s random coins, outputs update trapdoors* $\tau_{upd,\Phi} = (\xi_1) \in \mathbb{Z}_p, \tau_{upd,\phi_j} = (\boldsymbol{\gamma}_1, \alpha_1) \in \mathbb{Z}_p^4 \times \mathbb{Z}_p$ *such that* $(\Sigma_{\Phi,1}, \{\rho_{\Phi,i}\}_{i=0,1}) = \mathsf{Setup.Upd}((\Sigma_{\Phi,0}, \rho_{\Phi,0}); \xi_1)$ *and* $(\sigma_{\phi_j,1}, \{\rho_{\phi_j,i}\}_{i=0,1}) = \mathsf{Drv.Setup}((\phi_j, \sigma_{\phi_j,0}, \rho_{\phi_j,0}); (\boldsymbol{\gamma}_1, \alpha_1))$*.*

Proof. We first show that under the N-$BPKE$ Assumption and the $(N+2)$-MC Assumption, we can extract the contribution of adversary $\mathcal{A}$ to the quadratic part and the linear part.

Let $\left(\{[s^i]_{1,2}\}_{i=0}^N, [\omega]_{1,2}\right)$ be a challenge for the N-$BPKE$ assumption. From this challenge, it is possible to compute $(\Sigma_{\Phi,0}, \rho_{\Phi,0})$ as an answer to the first oracle query and give it as input to $\mathcal{A}$. Adversary $\mathcal{A}$ outputs as part of its query $(\mathsf{final}\ \Sigma, \Sigma_{\Phi,1}, \{\rho_{\Phi,i}\}_{i=0,1})$ the elements $[\xi_{1,1}]_1$ and $[\xi_{1,2}]_2$, and from the fact that $(\Sigma_{\Phi,1}, \{\rho_{\Phi,i}\}_{i=0,1})$ is accepted by the setup verification algorithm, it follows that

$$e([\xi_{1,1}]_1, [1]_2) = e([1]_1, [\xi_{1,2}]_2), \tag{3}$$

which in particular implies that we can define $\xi_1 = \xi_{1,1} = \xi_{1,2}$.

When $\mathcal{A}$ makes the query $(\mathsf{setup}\ \sigma, \phi_i)$, the challenger sets $\psi_{0,i} = \omega\hat{\psi}_{0,i}$, for some $\hat{\psi}_{0,i} \leftarrow \mathbb{Z}_p^*$ and samples $\boldsymbol{k}_i \leftarrow \mathbb{Z}_p^4$ to generate the rest of the parameters for the linear crs for circuit ϕ_i. Finally, $\mathcal{A}$ outputs $(\mathsf{final}\ \sigma, \phi_j, \sigma_{\phi_j,1}, \{\rho_{\phi_j,i}\}_{i=0,1})$ for some $j \in \{1, \ldots, q_c\}$. Since this crs is accepted by the Drv.Vrfy algorithm, it follows the adversary's output includes among other things, values $[\psi_{1,1}]_1, [\psi_{1,2}]_2$ such that

$$e([\psi_{1,1}]_1, [1]_2) = e([1]_1, [\psi_{1,2}]_2), \tag{4}$$

so $\psi_1 = \psi_{1,1} = \psi_{1,2}$ is correctly defined. Further, it holds that $e([\psi_1]_1, [1]_2) = e([\alpha_1]_1, [\psi_{0,j}]_2)$, therefore:

$$\psi_1 = \alpha_1 \psi_{0,j} = \alpha_1 \hat{\psi}_{0,j}\omega. \tag{5}$$

It also holds that $e([\psi_{0,j}]_1, [\boldsymbol{\gamma}_1]_2) = e([1]_1, [\boldsymbol{\gamma}_{\psi,1}]_1)$, which implies:

$$e([\omega]_1, [\boldsymbol{\gamma}_1]_2) = e([1]_1, [\boldsymbol{\gamma}_{\psi,1}/\hat{\psi}_{0,j}]_2). \tag{6}$$

From equations (3), (4), (6) it hods that ψ_1, ξ_1 are polynomials in the span of $\{1, X, \ldots, X^N, Y\}$ evaluated at (s, ω), $\boldsymbol{\gamma}_1$ is a degree 1 polynomial evaluated at ω in $\mathbb{Z}_p[Y]$ which can be extracted under the N-$BPKE$ assumption.

Further, equation (5), together with the Schwartz-Zippel Lemma implies that ψ_1 is the result of evaluating at ω the polynomial $\alpha_1\hat{\psi}_{0,j}Y$, i.e. this polynomial is in fact independent of X. In particular, from the extracted coefficients it is possible to recover $\boldsymbol{\gamma}_1, \alpha_1\hat{\psi}_{0,j}$ and recover $\boldsymbol{\gamma}_1, \alpha_1$.

On the other hand, for the quadratic part, the extracted coefficients are $\zeta_0, \ldots \zeta_N, \zeta_{N+1}$, such that if $\zeta(X, Y) = \sum_{j=0}^N \zeta_j X^j + \zeta_{N+1}Y$, $\xi_{1,1} = \xi_{1,2} = \xi_1 = \sum_{j=0}^N \zeta_j s^j + \zeta_{N+1}\omega = \zeta(s, \omega)$. Let j^* be the largest index in the range $\{1, \ldots, N\}$ such that $\zeta_{j^*} \neq 0$. Since $s_1 = \xi_1 s$ it follows that $s_1^N = p(s, \omega)$, where $p(X, Y) = X^N(\zeta(X, Y))^N$ is a polynomial of degree $N(1 + j^*)$ in X, degree at least N in Y. From the latter, we conclude that if $\zeta_{N+1} \neq 0$ or $j^* \geq 1$ it holds that $p(X, Y) \notin \mathrm{Span}(\{X^i\}_{i=0}^N, Y)$, which would break the $(N+2)$-MC Assumption. Therefore, the polynomial p does not depend on Y and $j^* = 0$. We conclude that we can extract $\xi_1 = \zeta_0 \in \mathbb{Z}_p$ such that $(\Sigma_{\Phi,1}, \{\rho_{\Phi,0}, \rho_{\Phi,1}\}) = \mathsf{Setup.Upd}(\Sigma_{\Phi,0}, \{\rho_{\Phi,0}, \rho_{\Phi,1}\}; \xi_1)$.

It is also important to note that in Lemma 6, to extract the contribution of the adversary to the final trapdoor it suffices to use the proof of correctness and not the common reference string. In particular, if an adversary does an honest setup for the quadratic part, and then h_Φ updates to the quadratic part, an honest update for the linear part and then h updates for the linear part, we can extract its (aggregated) contribution to the global crs by applying several times Lemma 6. In other words, for extraction we do not need to have all the intermediate common reference strings but the contribution of the adversary can be extracted from the proofs of correctness.

Similarly as in [24], updates can be combined, that is $\mathsf{Setup.Upd}(\mathsf{Setup.Upd}(\mathsf{Setup}(R_\Phi;\xi_0);\xi_1);\xi_2) = \mathsf{Setup}(R_\Phi;\xi_0\xi_1\xi_2)$ and $\mathsf{Drv.Upd}(\phi,(\mathsf{Drv.Upd}(\phi,(\mathsf{Drv.Setup}(\phi,n_{pub},\Sigma_\Phi,\{\rho_i\}_{i=1}^{h_\Phi};\boldsymbol{\gamma}_0,\alpha_0);\boldsymbol{\gamma}_1,\alpha_1);\boldsymbol{\gamma}_2,\alpha_2) = \mathsf{Drv.Setup}(\phi, n_{pub},\Sigma_\Phi,\{\rho_{\Phi,i}\}_{i=1}^{h_\Phi};\boldsymbol{\gamma}_0+\boldsymbol{\gamma}_1+\boldsymbol{\gamma}_2;\alpha_0\alpha_1\alpha_2)$.

As noted by Groth et al. in [24], this enables the following simulations:

– Given the trapdoor s_0 of $\Sigma_{\Phi,0}$ and the elements $[s_1]_1,[s_1]_2$ of $\Sigma_{\Phi,1}$, we can simulate the update proof $\rho_{\Phi,1} = ([s_1]_{1,2}/s_0,[s_1]_2)$ of $\Sigma_{\Phi,1}$ being an update of $\Sigma_{\Phi,0}$. We denote this simulated update proof by $\rho^{\mathsf{sim}}_{\Sigma_{\Phi,1}\leftarrow\Sigma_{\Phi,0}}(\Sigma_{\Phi,1},\Sigma_{\Phi,0},\tau_{\Phi,0})$.
– Given the update trapdoor ξ_1 for $\Sigma_{\Phi,1}$ being an update of $\Sigma_{\Phi,0}$, and the update proof $\rho_\Phi = ([\xi]_{1,2},[s]_2)$ for Σ_Φ being an update of $\Sigma_{\Phi,0}$, we can simulate the update proof $\rho'_\Phi = ([\xi]_{1,2}/\xi_1,[s]_2)$ for Σ_Φ being an update of $\Sigma_{\Phi,1}$. We denote this simulated update proof by $\rho^{\mathsf{sim}}_{\Sigma_\Phi\leftarrow\Sigma_{\Phi,1}}(\Sigma_{\Phi,1},\Sigma_\Phi,\tau_{upd,\Sigma_{\Phi,1}\leftarrow\Sigma_{\Phi,0}},\rho_{\Sigma_\Phi\leftarrow\Sigma_{\Phi,0}})$.
– Given $\boldsymbol{k}_0,\psi_0$ for $\sigma_{\phi,0}$ and the elements $[\mathbf{Q}]_1,[\mathbf{Q}^\top\boldsymbol{k}_1]_1,[\boldsymbol{k}_1]_2$ of the crs and $[\psi_1]_{1,2}$ of the proof $\rho_{\phi,1}$ of correct setup of $\sigma_{\phi,1}$, we can simulate the update proof $\rho'_{\phi,1} = ([\boldsymbol{q}_1^\top\boldsymbol{k}_1]_1-[\boldsymbol{q}_1^\top\boldsymbol{k}_0]_1,[\boldsymbol{k}_1]_2-[\boldsymbol{k}_0]_2,\psi_0([\boldsymbol{k}_1]_2-[\boldsymbol{k}_0]_2),[\psi_1]_{1,2},[\psi_1]_1/\psi_0)$ of $\sigma_{\phi,1}$ being an update of $\sigma_{\phi,0}$. We denote this simulated update proof by $\rho^{\mathsf{sim}}_{\sigma_{\phi,1}\leftarrow\sigma_{\phi,0}}(\sigma_{\phi,1},\sigma_{\phi,0},\rho_{\phi,1},(\boldsymbol{k}_0,\psi_0))$.
– Given the update trapdoor $(\boldsymbol{\gamma}_1,\alpha_1)$ for $\sigma_{\phi,1}$ being an update of $\sigma_{\phi,0}$, and the update proof $\rho_\phi = ([\boldsymbol{q}_1^\top\boldsymbol{\gamma}]_1,[\boldsymbol{\gamma}]_2,[\psi]_{1,2},[\alpha]_1)$ for σ_ϕ being an update of $\sigma_{\phi,0}$, we can simulate the update proof $\rho_\phi = ([\boldsymbol{q}_1^\top\boldsymbol{\gamma}]-[\boldsymbol{q}_1^\top\boldsymbol{\gamma}_1]_1,[\boldsymbol{\gamma}]_2-[\boldsymbol{\gamma}_1]_2,[\psi]_{1,2}/\alpha,[\alpha_1]_1/\alpha)$ for σ_ϕ being an update of $\sigma_{\phi,1}$. We denote this simulated update proof of σ_ϕ being an update of $\sigma_{\phi,1}$ by $\rho^{\mathsf{sim}}_{\sigma_\phi\leftarrow\sigma_{\phi,1}}(\sigma_\phi,\sigma_{\phi,1},\rho_\phi,(\boldsymbol{\gamma}_1,\alpha_1))$.

Putting all these ingredients together, we next prove Lemma 7, which says that one can consider, without loss of generality, adversaries against $\mathsf{U\text{-}KSND}$ of the type considered in Lemma 6, that is, adversaries who make an honest setup and a single adversarial update.

The following lemma is just an adaptation of [24] to our two-stage crs generation.

Lemma 7 (Single adversarial updates imply full updatable knowledge soundness). *For any adversary $\mathcal{A}$ against* $\mathsf{U\text{-}KSND}$ *there exists another "single update" adversary $\mathcal{B}$, as the one considered in Lemma 6, such that* $|\Pr[\mathsf{U\text{-}KSND}\mathcal{A},\chi_\mathcal{A}(R_\Phi)] - \Pr[\mathsf{U\text{-}KSND}_{\mathcal{B},\chi_\mathcal{B}}(R_\Phi)]|$ *is negligible in* λ.

Proof. We split $\mathcal{A}$ into three stages $\mathcal{A}_1,\mathcal{A}_2,\mathcal{A}_3$. The first stage ends with a successful query with $\mathsf{intent} = \mathsf{final}\ \Sigma$, and the second stage ends with a successful query with $\mathsf{intent} = \mathsf{final}\ \sigma$.

We construct an adversary $\mathcal{B}$ which first queries its oracle $\mathcal{O}_{\mathsf{upd}}$ on $\mathsf{setup}\ \Sigma$, and receives $(\Sigma_{\Phi,0},\rho_{\Phi,0})$ as answer. It initializes an empty database $D_{\rho,\Phi}$ of updates and corresponding randomness. It runs $\mathcal{A}_1(R_\Phi;r)$, for $r\leftarrow\{0,1\}^{\mathcal{A}_1.\mathsf{rl}(\lambda)}$, and answers its queries to the $\mathcal{O}_{\mathsf{upd}}$ oracle as follows:

$\mathsf{setup}\ \Sigma$: Add $(\rho_{\Phi,0},1)$ to $D_{\rho,\Phi}$ and return $(\Sigma_{\Phi,0},\rho_{\Phi,0})$

$\mathsf{update}\ \Sigma,(\Sigma_{\Phi,h},\{\rho_{\Phi,i}\}_{i=0}^h)$: If $\{\rho_{\Phi,i}\}_{i=0}^h$ does not contain any honest update, use the extractor of Lemma 5 to extract s_h such that $(\Sigma_{\Phi,h},\rho'_{\Phi,h}) = \mathsf{Setup}(R_\Phi;s_h)$, for some $\rho'_{\Phi,h}$. If $\{\rho_{\Phi,i}\}_{i=0}^h$ does contain honest updates, $\mathcal{A}_1$ can be perfectly simulated by an adversary $\mathcal{C}$ which on input $(\Sigma_{\Phi,0},\rho_{\Phi,0})$ runs $\mathcal{A}_1$ and answers oracle queries itself using the corresponding random coins from $D_{\rho,\Phi}$. Then $\mathcal{B}$ can extract ξ such that $(\Sigma_{\Phi,h},\{\rho_{\Phi,0},\rho''_{\Phi,h}\}) = \mathsf{Setup.Upd}((\Sigma_{\Phi,0},\rho_{\Phi,0});\xi)$ using the extractor from Lemma 6 repeatedly.

Adversary $\mathcal{B}$ chooses a valid update of $\Sigma_{\Phi,0}$, $(\hat{\Sigma}_\Phi,\{\rho_{\Phi,0},\hat{\rho}_\Phi\}) \overset{\hat{\xi}}{\leftarrow} \mathsf{Setup.Upd}(\Sigma_{\Phi,0},\rho_{\Phi,0})$ and explains it as an update of $\Sigma_{\Phi,h}$ by defining $\rho_{\Phi,h+1} = \rho^{\mathsf{sim}}_{\hat{\Sigma}_\Phi\leftarrow\Sigma_{\Phi,h}}(\hat{\Sigma}_\Phi,\Sigma_{\Phi,h},s_h)$ or $\rho_{\Phi,h+1} = \rho^{\mathsf{sim}}_{\hat{\Sigma}_\Phi\leftarrow\Sigma_{\Phi,h}}(\hat{\Sigma}_\Phi,\Sigma_{\Phi,h},\xi,\hat{\rho}_\Phi)$, depending on whether the query includes or not an honest update, and defines $\Sigma_{\Phi,h+1} = \hat{\Sigma}_\Phi$.

Finally, $\mathcal{B}$ adds the entry $(\hat{\rho}_\Phi,\hat{\xi})$ to $D_{\rho,\Phi}$ and returns $(\Sigma_{\Phi,h+1},\{\rho_{\Phi,i}\}_{i=0}^{h+1})$.

$\mathsf{final}\ \Sigma,(\Sigma_{\Phi,h_\Phi},\{\rho_{\Phi,i}\}_{i=0}^{h_\Phi})$: This query is honestly answered, that is, reject if the crs does not verify or the updates do not include any honest update or setup.

Adversary $\mathcal{B}$ uses the extractor from Lemma 6 repeatedly to construct proof ρ'''_{Φ,h_Φ} such that $(\Sigma_{\Phi,h_\Phi},\{\rho_{\Phi,0},\rho'''_{\Phi,h_\Phi}\}) = \mathsf{Setup.Upd}((\Sigma_{\Phi,0},\rho_{\Phi,0});\xi)$ for some ξ. $\mathcal{B}$ sends $(\mathsf{final}\ \Sigma,\Sigma_{\Phi,h_\Phi},\{\rho_{\Phi,0},\rho'''_{\Phi,h_\Phi}\})$ to its oracle $\mathcal{O}_{\mathsf{upd}}$ and sets

$\Sigma_\Phi = \Sigma_{\Phi,h_\Phi}$ and $S_\Phi = \{\rho_{\Phi,0}, \rho'''_{\Phi,h_\Phi}\}$. Then, it initializes an empty database $D_{\rho,\phi}$ of circuits, updates and corresponding randomness. It runs $\mathcal{A}_2(R_\phi; r_2)$, for $r_2 \leftarrow \{0,1\}^{\mathcal{A}_2.\mathsf{rl}(\lambda)}$, and answers its queries to the $\mathcal{O}_{\mathsf{upd}}$ oracle as follows:

$\mathsf{setup}\ \sigma, \phi$**:** If $(\phi, *, *)$ is already in $D_{\rho,\phi}$ it returns $\perp$. Otherwise, $\mathcal{B}$ forwards the query to its own oracle $\mathcal{O}_{\mathsf{upd}}$ obtaining $(\sigma_{\phi,0}, \rho_{\phi,0})$ as answer. It adds $(\phi, \rho_{\phi,0}, 1)$ to $D_{\rho,\phi}$ and returns $(\sigma_{\phi,0}, \rho_{\phi,0})$.

$\mathsf{update}\ \sigma, (\sigma_{\phi,h}, \{\rho_{\phi,i}\}_{i=0}^{h})$**:** If $\{\rho_{\phi,i}\}_{i=0}^{h}$ does not contain any honest update, use the extractor of Lemma 5 to extract $(\boldsymbol{k}_h, \psi_h)$ such that $(\sigma_{\phi,h}, \rho'_{\phi,h}) = \mathsf{Drv.Setup}((\phi, n_{pub}, \Sigma_\Phi, S_\Phi); (\boldsymbol{k}_h, \psi_h))$, for some $\rho'_{\phi,h}$. If $\{\rho_{\phi,i}\}_{i=0}^{h}$ does contain honest updates, $\mathcal{A}$ can be perfectly simulated by an adversary $\mathcal{C}$ which on input $(\sigma_{\phi,0}, \rho_{\phi,0})$ runs $\mathcal{A}$ and answers oracle queries itself using the corresponding random coins from $D_{\rho,\phi}$. Then $\mathcal{B}$ can extract $(\boldsymbol{\gamma}, \alpha)$ and construct $\rho''_{\phi,h}$ using the extractor from Lemma 6 such that $(\sigma_{\phi,h}, \{\rho_{\phi,0}, \rho''_{\phi,h}\}) = \mathsf{Drv.Upd}((\phi, \sigma_{\phi,0}, \rho_{\phi,0}); (\boldsymbol{\gamma}, \alpha))$.

$\mathcal{B}$ picks an update of $\sigma_{\phi,0}$, $(\hat{\sigma}_\phi, \{\rho_{\phi,0}, \hat{\rho_\phi}\}) \overset{(\hat{\gamma},\hat{\alpha})}{\leftarrow} \mathsf{Drv.Upd}(\sigma_{\phi,0}, \rho_{\phi,0})$ and explains it as an update of $\sigma_{\phi,h}$ computing $\rho_{\phi,h+1} = \rho^{\mathsf{sim}}_{\hat{\sigma}_\phi \leftarrow \sigma_{\phi,h}}(\hat{\sigma}_\phi, \sigma_{\phi,h}, \hat{\rho_\phi}, (\boldsymbol{k}_h, \psi_h))$ or $\rho_{\phi,h+1} = \rho^{\mathsf{sim}}_{\hat{\sigma}_\phi \leftarrow \sigma_{\phi,h}}(\hat{\sigma}_\phi, \sigma_{\phi,h}, \hat{\rho}_\phi, (\boldsymbol{\gamma}, \alpha))$, depending of whether or not the query includes an honest update, and defines $\sigma_{\phi,h+1} = \hat{\sigma}_\phi$.
Finally, $\mathcal{B}$ adds the entry $(\phi, \hat{\rho_\phi}, (\hat{\gamma}, \hat{\alpha}))$ to $D_{\rho,\phi}$ and returns $(\sigma_{\phi,h+1}, \{\rho_{\phi,i}\}_{i=0}^{h+1})$.

$\mathsf{final}\ \sigma, (\sigma_{\phi,h}, \{\rho_{\phi,i}\}_{i=0}^{h})$**:** This query is honestly answered, that is, reject if the crs doesn't verify or the updates do not include any honest update or setup.

By the same argument as for update queries, $\mathcal{B}$ can extract $(\boldsymbol{\gamma}, \alpha)$ and construct $\rho''_{\phi,h}$ using the extractor from Lemma 6 such that $(\sigma_{\phi,h}, \{\rho_{\phi,0}, \rho''_{\phi,h}\}) = \mathsf{Drv.Upd}((\phi, \sigma_{\phi,0}, \rho_{\phi,0}); (\boldsymbol{\gamma}, \alpha))$. Adversary $\mathcal{B}$ sends $(\mathsf{final}\ \sigma, (\sigma_{\phi,h}, \{\rho_{\phi,0}, \rho''_{\phi,h}\})$ to its oracle $\mathcal{O}_{\mathsf{upd}}$ and sets $\sigma_\phi = \sigma_{\phi,h}$. Finally, $\mathcal{B}$ runs $\mathcal{A}_3$ and outputs whatever $\mathcal{A}_3$ outputs.

Simulation is perfect and an extractor for $\mathcal{A}$ using the extractor of $\mathcal{B}$ as described in [24].

5.2 Security Proof

Lemma 8. *The argument is perfectly complete.*

Proof. If $a_i b_i = c_i$, then $p(X)$ is divisible by $t(X)$. This is because $p(r_i) = a_i b_i - c_i$ for $i = n+1, \ldots, n+m$ and $p(r_i) = 0$ for $i = 1, \ldots, n$, because $\lambda_j(r_i) = 0$ if $i \neq j$. Therefore, there exists a polynomial $h(X)$ of degree at most N such that $p(X) = h(X)t(X)$ and $p(s) = h(s)t(s)$, i.e. the first verification equation is verified. On the other hand, the second verification equation holds because by construction, $[\pi]_1 = [\boldsymbol{z}^\top \boldsymbol{k}]$ (i.e. the perfect completeness of the QANIZK argument of membership in linear spaces). Finally, the last equation holds because in an honestly constructed proof, $[R]_1$ and $[\hat{R}]_2$ have the same discrete logarithm.

Before we prove soundness, we observe that without loss of generality, if $\mathbf{Q}(X)$ is the matrix of polynomials such that $\mathbf{Q} = \mathbf{Q}(s)$, for $s \leftarrow \mathbb{Z}_p^*$, we can assume that the columns of $\mathbf{Q}(X)$, $\{q_i(X)\}_{i=1}^{n+m}$ are a set of linearly independent polynomials. To see this, first observe that $\{q_i(X)\}_{i=n+1}^{n+m}$ is a set of linearly independent polynomials and independent of the rest of the columns. This is because the second row of $\mathbf{Q}(X)$ consists of n zeros and the polynomials $(\lambda_{n+1}(X), \ldots, \lambda_{n+m}(X))$, which are linearly independent polynomials (to see this, observe that when evaluated at r_i, $i = n+1, \ldots, n+m$, only $\lambda_i(X)$ is non-zero). On the other hand, the last two rows of $\boldsymbol{q}_i(X)$ for $i = 1, \ldots, n$ are $\left\{ \begin{pmatrix} v_i(X) \\ w_i(X) \end{pmatrix} \right\}_{i=1}^{n}$. So, to argue that the columns of $\mathbf{Q}(X)$ can be assumed to be linearly independent polynomials, it suffices to see that without loss of generality, these polynomials are linearly independent.

Indeed, if they are not linearly independent, there exists another circuit $\hat{\phi} : \mathbb{Z}_p^{\hat{n}} \to \mathbb{Z}_p^{n'}$ with $\hat{n} < n$ such that if $\phi(\boldsymbol{x}) = \boldsymbol{y}$, then from $\boldsymbol{x}$ it is possible to efficiently compute $\hat{\boldsymbol{x}}$ such that $\hat{\phi}(\hat{\boldsymbol{x}}) = \boldsymbol{y}$ and such that the associated matrix $\hat{\mathbf{Q}}$ has linearly independent columns. More specifically, the circuit $\hat{\phi}$ can be built by successively applying the following rule: for any linear relation

$$\boldsymbol{q}_j(X) = \sum_{i=1, i\neq j}^{n} \ell_i \boldsymbol{q}_i(X),$$

define $\hat{n} = n-1$, eliminate the jth column from $\mathbf{Q}$ and replace the input $(x_1, \dots, x_n)$ by $(x_1+\ell_1 x_j, \dots, x_{j-1}+\ell_{j-1}x_j, x_{j+1}+\ell_1 x_j, \dots, x_n+\ell_n x_j)$.

Lemma 9. *The argument has updatable computational knowledge soundness in the generic bilinear asymmetric group model.*

Proof. The checks of $\mathsf{Drv.Vrfy}$ imply that the crs has the right structure, that is, $\sigma_{lin} = ([\mathbf{Q}^\top \boldsymbol{k}]_1, [\boldsymbol{k}]_2)$ for some $\boldsymbol{k} \in \mathbb{Z}_p^4$ and the same holds for Σ_Φ. Further, as we have argued in Lemma 7, it suffices to prove security against an adversary who makes an honest setup and a single adversarial updates. In particular, in this case, the associated trapdoors to σ_{lin}, Σ_Φ are $s = s_0 s_1$, $k = k_0 + k_1$, for s_1, k_1 chosen by the soundness adversary and $s_0 \leftarrow \mathbb{Z}_p$, $\boldsymbol{k}_0 \leftarrow \mathbb{Z}_p^4$.

We first show that a generic adversary that outputs a valid $[\boldsymbol{\pi}]_1$ must know a witness which satisfies the linear constraints. That is, we show implicitly that for this particular distribution of $\mathbf{Q}$, the QANIZK argument has knowledge soundness, similarly as it was proven in [13] for another subspace distribution. Then we show that, if the extracted witness does not satisfy the set of quadratic constraints contradicts the fact that the proof is accepted by the verifier.

From our previous discussion, it follows that the crs is the result of evaluating polynomials, $\boldsymbol{k}(\boldsymbol{X}) = (X_{K,1}+k_{1,1}, X_{K,2}+k_{1,2}, X_{K,3}+k_{1,3}, X_{K,4}+k_{1,4})$, $\boldsymbol{s}^j(X) = X_S^j s_1^j$ and $\{\boldsymbol{q}_j(\boldsymbol{X})\}_{j=1}^{m+n}$, $\{\boldsymbol{q}_j(\boldsymbol{X})^\top \boldsymbol{k}(\boldsymbol{X})\}_{j=1}^{m+n}$ at a random point $\boldsymbol{x}_0 = (k_{0,1}, \dots, k_{0,4}, s_0)$ (plus some other polynomials to define the Lagrangian polynomials and the target polynomial t evaluated at s).

Since the proof produced by the adversary satisfies the verification equation, it follows that $\pi(\boldsymbol{x}) = \boldsymbol{z}(\boldsymbol{x})^\top \boldsymbol{k}(\boldsymbol{x})$. By the Schwartz-Zippel lemma, the last equation implies that the correspondent polynomial equation holds with overwhelming probability, which we write

$$\pi(\boldsymbol{X}) = \sum_{i=1}^{4} \boldsymbol{z}_i(\boldsymbol{X})(X_{K,i}+k_{1,i}) \tag{7}$$

The only way to generically construct such a proof is to compute linear combinations of $\{\boldsymbol{q}_j(\boldsymbol{X})^\top \boldsymbol{k}(\boldsymbol{X})\}_{j=1}^{m+n}$ (which are the only terms with variables $X_{K,i}$ in the same group of definition as π). Therefore, the proof must satisfy

$$\pi(\boldsymbol{X}) = \sum_{j=1}^{m+n} w_j \sum_{i=1}^{4} \boldsymbol{q}_j(\boldsymbol{X})^\top \boldsymbol{k}(\boldsymbol{X}) = \sum_{i=1}^{4} \left(\sum_{j=1}^{n} \boldsymbol{q}_j(\boldsymbol{X}) w_j \right) (X_{K,i}+k_{1,i}).$$

As we have seen, we can assume without loss of generality that $\{\boldsymbol{q}_j(\boldsymbol{X})^\top \boldsymbol{k}(\boldsymbol{X})\}_{j=1}^{m+n}$ is a set of linearly independent polynomials. Then because of last equation, we can conclude that we can extract $\boldsymbol{w}$ such that $\boldsymbol{z} = \mathbf{Q}\boldsymbol{w} = \left(\sum_{j=1}^{n} \boldsymbol{q}_j w_j\right)$.

Now we show that, if the extracted witness does not satisfy the set of quadratic constraints, the first verification equation is satisfied with only negligible probability, following a standard argument. If $\boldsymbol{w}$ is the witness extracted, define $(\boldsymbol{c}^\top, \boldsymbol{\delta}^\top) = \boldsymbol{w}^\top$. It follows from the soundness property of this argument that $O = \sum_{i=n+1}^{n+m} c_i \lambda_i(s) + \delta_1 t(s)$, $L = v_0(s) + \sum_{i=n+1}^{n+m} c_i v_i(s) + \delta_2 t(s)$, and $R = w_0(s) + \sum_{i=n+1}^{n+m} c_i w_i(s) + \delta_3 t(s)$.

On the other hand from $\boldsymbol{c}$, the reduction can compute $\hat{L}(X_S) = v_0(s_1 X_S) + \sum_{i=n+1}^{n+m} c_i v_i(s_1 X_S)$, $\hat{R}(X_S) = w_0(s_1 X_S) + \sum_{i=n+1}^{n+m} c_i w_i(s_1 X_S)$ and $\hat{O}(X_S) = \sum_{i=n+1}^{n+m} c_i \lambda_i(s_1 X_S)$, such that $[\hat{L}(s)]_1 = [L]_1$, $[\hat{R}(s)]_1 = [R]_1$, $[\hat{O}(s)]_1 = [O]_1$. If the adversary breaks soundness, there exists some j such that $(\hat{L}\hat{R}-\hat{O})(r_j) = \mu \neq 0$, which implies that $\hat{L}(X_S)\hat{R}(X_S)-\hat{O}(X_S)$ is not divisible by $(X_S - r_j)$ and in particular, by $\hat{t}(X_S) = \prod_{r \in \mathcal{R}}(s_1 X_S - r)$.

Therefore, the first verification equation, $e([L]_1, [R]_2) - e([O]_1, [1]_2) = e([H]_1, [t(s)]_2)$, can only hold with negligible probability.

Lemma 10. *The argument has perfect subvertible zero-knowledge in the generic asymmetric bilinear group model.*

Proof. To prove this lemma, we need to show both the existence of an extractor which outputs the simulation trapdoor from any adversary which outputs valid parameters and an efficient simulator which outputs proofs with the same distribution as real proofs. The existence of such an extractor is a direct consequence of Lemma 5.

On the other hand, a proof with the same distribution as the honest setting can be computed by sampling $O, L, R \leftarrow \mathbb{Z}_p$ and setting: $[H]_1 = [(LR - O)/t(s)]_1$, which can be efficiently computed given s, and $[\pi]_1 = k_1[O_{pub}]_1 + k_2[O]_1 + k_3[L]_1 + k_4[R]_1$. Both real and simulated O, L, R are uniformly distributed over $\mathbb{Z}_p$ while H is the unique solution to the verification equation. On the other hand, the linear proof is uniquely determined by $\boldsymbol{k}$, so $[\pi]_1$ follows exactly the same distribution as its honest counterpart.

5.3 Simplifications and Improvements

For simplicity, we have omitted some improvements of our argument which reduce significantly the size of the structured crs.

1. In the quadratic crs, the terms $\{[s^i]_2\}_{i=1}^N$ are never used by the prover (in fact, they are redundant and can be derived from the other terms) and are just included to simplify the exposition of the update algorithm.
2. A closer examination of our proof reveals that the only point where we need a structured common reference string is for proving the quadratic equations. For the linear proof, what we need is that the columns of $\mathbf{Q}$ are independent generically. Therefore, the Lagrangians $\lambda_i(s)$, for $i = 1, \dots, n$ can be replaced by uniform random elements in $\mathbb{Z}_p^*$. The interpolation set $\mathcal{R}$ can be chosen to be of cardinal M, the Lagrangians of degree $M-1$ and the target polynomial of degree M, for some M which is a bound on the number of multiplication gates. This seems like a really useful simplification, as the non-structured elements of the crs can be chosen in some public way without a trusted procedure and do not need to be updated.
3. Finally, the specialized crs for a circuit ϕ does not need to include all the quadratic crs Σ_{Φ}, but the terms which of degree larger than m, the number of multiplication gates of ϕ, can be eliminated.

Putting all these improvements together, the quadratic universal crs is

$$\Sigma_{\Phi} = (gk, \{[\lambda_i(s)]_{1,2}\}_{i=1}^M, \{[s^i]_1\}_{i=1}^{M-1}, [t(s)]_{1,2}, \{[u_i]_1\}_{i=1}^{N_{pub}}),$$

where M is a bound on the number of multiplication gates, N is a bound on the public input length and $\{[u_i]_1\}_{i=1}^N$ are uniformly random elements of $\mathbb{Z}_p^*$,

The matrix $\mathbf{Q}$ associated to a certain circuit ϕ is:

$$\mathbf{Q} = \begin{pmatrix} \mathbf{U}_{pub} & \mathbf{0} & \boldsymbol{\Lambda}_{opt} & 0 & 0 & 0 \\ \mathbf{0} & \boldsymbol{\Lambda}_{mid} & \boldsymbol{\Lambda}_{opt} & t(s) & 0 & 0 \\ \mathbf{V}_{pub} & \mathbf{V}_{mid} & \mathbf{0} & 0 & t(s) & 0 \\ \mathbf{W}_{pub} & \mathbf{W}_{mid} & \mathbf{0} & 0 & 0 & t(s) \end{pmatrix}, \qquad \boldsymbol{q}_0 = \begin{pmatrix} 0 \\ 0 \\ v_0(s) \\ w_0(s) \end{pmatrix}, \tag{8}$$

where $\mathbf{U}_{pub} = (u_1, \dots, u_{n_{pub}})$, $\boldsymbol{\Lambda}_{mid} = (0, \dots, 0, \lambda_1(s), \dots, \lambda_{m-n'}(s))$, $\boldsymbol{\Lambda}_{opt} = (\lambda_{m-n'+1}(s), \dots, \lambda_m(s))$, $\mathbf{V}_{pub} = (v_1(s), \dots, v_{n_{pub}}(s))$, $\mathbf{V}_{mid} = (v_{n_{pub}+1}(s), \dots, v_{n+m-n'}(s))$, $\mathbf{W}_{pub} = (w_1(s), \dots, w_{n_{pub}}(s))$, $\mathbf{W}_{mid} = (w_{n_{pub}+1}(s), \dots, w_{n+m-n'}(s))$, and the sets of polynomials $\mathcal{V} = \{v_i(X)\}_{i=0}^{n+m}, \mathcal{W} = \{w_i(X)\}_{i=0}^{n+m}$ are the ones associated to the circuit as specified in Lemma 1 and Lemma 3.

The crs for the specialized circuit is

$$\Sigma_{\Phi} = (gk, \{[\lambda_i(s)]_{1,2}\}_{i=1}^m, \{[s^i]_1\}_{i=1}^{m-1}, [t(s)]_{1,2}, \{[u_i]_1\}_{i=1}^{n_{pub}}, [\mathbf{Q}]_1, [\boldsymbol{q}_0]_1, \phi, n_{pub}, \sigma_{\phi,\mathsf{lin}}),$$

where $\sigma_{\phi,\mathsf{lin}}$ is defined as before.

The update of the linear crs is defined as before. For the quadratic part, since now $[s^i]_2$ is not included in the crs, we have to show how to compute $[\lambda_i(ss_1)]_2$ for any $s_1 \in \mathbb{Z}_p$ and $[\lambda_i(s)]_2$.

For any polynomial $z(X)$ of degree at most $M-1$, $z(X) = \sum_{j=1}^{m} \lambda_j(X) z(r_j)$. Therefore, if we define $\hat{\lambda}_i(X)$ as the unique polynomial of degree $m-1$ such that $\hat{\lambda}_i(X) = \lambda_i(Xs_1)$,

$$\lambda_i(s \cdot s_1) = \hat{\lambda}_i(s) = \sum_{j=1}^{m} \lambda_j(s) \hat{\lambda}_i(r_j) = \sum_{j=1}^{m} \lambda_j(s) \left(\prod_{k=1, k \neq i}^{m} \frac{r_j s_1 - r_k}{r_i - r_k} \right).$$

Therefore, $[\lambda_i(ss_1)]_2$ can be computed from $[\lambda_j(s)]_2$, $j = 1, \ldots, m$, s_1 and $\mathcal{R}$.

On the other hand, to verify the correctness of the update algorithm, it suffices to observe that $[s^j]_2 = \sum_{i=n+1}^{n+m} r_i^j [\lambda_i(s)]_2$. The checks of the update algorithm can be done using this expression for the checks which involve $[s^j]_2$.

6 An New Argument under Weaker Assumptions

In this section we describe our construction for proving correct evaluation of an arithmetic circuit. As explained in the introduction, it makes use of two subarguments: a quadratic and a linear "knowledge transfer" subarguments. The reason why we use the term "knowledge transfer" is because these arguments will ensure that, if the prover knows a witness for the circuit evaluation up to level i which is also a valid opening up to level i of a set of shrinking commitments to the corresponding wires, it also knows a valid opening to the commitments of the wires at level $i+1$.

Since the input of the circuit is public, the idea is that these arguments allow to "transfer" the knowledge of the witness for correct evaluation (a consistent assignment to all wires) to lower levels of the circuit. Any adversary against soundness needs to break the "chain" of consistent evaluations at some point and thus, break the soundness of one of the two subarguments. This technique allows us to avoid using binding commitments to the wires at each level, while still being able to define what it means to break soundness. Intuitively, the difficulty we have to circumvent is to reason about wether the openings of shrinking commitments satisfy a certain equation without assuming that the adversary is generic, as there are many possible such openings.

Such intuitive notion of knowledge transfer argument can be formalized as a promise problem defined by a language of good instances $\mathcal{L}_{YES}$ and of bad instances $\mathcal{L}_{NO}$. Completeness guarantees that proofs are accepted for all instances of $\mathcal{L}_{YES}$, while soundness guarantees that no argument will be accepted for instances of $\mathcal{L}_{NO}$. Nothing is claimed when $x \notin (\mathcal{L}_{YES} \cup \mathcal{L}_{NO})$. In our case, membership in $\mathcal{L}_{YES}$ and $\mathcal{L}_{NO}$ can be efficiently decided with a number of operations which is proportional to the size of the statement, but the reason why our subarguments are interesting (and not trivial) is that the verifier does only a constant number of public key operations (ignoring the need to read the full statement).

We note that the two subarguments are not zero-knowledge and in fact their soundness proof heavily relies on the fact that the commitments given by the prover are deterministic.

This section is organized as follows: we first present the description of our argument in terms of the two subarguments, then we present in detail each of the subarguments, give the proof of security and finally we discuss its efficiency.

6.1 Argument Description

In this section we describe our construction for proving correct evaluation of an arithmetic circuit. We use a simplified syntax and consider simply 3 algorithms ($\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify}$) and we assume that Setup takes as input a relation $R_\phi = \{(\boldsymbol{x}, \boldsymbol{y}) : \phi(\boldsymbol{x}) = \boldsymbol{y}\}$ for some circuit $\phi : \mathbb{Z}_p^{n_0} \to \mathbb{Z}_p^{n_d}$ as described in Sect. 4.1. Let $n_{max} = \max_{i \in [d]} n_i$. This is only for simplicity and our scheme could be trivially written using the syntax of Sect. 3 and be fully updatable and have a partially universal crs.

$\mathsf{Setup}(R_\Phi)$**:** Generate the CRS for the quadratic knowledge transfer argument, defined in section 6.2, crs_Φ which includes $\{[\lambda_i(s)]_{1,2}\}_{i=1}^{n_{max}}$. Generate also a crs for the linear knowledge transfer argument, defined in fig. 3, crs_ϕ for proving membership in the space $([\mathbf{M}^\top]_1, [\mathbf{N}^\top]_1, [\mathbf{P}^\top]_2)$, where matrices $\mathbf{M}, \mathbf{N}, \mathbf{P}$ are matrices which define the affine constraints as defined in equations (11),(12),(13), respectively.

```
K(gk, {[s^i]_1, [s^j]_2}_{i∈[m−1], j∈[m]}):
    Output crs =
    (gk, {[λ_1(s)]_γ, ..., [λ_m(s)]_γ}_{γ∈{1,2}},
    {[s^i]_1}_{i∈{1,...,m−2}}, [t(s)]_2).

V(crs, a, b, [L]_1, [R]_2, [O]_1, [H]_1):
    Check if:
    e([L]_1, [R]_2) − e([O]_1, [1]_2) = e([H]_1, [t(s)]_2);
    output 1 in this case and 0 otherwise.

P(crs, a, b):
    ℓ(X) = Σ_{i=1}^m a_i λ_i(X);
    r(X) = Σ_{i=1}^m b_i λ_i(X);
    o(X) = Σ_{i=1}^m c_i λ_i(X);
    h(X) = (ℓ(X)r(X) − o(X))/t(X);
    [L]_1 = [ℓ(s)]_1; [R]_2 = [r(s)]_2;
    [O]_1 = [o(s)]_1; [H]_1 = [h(s)]_1;
    Output [H]_1.
```

Fig. 2. Our argument for componentwise product. $\lambda_i(X)$ is the ith Lagrange polynomial associated to $\mathcal{R}$, $t(X)$ is the polynomial which has as roots all the elements of $\mathcal{R}$. Both $\boldsymbol{a}$ and $\boldsymbol{b}$ are m-dimensional vectors in $\mathbb{Z}_p$.

$\mathsf{Prove}(\text{crs}, (\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}) \in \mathcal{R}_\phi)$**:** Given the input $\boldsymbol{x}$, the output $\boldsymbol{y}$, and $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$ a valid assignment to left, right and output wires as described in Lemma 4, the prover proceeds as follows:

1. For each $i \in \{1, \ldots, d\}$, commit to $\boldsymbol{a}_i, c_i$ in $\mathbb{G}_1$ and to $\boldsymbol{b}_i$ in $\mathbb{G}_2$ as: $[L_i]_1 = \sum_{j=1}^{n_i} a_{i,j}[\lambda_i]_1 = [\boldsymbol{\Lambda}_i]_1 \boldsymbol{a}_i$, $[R_i]_2 = \sum_{j=1}^{n_i} b_{i,j}[\lambda_i]_2 = [\boldsymbol{\Lambda}_i]_2 \boldsymbol{b}_i$, $[O_i]_1 = \sum_{j=1}^{n_i} c_{i,j}[\lambda_i]_1 = [\boldsymbol{\Lambda}_i]_1 \boldsymbol{c}_i$.
2. (Quadratic Constraints) For each $i \in \{1, \ldots, d\}$, compute a proof Π_Q that the vector $\boldsymbol{a}_i \circ \boldsymbol{b}_i$, which is the componentwise product of the openings of $[L_i]_1, [R_i]_2$, is an opening of $[O_i]_1$.
3. (Linear Constraints) For all $i \in \{1, \ldots, d\}$, compute a proof Π_L that $[L_i]_1$ and $[R_i]_2$ are commitments to the correct evaluation of all the left and right wires at level i, that is, that they satisfy the affine linear constraints which relate them to the outputs of gates at levels $j = 0, \ldots, i-1$.
4. Output $(\mathcal{C} = \{[\boldsymbol{L}]_1, [\boldsymbol{R}]_2, [\boldsymbol{O}]_1\}_{i=1}^d, \Pi_Q, \Pi_L)$ as the proof.

$\mathsf{Verify}(\text{crs}, (\boldsymbol{x}, c), (\mathcal{C}, \Pi_Q, \Pi_L))$**:** Output 1 if the following two checks are successful and 0 otherwise:

1. Verify Π_Q, Π_L.
2. Check that $[O_d]_1 = \sum_{i=1}^{n'} [\lambda_i]_1 y_i$.

The proof of security is deferred to Sect. 6.4.

6.2 Aggregated Argument for Quadratic Constraints

Let $m \in \mathbb{N}$. We give an argument for the promise problems defined by languages $\mathcal{L}_{YES}^{\mathsf{quad}}, \mathcal{L}_{NO}^{\mathsf{quad}}$, which are parameterized by $m \in \mathbb{N}$ and a multiPedersen commitment key $ck := ([\boldsymbol{\Lambda}]_1, [\boldsymbol{\Lambda}]_2)$ and are defined as

$$\mathcal{L}_{YES}^{\mathsf{quad}} := \left\{ \begin{array}{l} (\boldsymbol{a}, \boldsymbol{b}, [L]_1, [R]_2, [O]_1) : \boldsymbol{c} = \boldsymbol{a} \circ \boldsymbol{b} \\ \text{ and } [L]_1 = [\boldsymbol{\Lambda}]_1 \boldsymbol{a}, [R]_2 = [\boldsymbol{\Lambda}]_2 \boldsymbol{b}, [O]_1 = [\boldsymbol{\Lambda}]_1 \boldsymbol{c} \end{array} \right\},$$

$$\mathcal{L}_{NO}^{\mathsf{quad}} := \left\{ \begin{array}{l} (\boldsymbol{a}, \boldsymbol{b}, [L]_1, [R]_2, [O]_1) : \boldsymbol{c} = \boldsymbol{a} \circ \boldsymbol{b}, \\ [L]_1 = [\boldsymbol{\Lambda}]_1 \boldsymbol{a} \text{ and } [R]_2 = [\boldsymbol{\Lambda}]_2 \boldsymbol{b}, \\ \text{but } [O]_1 \neq [\boldsymbol{\Lambda}]_1 \boldsymbol{c} \end{array} \right\}.$$

Perfect completeness. The argument described in Fig. 2 has perfect completeness as the values $[L]_1, [O]_1$ can be computed from $\{[\lambda_i(s)]_1 \ldots, [\lambda_m(s)]_1\}$, and $[R]_1$ from $\{[\lambda_i(s)]_2 \ldots, [\lambda_m(s)]_2\}$. Further, by definition, the polynomial $L(X)R(X) - O(X)$ takes the value $a_i b_i - c_i = 0$ at point $r_i \in \mathcal{R}$. Therefore, $L(X)R(X) - O(X)$ is divisible by $t(X)$, so $H(X)$ is well defined. Further, the degree of H is at most $m-2$ (since $L(X)R(X)$ has degree $2m-2$ and $t(X)$ has degree m) and thus $[H]_1$ can be computed from $\{[s]_1, \ldots, [s^{m-2}]_1\}$.
Computational Soundness. We argue that if $\mathcal{A}$ produces an accepting proof for $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, [L]_1, [R]_2, [O]_1) \in \mathcal{L}_{NO}^{\mathsf{quad}}$ then we can construct an adversary $\mathcal{B}$ against the $(\mathcal{R}, \boldsymbol{m})$-Rational Strong Diffie-Hellman Assumption. Given a challenge $gk, \{[s^i]_1\}_{i=1}^{m-1}, \{[s^i]_2\}_{i=1}^{m}$, adversary $\mathcal{B}$ can simulate the common reference string perfectly

because $\lambda_i(X)$ is a polynomial whose coefficients in $\mathbb{Z}_p$ depend only on $\mathcal{R}$ of degree at most $m-1$. Therefore, $[\lambda_i(s)]_1, [\lambda_i(s)]_2$ can be computed from $\{s^i\}_{i=1}^{m-1}$ in both the source groups. On the other hand, $t(X)$ is a polynomial with coefficients in $\mathbb{Z}_p$ which depend only on $\mathcal{R}$ of degree at most m. So $[t(s)]_2$ can be computed in $\mathbb{G}_2$ given $\{[s^i]_2\}_{i=1}^{m}$.

Adversary $\mathcal{A}$ outputs $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, [L]_1, [R]_2, [O^\dagger]_1, [H^\dagger]_1)$ which is accepted by the verifier and $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, [L]_1, [R]_2, [O^\dagger]_1) \in \mathcal{L}_{NO}^{\mathsf{quad}}$, which in particular means that, for $L = L(s)$, $R = R(s)$, the equation

$$e([L]_1, [R]_2) - e([O^\dagger]_1, [1]_2) = e([H^\dagger]_1, [t(s)]_2) \tag{9}$$

holds but $O^\dagger \neq O(s)$.

Since adversary $\mathcal{B}$ received $\boldsymbol{a}, \boldsymbol{b}$ as part of $\mathcal{A}$'s output, it can run the honest prover algorithm and obtain O, H which satisfy that

$$e([L]_1, [R]_2) - e([O]_1, [1]_2) = e([H]_1, [t(s)]_2) \tag{10}$$

and $O = O(s)$.

Subtracting equations (9) and (10), we get $e([O^\dagger - O]_1, [1]_2) = e([H^\dagger - H]_1, [t(s)]_2)$. Therefore, $([O^\dagger - O]_1, [H^\dagger - H]_1)$ is a solution to the $(\mathcal{R}, \boldsymbol{m})$-Rational Strong Diffie-Hellman Assumption.

We note that the verification algorithm never uses $(\boldsymbol{a}, \boldsymbol{b})$ which are part of the statement. When using the scheme as a building block, we omit $(\boldsymbol{a}, \boldsymbol{b})$ from the input of the verifier of the quadratic relations.

6.3 Aggregated Argument for Affine Constraints

In this section we show how to prove that the linear constraints which express correct circuit evaluation We show that if we have some commitments O_i to known values $\boldsymbol{c}_i$, until a certain depth j, then we can guarantee that O_{j+1} opens to a fixed linear combination of $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_j$. This can be used to prove that if all the commitment to the output gates are correct up to level j, then certain commitments L_{j+1}, (respectively R_{j+1}) open to left wires (resp. right wires) at level $j+1$.

Encoding Affine Constraints as Membership in Linear Spaces.

$$\begin{pmatrix} \boldsymbol{x} \\ O_1 \\ O_2 \\ O_3 \\ \vdots \\ O_{d-1} \end{pmatrix} = \begin{pmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Lambda}_1 & \mathbf{0} & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{\Lambda}_2 & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{\Lambda}_3 & & \mathbf{0} \\ \vdots & \vdots & \vdots & & \ddots & \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \ldots & \boldsymbol{\Lambda}_{d-1} \end{pmatrix} \begin{pmatrix} \boldsymbol{c}_0 \\ \boldsymbol{c}_1 \\ \boldsymbol{c}_2 \\ \boldsymbol{c}_3 \\ \vdots \\ \boldsymbol{c}_{d-1} \end{pmatrix} \tag{11}$$

$$\begin{pmatrix} L_1 \\ L_2 \\ L_3 \\ \vdots \\ L_d \end{pmatrix} = \begin{pmatrix} \mathbf{F}_{1,0} & \mathbf{0} & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{F}_{2,0} & \mathbf{F}_{2,1} & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{F}_{3,0} & \mathbf{F}_{3,1} & \mathbf{F}_{3,2} & \ldots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{F}_{d-1,0} & \mathbf{F}_{d-1,1} & \mathbf{F}_{d-1,2} & \ldots & \mathbf{F}_{d-1,d-2} \end{pmatrix} \begin{pmatrix} \boldsymbol{c}_0 \\ \boldsymbol{c}_1 \\ \boldsymbol{c}_2 \\ \vdots \\ \boldsymbol{c}_{d-1} \end{pmatrix} + \begin{pmatrix} \hat{L}_1 \\ \hat{L}_2 \\ \hat{L}_3 \\ \vdots \\ \hat{L}_d \end{pmatrix}, \tag{12}$$

$$\begin{pmatrix} R_1 \\ R_2 \\ R_3 \\ \vdots \\ R_{d-1} \end{pmatrix} = \begin{pmatrix} \mathbf{G}_{1,0} & \mathbf{0} & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{G}_{2,0} & \mathbf{G}_{2,1} & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{G}_{3,0} & \mathbf{G}_{3,1} & \mathbf{G}_{3,2} & \ldots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{G}_{d-1,0} & \mathbf{G}_{d-1,1} & \mathbf{G}_{d-1,2} & \ldots & \mathbf{G}_{d-1,d-2} \end{pmatrix} \begin{pmatrix} \boldsymbol{c}_0 \\ \boldsymbol{c}_1 \\ \boldsymbol{c}_2 \\ \vdots \\ \boldsymbol{c}_{d-1} \end{pmatrix} + \begin{pmatrix} \hat{R}_1 \\ \hat{R}_2 \\ \hat{R}_3 \\ \vdots \\ \hat{R}_d \end{pmatrix}, \tag{13}$$

Before we give details of our argument we write in matrix form the expression of $(\boldsymbol{x}, [\boldsymbol{O}]_1, [\boldsymbol{L}]_1, [\boldsymbol{R}]_2)$ in terms of the internal wires of the circuit, following section 4.1. The commitments to the output values $[\boldsymbol{O}]_1$

```
$\underline{\mathsf{K}(gk, [\mathbf{M}]_1, [\mathbf{N}]_1, [\mathbf{P}]_2, \phi, \{[s^i]_1, [s^j]_2\}_{i\in[m-1], j\in[m]})}$:
    $\mathbf{K}_0 \leftarrow \mathbb{Z}_p^{n_0+d-1\times 2}$; $\mathbf{K}_1 \leftarrow \mathbb{Z}_p^{d\times 2}$,$\mathbf{K}_2 \leftarrow \mathbb{Z}_p^{d\times 2}$;
    Sample $\overline{\mathbf{A}} \leftarrow \overline{\mathcal{D}}_2$;
    Sample $\mathbf{\Gamma} \leftarrow \mathbb{Z}_p^{n\times 2}$;
    Output crs $= \Big(gk, [\mathbf{M}^\top \mathbf{K}_0 + \mathbf{N}^\top \mathbf{K}_1 + \mathbf{\Gamma}]_1,$
        $[\mathbf{P}^\top \mathbf{K}_2 - \mathbf{\Gamma}]_2, \left[\begin{pmatrix}\mathbf{K}_0\overline{\mathbf{A}}\\ \mathbf{K}_1\overline{\mathbf{A}}\end{pmatrix}\right]_2, [\mathbf{K}_2\overline{\mathbf{A}}]_1\Big)$.

$\underline{\mathsf{P}(\mathrm{crs}, \boldsymbol{x}, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})}$:
    $[\boldsymbol{u}]_1 = \begin{pmatrix}[\mathbf{M}]_1\\ [\mathbf{N}]_1\end{pmatrix}\boldsymbol{c}$ ;
    $[\boldsymbol{v}]_2 = [\mathbf{P}]_2\boldsymbol{c}$ ;
    $\boldsymbol{\rho} \leftarrow \mathbb{Z}_p^2$;
    $[\boldsymbol{\pi}]_1 = \boldsymbol{c}^\top[\mathbf{M}^\top\mathbf{K}_0 + \mathbf{N}^\top\mathbf{K}_1 + \mathbf{\Gamma}]_1 + [\boldsymbol{\rho}]_1$;
    $[\boldsymbol{\theta}]_2 = \boldsymbol{c}^\top[\mathbf{P}^\top\mathbf{K}_2 - \mathbf{\Gamma}]_2 - [\boldsymbol{\rho}]_2$;
    output
        $([\boldsymbol{u}]_1, [\boldsymbol{v}]_2, [\boldsymbol{\pi}]_1, [\boldsymbol{\theta}]_2)$;

$\underline{\mathsf{V}(\mathrm{crs}, \boldsymbol{x}, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, [\boldsymbol{u}]_1, [\boldsymbol{v}]_2, [\boldsymbol{\pi}]_1, [\boldsymbol{\theta}]_2)}$:
    Check if:
    $e([\boldsymbol{\pi}]_1, [\overline{\mathbf{A}}]_2) - e([\boldsymbol{u}]_1, \left[\begin{pmatrix}\mathbf{K}_0\overline{\mathbf{A}}\\ \mathbf{K}_1\overline{\mathbf{A}}\end{pmatrix}\right]_2)$
        $= e([\boldsymbol{\theta}]_2, [\overline{\mathbf{A}}]_1) - e([\boldsymbol{v}]_2, [\mathbf{K}_2\overline{\mathbf{A}}]_1)$;

    output 1 in this case and 0 otherwise.
```

Fig. 3. The Lin argument for proving that the vector is well formed. The argument is just a rewriting of the argument of membership in linear spaces of [20], but we will be proving a stronger notion of soundness. The matrix $\overline{\mathbf{A}}$ is sampled from a distribution $\mathcal{D}_2$ such that the $\mathcal{D}_2$-SKerMDH Assumption holds, and eliminating the last row. In particular, we can choose $\overline{\mathbf{A}}$ to be a random diagonal matrix.

should be such that $[\boldsymbol{O}_i]_1 = [\mathbf{\Lambda}_i]_1\boldsymbol{c}_i$, where $\mathbf{\Lambda}_i = (\lambda_1(s), \ldots, \lambda_{n_i}(s))$, and the input $\boldsymbol{x} = \boldsymbol{c}_0$ is public. These constraints can be expressed in matrix form in equation (11) We denote the matrix on the right hand side of (11) as $\mathbf{M}$, so this equation reads $\left(\begin{smallmatrix}\boldsymbol{x}\\ \boldsymbol{O}\end{smallmatrix}\right) = \mathbf{M}\boldsymbol{c}$. On the other hand, the constraints satisfied by the left wires in terms of the output wires of previous levels can be written in matrix form as shown in equation (12): that is, for each i, $L_i = \sum_{k=1}^{i-1}\mathbf{F}_{i,k}\boldsymbol{c}_k + \hat{L}_i$, where

$$\mathbf{F}_{i,k} = \left(\sum_{j=1}^{n_k} f_{i,j,k,1}\lambda_j(s), \sum_{j=1}^{n_k} f_{i,j,k,2}\lambda_\ell(s), \ldots \sum_{j=1}^{n_k} f_{i,j,k,n_k}\lambda_j(s)\right) \tag{14}$$

and $\hat{L}_i = \sum_{j=1}^{n_i} f_{i,j}\lambda_j(s)$, for the constants which are defined in Lemma 4. We denote the matrix on the right hand side of equation (12) as $\mathbf{N}$, so this equation reads $\boldsymbol{L} = \mathbf{N}\boldsymbol{c} + \hat{\boldsymbol{L}}$. The constraints satisfied by the right wires in terms of the output wires of previous levels can be written in a similar form as shown in equation (13) that is, for each i, $R_i = \sum_{k=1}^{i-1}\mathbf{G}_{i,k}\boldsymbol{c}_k + \hat{R}_i$, where

$$\mathbf{G}_{i,k} = \left(\sum_{j=1}^{n_k} g_{i,j,k,1}\lambda_j(s), \sum_{j=1}^{n_k} g_{i,j,k,2}\lambda_\ell(s), \ldots \sum_{j=1}^{n_k} g_{i,j,k,n_k}\lambda_j(s)\right), \tag{15}$$

and $\hat{R}_i = \sum_{j=1}^{n_i} g_{i,j}\lambda_j(s)$. We denote the matrix on the right hand side of equation (13) as $\mathbf{P}$, so this equation reads $\boldsymbol{R} = \mathbf{P}\boldsymbol{z} + \hat{\boldsymbol{R}}$.

With the notation defined, satisfaction of the affine constraints can be written as $\begin{pmatrix}[\boldsymbol{O}']_1\\ [\boldsymbol{L}]_1 - [\hat{\boldsymbol{L}}]_1\\ [\boldsymbol{R}]_2 - [\hat{\boldsymbol{R}}]_2\end{pmatrix} \in \mathbf{Im}\begin{pmatrix}[\mathbf{M}]_1\\ [\mathbf{N}]_1\\ [\mathbf{P}]_2\end{pmatrix}$, where $[\boldsymbol{O}'^\top]_1 = ([\boldsymbol{x}^\top]_1, [\boldsymbol{O}^\top]_1)$. That is, the linear constraints are satisfied if a certain vector is in a subspace generated by some matrix which depends on the circuit. This can be proven with a QANIZK argument for bilateral spaces (linear spaces with components in different source groups) due to [20]. Note that the matrices $[\mathbf{M}]_1, [\mathbf{N}]_1, [\mathbf{P}]_2$ which define the language are witness samplable (they can be sampled along with their discrete logarithm), so we can use the most efficient of the two arguments described in [20]. The proof size depends on the choice of a secure $\mathcal{D}_k$-Split Kernel Diffie-Hellman Assumption (Assumption 1). The minimal proof size is $2|\mathbb{G}_1| + 2|\mathbb{G}_2|$ (choosing $k = 2$, since the assumption is insecure when $k = 1$).

The full argument adapted to our setting is described in Figure (3). Perfect completeness, perfect zero-knowledge and computational soundness under the Split Kernel Assumption, is proven in [20]. The argument is very close to the argument of membership in (unilateral) linear spaces of [34] for witness samplable matrices, but where the information is divided in different groups $\mathbb{G}_1, \mathbb{G}_2$. Since part of the argument of [34] is information theoretic, the key step in the proof of [20] is to make sure that this splitting in two groups does not leak additional information.

We note that the verification algorithm never uses $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$ which are part of the statement. When using the scheme as a building block, we omit $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$ from the input of the verifier of the linear relations.

Intuition: Standard soundness is not enough. At first sight, it might look that the soundness property of the argument of membership in linear spaces is sufficient to show that the affine constraints are satisfied. However, it only guarantees that there is a witness which satisfies all the linear constraints. Since the commitments are perfectly hiding there might be several possible openings and the adversary could do an opening "switch", starting with the opening determined by the input and switch to another possible input in some step.

More in detail, for some input $\boldsymbol{x}$, let the witness for correct evaluation be $(\boldsymbol{a}^\dagger, \boldsymbol{b}^\dagger, \boldsymbol{c}^\dagger)$. If an adversary $\mathcal{A}$ is capable of finding another vector $\boldsymbol{c}$ such that $\left(\begin{smallmatrix}\boldsymbol{x}\\ [\boldsymbol{O}]_1\end{smallmatrix}\right)\boldsymbol{c}^\dagger = \left(\begin{smallmatrix}\boldsymbol{x}\\ [\boldsymbol{O}]_1\end{smallmatrix}\right)\boldsymbol{c}$, then the adversary can construct the values L_i, R_i using $\boldsymbol{c}$, and prove that $\begin{pmatrix}[\boldsymbol{x}]_1\\ [\boldsymbol{O}]_1\\ [\boldsymbol{L}]_1-[\hat{\boldsymbol{L}}]_1\\ [\boldsymbol{R}]_2-[\hat{\boldsymbol{R}}]_2\end{pmatrix} = \begin{pmatrix}[\mathbf{M}]_1\\ [\mathbf{N}]_1\\ [\mathbf{P}]_2\end{pmatrix}\boldsymbol{c}$. This will convince the verifier that the output of evaluation of the circuit is $\boldsymbol{c}_d$ instead of the right value $\boldsymbol{c}_d^\dagger$, although the adversary did not break the soundness of the proof of membership in bilateral spaces.

Now the attack we just described seems easy to rule out, because the Lagrangian Pedersen commitments are computationally binding and such a pair $(\boldsymbol{c}, \boldsymbol{c}^\dagger)$ cannot be found efficiently. However, there is no easy reduction of the attack to breaking the binding property of the commitments. Since $\boldsymbol{x}$ is public, $\boldsymbol{c}^\dagger$ can be computed from $\boldsymbol{x}$ but the output of the adversary is too short to extract $\boldsymbol{c}$, so we do not know how to prove that it broke the binding property.

Formal Analysis. We now define the security properties satisfied by the argument in figure 3 and analyze its security. The argument satisfies completeness and soundness for the promise problem will show that the QANIZK argument of membership in linear spaces as described is an argument for the promise problem defined by the languages

$$\mathcal{L}_{YES}^{\mathsf{lin}} = \{(\boldsymbol{c}, [\boldsymbol{O}']_1, [\boldsymbol{L}]_1, [\boldsymbol{R}]_2) : [\boldsymbol{O}]_1 = [\mathbf{M}]_1\boldsymbol{c}, [\boldsymbol{L}]_1 = [\mathbf{N}]_1\boldsymbol{c}, [\boldsymbol{R}]_1 = [\mathbf{P}]\boldsymbol{c}\}$$
$$\mathcal{L}_{NO}^{\mathsf{lin}} = \left\{\begin{array}{c}(\boldsymbol{c}, [\boldsymbol{O}']_1, [\boldsymbol{L}]_1, [\boldsymbol{R}]_2) : \exists i^* \leq d \text{ s.t. } [\overline{\boldsymbol{O}}_{i^*}]_1 = [\overline{\mathbf{M}}_{i^*}]_1\boldsymbol{c} \text{ but}\\ [L_{i^*+1}]_1 \neq [\mathbf{F}_{i^*+1}]_1\boldsymbol{c} \text{ or } [R_{i^*+1}]_2 \neq [\mathbf{G}_{i^*+1}]_2\boldsymbol{c}\end{array}\right\},$$

where $\overline{\mathbf{X}}_i$ denotes the first i rows of matrix $\mathbf{X}$.

To do so we assume that the membership proof given in Figure (3) satisfies an additional property when the matrices $[\mathbf{M}]_1, [\mathbf{N}]_1, [\mathbf{P}]_2$ are sampled from the distribution specified above (which depends on the constants of the circuit and the choice of s).

Given an adversary that produces a valid proof for a statement in $\mathcal{L}_{NO}^{\mathsf{lin}}$, successful attacks can be divided in two categories.

Type I: In this attack $\begin{pmatrix}[\boldsymbol{u}]_1\\ [\boldsymbol{v}]_2\end{pmatrix}$ is not in the image of $\mathbf{Q} = \begin{pmatrix}[\mathbf{M}]_1\\ [\mathbf{N}]_1\\ [\mathbf{P}]_2\end{pmatrix}$.

Type II: In this type of attack, the vector is in the right subspace.

The subargument is sound under the $\mathcal{D}_k$-$\mathsf{SKerMDH}$ Assumption, which rules out Type I attacks, and the "tautological" assumption that Type II attacks are infeasible. We note that it is falsifiable, as one can sample $\mathbf{Q}, \mathbf{K}$ with the right distribution and decide whether the adversary has been successful with Type II attack.

Definition 5. *(Hardness of Type II attacks) Let $R_\phi \in R_\Phi$. For the matrices $([\mathbf{M}]_1, [\mathbf{N}]_1, [\mathbf{P}]_2)$ chosen according to the distribution specified above, every polynomial time prover of the* Lin *argument has negligible probability of outputting a tuple $(\boldsymbol{c}, [\boldsymbol{O}']_1, [\boldsymbol{L}]_1, [\boldsymbol{R}]_2, [\boldsymbol{\pi}]_1, [\boldsymbol{\theta}]_2)$ such that:*

– $[\boldsymbol{O}']_1, [\boldsymbol{L}]_1, [\boldsymbol{R}]_2, [\boldsymbol{\pi}]_1, [\boldsymbol{\theta}]_2$ *is accepted by the verifier,*
– $([\boldsymbol{O}']_1, [\boldsymbol{L}]_1, [\boldsymbol{R}]_2) \in \mathrm{Im}(\mathbf{Q})$.
– $(\boldsymbol{c}, [\boldsymbol{O}']_1, [\boldsymbol{L}]_1, [\boldsymbol{R}]_2) \in \mathcal{L}_{NO}^{\mathsf{Lin}}$.

An equivalent formulation of the assumption says that it is not possible to prove membership in the image of $\mathbf{D}$ of vectors of a certain form, namely, any vector which has i^* 0's in the first d rows and but has less than i^* zeros in the rows from $d+1, \ldots, 2d$ or in the in the rows from $2d+1, \ldots, 3d$. In the supplementary material we prove that the assumption is generically equivalent to breaking the soundness property of the Pedersen commitment.

Generic Hardness of Type II Attacks The proof can be generalized to any matrix $\overline{\mathbf{A}}$ associated to a kernel assumption, but for simplicity in the analysis we will choose $\overline{\mathbf{A}}$ to be a random diagonal matrix in $\mathbb{Z}_p^{2\times 2}$.

Theorem 1. *An adversary $\mathcal{A}$ successful against type II attacks can be used to construct an adversary $\mathcal{B}$ which receives $(gk, \{[s^i]_1\}_{i=1}^{n_{max}-1}, \{[s^j]_2\}_{j=1}^{n_{max}}$ and outputs $\hat{\boldsymbol{w}}$ such that $\boldsymbol{\Lambda}\hat{\boldsymbol{w}} = 0$, for $\boldsymbol{\Lambda} = (\lambda_1(s), \ldots, \lambda_{n_{max}}(s))$ in the asymmetric generic bilinear group model.*

Proof. Adversary $\mathcal{A}$ receives $(gk, \{[s^i]_1\}_{i=1}^{n_{max}-1}, \{[s^j]_2\}_{j=1}^{n_{max}}$, and generates the common reference string $\mathsf{crs}_{\mathsf{Lin}}$, which is given to $\mathcal{A}$. Eventually, $\mathcal{A}$ outputs a successful Type II attack $(\boldsymbol{c}, [\boldsymbol{u}]_1, [\boldsymbol{v}]_2, [\boldsymbol{\pi}]_1, [\boldsymbol{\theta}]_2)$, where $\boldsymbol{u}^\top = (\boldsymbol{x}^\top, \boldsymbol{O}^\top, \boldsymbol{L}^\top)$ and $\boldsymbol{v} = \boldsymbol{R}$.

For simplicity, we introduce the following notation:

$$\mathbf{K}_u = \begin{pmatrix}\mathbf{K}_0\\ \mathbf{K}_1\end{pmatrix}, \mathbf{K}_v = \begin{pmatrix}\mathbf{K}_2\end{pmatrix}, \mathbf{Q}_1 = \begin{pmatrix}\mathbf{M}\\ \mathbf{N}\end{pmatrix}, \mathbf{Q}_2 = \begin{pmatrix}\mathbf{P}\end{pmatrix}$$

.

We show that if $\mathcal{A}$ is generic, we can extract from the adversary some vector $\boldsymbol{w}$ such that $\boldsymbol{u} = \mathbf{Q}_1\boldsymbol{w}, \boldsymbol{u} = \mathbf{Q}_1\boldsymbol{w}$. Together with the vector of consistent evaluations $\boldsymbol{c}$ which can be computed from $\boldsymbol{c}$, this allows to compute an element in the kernel of $\boldsymbol{\Lambda}$.

The crs includes the result of evaluating matrices of polynomials (of the appropriate dimensions) $\overline{\mathbf{A}}(\boldsymbol{Z}) := \begin{pmatrix}Z_{A,1,1} & 0\\ 0 & Z_{A,2,2}\end{pmatrix}$, $\mathbf{K}_u(\boldsymbol{Z}) := (Z_{K_u,j,k})$, $\mathbf{K}_v(\boldsymbol{Z}) := (Z_{K_v,j,k})$, $\boldsymbol{\Gamma}(\boldsymbol{Z}) = (Z_{\Gamma,i,j})$, $\mathbf{M}(Z_S)$, $\mathbf{N}(Z_S)$, $\mathbf{P}(Z_S)$, $\mathbf{Q}_{K,1}(\boldsymbol{Z}) = (\mathbf{Q}_1(X_S))^\top\mathbf{K}_u(\boldsymbol{Z}) + \boldsymbol{\Gamma}(\boldsymbol{Z})$ and $\mathbf{Q}_{K,2}(\boldsymbol{Z}) = (\mathbf{Q}_2(X_S))^\top\mathbf{K}_v(\boldsymbol{Z}) - \boldsymbol{\Gamma}(\boldsymbol{Z})$ at a random point $\boldsymbol{z} = (a_{1,1}, a_{2,2}, k_{u,1,1}, \ldots, \gamma_{n,2}, s)$.

The adversary outputs $[\boldsymbol{u}]_1, [\boldsymbol{v}]_2, [\boldsymbol{\pi}]_1, [\boldsymbol{\theta}]_2$ which are the result of evaluation polynomials $\boldsymbol{u}(\boldsymbol{Z}), \boldsymbol{v}(\boldsymbol{Z}), \boldsymbol{\pi}(\boldsymbol{Z}), \boldsymbol{\theta}(\boldsymbol{Z})$ in $\boldsymbol{z}$, and the coefficients can be efficiently extracted in the generic group model. We now argue that these coefficients are in fact some vector $\boldsymbol{w} \in \mathbb{Z}_p^n$ such that $\boldsymbol{u} = \mathbf{Q}_1\boldsymbol{w}, \boldsymbol{v} = \mathbf{Q}_2\boldsymbol{w}$.

Since the proof produced by the adversary satisfies the verification equation, it follows that

$$\begin{aligned}\boldsymbol{\pi}(\boldsymbol{z})^\top\overline{\mathbf{A}}(\boldsymbol{z}) - \boldsymbol{u}(\boldsymbol{z})^\top\mathbf{K}_u(\boldsymbol{z}) &= \boldsymbol{\theta}(\boldsymbol{z})^\top\overline{\mathbf{A}}(\boldsymbol{z}) - \boldsymbol{v}(\boldsymbol{z})^\top\mathbf{K}_v(\boldsymbol{z})\\ \iff (\boldsymbol{\pi}(\boldsymbol{z}) + \boldsymbol{\theta}(\boldsymbol{z}))^\top\overline{\mathbf{A}}(\boldsymbol{z}) &= \boldsymbol{u}(\boldsymbol{z})^\top\mathbf{K}_u(\boldsymbol{z}) + \boldsymbol{v}(\boldsymbol{z})^\top\mathbf{K}_v(\boldsymbol{z}),\end{aligned}$$

By the Schwartz-Zippel lemma, $\overline{\mathbf{A}}(\boldsymbol{z})$ is invertible and the last equation implies that the following polynomial equation holds with overwhelming probability, which we write as:

$$\pi_j(\boldsymbol{Z}) + \theta_j(\boldsymbol{Z}) = \sum_{i=1}^{n_0+2d} u_i(\boldsymbol{Z})Z_{K_u,i,j} + \sum_{i=1}^{d} v_i(\boldsymbol{Z})Z_{K_v,i,j} \text{ for } j = 1,2 \tag{16}$$

We now use that we are in the asymmetric generic group model and that π_j, u_i (resp. θ_j, v_i) must be constructed as linear combinations of the polynomials which are evaluated in $\mathbb{G}_1$ (resp. in $\mathbb{G}_2$). In particular,

– From equation (16), it follows that π_j must be computed as a linear combination of the polynomials $\mathbf{K}_v\overline{\mathbf{A}}$, $\{\boldsymbol{q}_{1,i}^\top \mathbf{K}_{u,j} + \mathit{\Gamma}_{i,j}\}$, $i = 1, \dots, n$, $j = 1, 2$ which are the polynomials in the first source group which have K terms.
– Similarly, it follows that θ_j must be computed as a linear combination of the polynomials $\mathbf{K}_u\overline{\mathbf{A}}$, $\{\boldsymbol{q}_{2,i}^\top \mathbf{K}_{v,j} - \mathit{\Gamma}_{i,j}\}$, $i = 1, \dots, n$, $j = 1, 2$ which are the polynomials given in the first subgroup which have K terms.
– The latter two points imply that $u_i(\boldsymbol{Z})$, $v_i(\boldsymbol{Z})$ cannot depend on the K-variables, because the left hand side of the equation depends only linearly in these variables. So $u_i(\boldsymbol{Z})$ is a linear combination of $\mathbf{Q}_1(Z_S), Z_S^j$ and $v_i(\boldsymbol{Z})$ is a linear combination of $\mathbf{Q}_2(Z_S), Z_S^j$.
– We conclude that θ_j must be computed as a linear combination of the polynomials $\{\boldsymbol{q}_{2,i}^\top \mathbf{K}_{v,j} - \mathit{\Gamma}_{i,j}\}$ and π_j must be computed as a linear combination of the polynomials $\{\boldsymbol{q}_{1,i}^\top \mathbf{K}_{u,j} + \mathit{\Gamma}_{i,j}\}$. This is because the right hand side does not include terms with $\overline{\mathbf{A}}$. Because of the variables $\mathit{\Gamma}_{i,j}$ must cancel, the coefficients must be equal.
– Finally, we argue that there is a unique set of possible coefficients. This follows because the columns of $\mathbf{Q} = \begin{pmatrix} \mathbf{Q}_1 \\ \mathbf{Q}_2 \end{pmatrix}$ are linearly independent polynomials, this is argued as in Lemma 9.

If the coefficients of the linear combination are called $\boldsymbol{w}$, it follows from the definition that $(\boldsymbol{\pi}(\boldsymbol{z}) + \boldsymbol{\theta}(\boldsymbol{z})) = \mathbf{Q}\boldsymbol{w}$ and they can be extracted from the adversary. Since the adversary breaks soundness $\boldsymbol{w} \neq \boldsymbol{c}$, and there exists some i^* such that $\boldsymbol{w}_{i^*} \neq \boldsymbol{c}_{i^*}$ but $\mathbf{\Lambda}_{i^*}\boldsymbol{w}_{i^*} = \mathbf{\Lambda}_{i^*}\boldsymbol{c}_{i^*}$ and $\boldsymbol{c}_{i^*}$ can be computed from $\boldsymbol{x}$. This concludes the proof.

6.4 Security Proof

We now prove the security of the argument given in Sect. **??**.

Perfect completeness is obvious, because if $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$ is a valid witness for satisfiability, then it satisfies both linear and quadratic constraints.

We sketch the proof of computational soundness. Let $\mathcal{A}$ be an adversary against the soundness of the scheme. We construct adversaries $\mathcal{B}_1$ and $\mathcal{B}_2$ against the quadratic or the linear knowledge transfer arguments, respectively.

Adversary $\mathcal{B}_1$ receives the common reference string of the quadratic subargument $(gk, \{[s^i]_1\}_{i=1}^{n_{max}-1}, \{[s^i]_2\}_{i=1}^{n_{max}})$ and creates the common reference string of the full argument. When it receives an accepting proof $(\mathcal{C} = \{[\boldsymbol{L}]_1, [\boldsymbol{R}]_2, [\boldsymbol{O}]_1\}_{i=1}^d, \mathit{\Pi}_Q, \mathit{\Pi}_L)$ from adversary $\mathcal{A}$ for some statement $(\boldsymbol{x}, \boldsymbol{y})$, adversary $\mathcal{B}_1$ computes the full witness for correct evaluation $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$. It then searches for an index i such that $[L_i]_1$ and $[R_i]_2$ are commitments to $\boldsymbol{a}_i$ and $\boldsymbol{b}_i$ but $[O_i]_1$ is not a valid commitment to $\boldsymbol{a}_i \circ \boldsymbol{b}_i$. If such an i does not exist, it aborts. Else, it returns $(\boldsymbol{a}_i, \boldsymbol{b}_i, [L_i]_1, [R_i]_2, [O_i]_1)$ as an instance of $\mathcal{L}_{NO}$ together with an accepting proof $[H_i]_1$.

Adversary $\mathcal{B}_2$ receives the common reference string of the linear subargument associated to some circuit ϕ, plus the necessary powers of $[s]_{1,2}$ to create the common reference string of the full argument $(gk, \{[s^i]_1\}_{i=1}^{m-1}, \{[s^i]_2\}_{i=1}^{m})$. When it receives an accepting proof $(\mathcal{C} = \{[\boldsymbol{L}]_1, [\boldsymbol{R}]_2, [\boldsymbol{O}]_1\}_{i=1}^d, \mathit{\Pi}_Q, \mathit{\Pi}_L)$ from adversary $\mathcal{A}$ for some statement $(\boldsymbol{x}, c)$, the adversary $\mathcal{B}_2$ computes the full witness $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$. It then searches for an index i such that $[O_1]_1, \dots, [O_{i-1}]_1$ are commitments to $\bar{\boldsymbol{c}}_1, \dots, \bar{\boldsymbol{c}}_{i-1}$ but either $[L_i]_1$ or $[R_i]_2$ are not valid commitments to $\boldsymbol{a}_i$ or $\boldsymbol{b}_i$. If such i does not exist, it aborts, else $(\bar{\boldsymbol{c}}, \boldsymbol{x}, [\boldsymbol{O}]_1, [\boldsymbol{L}]_1 - [\hat{\boldsymbol{L}}], [\boldsymbol{R}]_2 - [\hat{\boldsymbol{R}}]_2)$ is a successful attack against the soundness of the argument for affine constraints.

For every successful adversary $\mathcal{A}$ at least one of the adversaries $\mathcal{B}_1, \mathcal{B}_2$ does not abort. This is because if the statement is false there must be some point in the "chain" where either $[L_i]_1, [R_i]_2$ are honestly computed but $[O_i]_1$ not, or $[O_i]_1$ is honestly computed but $[L_{i+1}]$ or $[R_{i+1}]$ not.

6.5 Efficiency

The proof size is $(3d+2)|\mathbb{G}_1| + (d+2)|\mathbb{G}_2|$ and naive verification requires to compute $3d$ pairings for the quadratic relations and $2(n_0 + 3d + 4)$ for the linear part. Using the "bilinear batching" techniques of Herold et al. [28] the number of pairings can be reduced to $n_0 + 3d + 4$ for the linear part. Since the input is known in $\mathbb{Z}_p$, n_0 pairings in this part can be replaced by n_0 exponentiations in $\mathbb{G}_T$. Finally, using standard batching techniques [14], the number of pairings for the quadratic part can be reduced to $d+2$. As a result the total number of pairings required for verification is $4d+6$, plus n_0 exponentiations in $\mathbb{G}_T$ and $O(n_0+d)$ exponentiations in the source group.

6.6 Adding Zero-Knowledge

In this section we argue how to add zero-knowledge to the argument for correct arithmetic circuit evaluation of Sect. 6. The same discussion applies for the argument for boolean circuit satisfiability discussed in App. C.5 for boolean circuits.

We have to distinguish two different situations. In the first one the input is public, and we can easily modify our proof so that it reveals nothing about the internal evaluation steps.

In most applications however, the input or part of the input must be secret. To deal with this second situation, the circuit input cannot be part of the verifier's input, at least not in the clear. A natural idea is to let the prover commit to it. The problem is that our "knowledge transfer" idea requires the reduction in the soundness proof to know this secret input, which means that the commitment to the input must be extractable so that we can efficiently extend it a vector of correct evaluations $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$. Even in a QANIZK setting where we can efficiently open the commitments, they are only F-extractable [12] (under falsifiable assumptions), which means that we can only extract in the source groups but not in $\mathbb{Z}_p$. This leaves us only with a couple of solutions, all of them unsatisfactory.

One of them is to commit to inputs bitwise and prove that this is done correctly. This is not acceptable in terms of concrete efficiency for arithmetic circuits, but it is a practical approach for boolean circuits.

The second one is to use a commitment to the input which is extractable under knowledge assumptions. Of course, then our construction is no longer secure under falsifiable assumptions. It is interesting in the sense that it indicates a tradeoff in SNARK construction: longer proof size and verification costs ($\Theta(d)$ group elements/ pairings, respectively) but shorter common reference string ($\Theta(\max_{i=0,\ldots,d} n_i)$). $\sum n_i \leq n$.

The last option is to use a proof of knowledge of the input in the random oracle model. This again deviates from our original purpose of constructing proofs under falsifiable assumptions. Using the recent results of [7,9], we can get such a proof of size $\Theta(\sqrt{n_0})$ with a constant number of rounds or $\Theta(\log n_0)$ with a logarithmic number of rounds. In this case, it still interesting because this might improve (although it depends on the circuit) the size of the total proof, from $\log n$ to $\log n_0 + d$, athough at the price of moving to a pairing group.

In any case, we leave for future work to explore the possibilities of this mixed approaches. We now give the technical details on how to add zero-knowledge to our argument for correct circuit evaluation, distinguishing the two aforementioned situations.

Adding Zero-knowledge to Correct Evaluation of Middle Wires. This step is straightforward. The argument is changed so that $[\boldsymbol{L}]_1, [\boldsymbol{R}]_2, [\boldsymbol{O}]_1$ are not given in the clear, but instead the prover gives GS commitments [27] to each of its components. For the quadratic argument, it gives a GS Proof that the verification equation is satisfied, that is, for each i it proves in zk that the pairing product equation:

$$e([L_i]_1, [R_i]_2) - e([O_i]_1, [1]_2) = e([H_i]_1, [T]_2)$$

is satisfied, where $[L_i]_1, [R_i]_2, [O_i]_1, [H_i]_1$ are hidden committed values.

For the linear argument, it suffices to give a GS proof of satisfiability of the the verification equation in Fig. 3. In its most efficient instantiation, the verification equation in Fig. 3 consists of 2 pairing product equations and hence the GS proof consists of 8 elements of each group.

An alternative, more efficient approach (which requires only $2|\mathbb{G}_1| + 2|\mathbb{G}_2|$ group elements) for the linear argument proves that the vectors of committed elements are in a certain linear (bilateral) space. The idea is quite simple but a little cumbersome, so we explain in Sect. 6.7.

Adding this zero-knowledge layer in the intermediate wires is not too costly. The total size of the proof is $4d|\mathbb{G}_1| + 2d|\mathbb{G}_2|$ for the commitments to the wires, $4d|\mathbb{G}_1| + 4d|\mathbb{G}_2|$ for the GS proofs of quadratic equation,

$2|\mathbb{G}_1|+2|\mathbb{G}_2|$ for the linear constraints part. Verification requires 26 pairings for each GS verification equation and $2(n_0+3d+4)$ for the linear proof. First, one can observe that in fact since the input is known in $\mathbb{Z}_p$, the n_0 pairings can be replaced by exponentiations in $\mathbb{G}_T$. Second, using the "bilinear batching" techniques of [28] this is reduced to $7d+3d+4$. Finally, using traditional batching techniques [14], the cost of verifying all the to GS equations can be reduced to $d+6$, resulting in a total cost of $4d+10$ pairings (and $O(n_0+d)$ exponentiations).

Hiding the input and output. Finally, we discuss how to use our results in a scenario where not only the middle values of the wires should be hidden but also the input and the output. In this case the prover commits to the input $\boldsymbol{x}$ using an extractable commitment (using one of the options described above). For instance, $\boldsymbol{c_x}$ can be just the concatenation of GS commitments to the inputs provided the prover submits also a proof of knowledge of their opening (giving additional bitwise commitments and a proof that $\boldsymbol{c_x}$ is of the right form, a proof of knowledge in the ROM) or a commitment of knowledge under extractable assumptions). In all these cases, $\boldsymbol{c_x}$ can be written as $[\boldsymbol{c_x}]_1 = [\mathbf{E}]_1\boldsymbol{x} + [\mathbf{V}]_1\boldsymbol{s}$ (or, if it has components in both source groups in $\mathbb{G}_1, \mathbb{G}_2$ in a similar way except that the matrices $\mathbf{E}$ and $\mathbf{V}$ will also have component in different groups).

The only difference in this case is that in the first n_0 rows of $\mathbf{M}$ we replace the identity matrix by the matrix $\mathbf{E}$ and we add columns of the form $\begin{pmatrix}\mathbf{E}, \mathbf{0}\end{pmatrix}$.

The output is never given in the clear but the commitment to $[\boldsymbol{O}_d]_1$ is a perfectly binding commitment to it.

6.7 Zero Knowledge Argument of Linear Knowledge Transfer

Given $[\mathbf{M}]_1, [\mathbf{N}]_1, [\mathbf{P}]_2$ it is straightforward to find matrices $[\widetilde{\mathbf{M}}]_1, [\widetilde{\mathbf{N}}]_1, [\widetilde{\mathbf{P}}]_2$ such that

$$\begin{pmatrix} \boldsymbol{x} \\ [\boldsymbol{O}]_1 \\ [\boldsymbol{L}]_1 - [\hat{\boldsymbol{L}}]_1 \\ [\boldsymbol{R}]_2 - [\hat{\boldsymbol{R}}]_2 \end{pmatrix} \in \mathbf{Im}\left(\begin{pmatrix} [\mathbf{M}]_1 \\ [\mathbf{N}]_1 \\ [\mathbf{P}]_2 \end{pmatrix}\right) \Longleftrightarrow \begin{pmatrix} \boldsymbol{x} \\ [\boldsymbol{c_O}]_1 \\ [\boldsymbol{c_L}]_1 - [\boldsymbol{c_{\hat{L}}}]_1 \\ [\boldsymbol{c_R}]_2 - [\boldsymbol{c_{\hat{R}}}]_2 \end{pmatrix} \in \mathbf{Im}\left(\begin{pmatrix} [\widetilde{\mathbf{M}}]_1 \\ [\widetilde{\mathbf{N}}]_1 \\ [\widetilde{\mathbf{P}}]_2 \end{pmatrix}\right), \tag{17}$$

where $[\boldsymbol{c}_{\hat{\boldsymbol{L}}}^\top]_1 = [(\hat{L}_1, 0, \hat{L}_2, 0, ..., \hat{L}_d, 0)]_1$ and $[\boldsymbol{c}_{\hat{\boldsymbol{R}}}^\top]_2 = [(\hat{R}_1, 0, \hat{R}_2, 0, ..., \hat{R}_d, 0)]_2$ are commitments (with 0 randomness) to the public constants and $\boldsymbol{c_W}$, for $\boldsymbol{W} \in \{\boldsymbol{L}, \boldsymbol{R}, \boldsymbol{O}\}$, is the vector of commitments to $\boldsymbol{W}$.

For instance,

$$\widetilde{\mathbf{M}} = \begin{pmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Lambda}_1 & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{U} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{\Lambda}_2 & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{U} & \dots & \mathbf{0} \\ \vdots & \vdots & \vdots & \dots & \vdots & & & & & \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \boldsymbol{\Lambda}_{d-1} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{U} \end{pmatrix}$$

where $\mathbf{U} = \begin{pmatrix}\boldsymbol{u}_1, \boldsymbol{u}_2\end{pmatrix}$ is the matrix whose columns are the commitment keys to elements of $\mathbb{G}_1$ in the SXDH instantiation of GS proofs. If $\boldsymbol{s}_i \in \mathbb{Z}_p^2$ is the randomness of the GS commitment to O_i, obviously,

$$[\boldsymbol{c_O}]_1 = [\widetilde{\mathbf{M}}]_1 \begin{pmatrix} \boldsymbol{z} \\ \boldsymbol{s}_1 \\ \vdots \\ \boldsymbol{s}_d \end{pmatrix}.$$

Similar matrices $\widetilde{\mathbf{N}}, \widetilde{\mathbf{P}}$ can be derived from $\mathbf{N}, \mathbf{P}$ and the commitment key so that equation (17) holds.

References

1. B. Abdolmaleki, K. Baghery, H. Lipmaa, and M. Zajac. A subversion-resistant SNARK. In T. Takagi and T. Peyrin, editors, *ASIACRYPT 2017, Part III*, volume 10626 of *LNCS*, pages 3–33. Springer, Heidelberg, Dec. 2017. 3
2. B. Abdolmaleki, H. Lipmaa, J. Siim, and M. Zajac. On QA-NIZK in the BPK model. *IACR Cryptology ePrint Archive*, 2018:877, 2018. 15
3. M. Bellare, G. Fuchsbauer, and A. Scafuro. NIZKs with an untrusted CRS: Security in the face of parameter subversion. In J. H. Cheon and T. Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 777–804. Springer, Heidelberg, Dec. 2016. 3
4. E. Ben-Sasson, A. Chiesa, D. Genkin, E. Tromer, and M. Virza. SNARKs for C: Verifying program executions succinctly and in zero knowledge. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 90–108. Springer, Heidelberg, Aug. 2013. 1
5. E. Ben-Sasson, A. Chiesa, M. Green, E. Tromer, and M. Virza. Secure sampling of public parameters for succinct zero knowledge proofs. In *2015 IEEE Symposium on Security and Privacy*, pages 287–304. IEEE Computer Society Press, May 2015. 1, 3
6. E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza. Scalable zero knowledge via cycles of elliptic curves. In J. A. Garay and R. Gennaro, editors, *CRYPTO 2014, Part II*, volume 8617 of *LNCS*, pages 276–294. Springer, Heidelberg, Aug. 2014. 1, 3, 13
7. J. Bootle, A. Cerulli, P. Chaidos, J. Groth, and C. Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In M. Fischlin and J.-S. Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 327–357. Springer, Heidelberg, May 2016. 2, 4, 11, 28
8. S. Bowe, A. Gabizon, and M. D. Green. A multi-party protocol for constructing the public parameters of the pinocchio zk-SNARK. Cryptology ePrint Archive, Report 2017/602, 2017. `http://eprint.iacr.org/2017/602`. 1
9. B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy*, pages 315–334. IEEE Computer Society Press, May 2018. 1, 3, 4, 28
10. G. Danezis, C. Fournet, J. Groth, and M. Kohlweiss. Square span programs with applications to succinct NIZK arguments. In P. Sarkar and T. Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 532–550. Springer, Heidelberg, Dec. 2014. 34, 39, 40
11. V. Daza, , A. González, Z. Pindado, C. Ràfols, and J. Silva. Shorter quadratic qanizk proofs. In *in Public Key Cryptography'19, to appear.* IACR, 2019. 40
12. A. Escala and J. Groth. Fine-tuning Groth-Sahai proofs. In H. Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 630–649. Springer, Heidelberg, Mar. 2014. 28, 32
13. P. Fauzi, H. Lipmaa, J. Siim, and M. Zajac. An efficient pairing-based shuffle argument. In T. Takagi and T. Peyrin, editors, *ASIACRYPT 2017, Part II*, volume 10625 of *LNCS*, pages 97–127. Springer, Heidelberg, Dec. 2017. 2, 3, 19
14. A. L. Ferrara, M. Green, S. Hohenberger, and M. Ø. Pedersen. Practical short signature batch verification. In M. Fischlin, editor, *CT-RSA 2009*, volume 5473 of *LNCS*, pages 309–324. Springer, Heidelberg, Apr. 2009. 28, 29
15. G. Fuchsbauer. Subversion-zero-knowledge SNARKs. In M. Abdalla and R. Dahab, editors, *PKC 2018, Part I*, volume 10769 of *LNCS*, pages 315–347. Springer, Heidelberg, Mar. 2018. 3
16. R. Gennaro, C. Gentry, B. Parno, and M. Raykova. Quadratic span programs and succinct NIZKs without PCPs. In T. Johansson and P. Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 626–645. Springer, Heidelberg, May 2013. 1, 2, 6, 11, 40
17. C. Gentry and D. Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In L. Fortnow and S. P. Vadhan, editors, *43rd ACM STOC*, pages 99–108. ACM Press, June 2011. 2
18. E. Ghadafi and J. Groth. Towards a classification of non-interactive computational assumptions in cyclic groups. In T. Takagi and T. Peyrin, editors, *ASIACRYPT 2017, Part II*, volume 10625 of *LNCS*, pages 66–96. Springer, Heidelberg, Dec. 2017. 10
19. S. Goldwasser, Y. T. Kalai, and G. N. Rothblum. Delegating computation: interactive proofs for muggles. In R. E. Ladner and C. Dwork, editors, *40th ACM STOC*, pages 113–122. ACM Press, May 2008. 6
20. A. González, A. Hevia, and C. Ràfols. QA-NIZK arguments in asymmetric groups: New tools and new constructions. In T. Iwata and J. H. Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 605–629. Springer, Heidelberg, Nov. / Dec. 2015. 2, 4, 5, 8, 24, 25, 40

21. J. Groth. Short non-interactive zero-knowledge proofs. In M. Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 341–358. Springer, Heidelberg, Dec. 2010. 3
22. J. Groth. Short pairing-based non-interactive zero-knowledge arguments. In M. Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 321–340. Springer, Heidelberg, Dec. 2010. 1
23. J. Groth. On the size of pairing-based non-interactive arguments. In M. Fischlin and J.-S. Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Heidelberg, May 2016. 1, 3, 6, 7, 13
24. J. Groth, M. Kohlweiss, M. Maller, S. Meiklejohn, and I. Miers. Updatable and universal common reference strings with applications to zk-SNARKs. In H. Shacham and A. Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 698–728. Springer, Heidelberg, Aug. 2018. 1, 3, 7, 8, 10, 13, 15, 17, 18
25. J. Groth, R. Ostrovsky, and A. Sahai. Perfect non-interactive zero knowledge for NP. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 339–358. Springer, Heidelberg, May / June 2006. 1, 5
26. J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. In N. P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, Heidelberg, Apr. 2008. 1, 5, 32
27. J. Groth and A. Sahai. Efficient noninteractive proof systems for bilinear groups. *SIAM J. Comput.*, 41(5):1193–1232, 2012. 28
28. G. Herold, M. Hoffmann, M. Klooß, C. Ràfols, and A. Rupp. New techniques for structural batch verification in bilinear groups with applications to groth-sahai proofs. In B. M. Thuraisingham, D. Evans, T. Malkin, and D. Xu, editors, *ACM CCS 17*, pages 1547–1564. ACM Press, Oct. / Nov. 2017. 27, 29
29. C. S. Jutla and A. Roy. Shorter quasi-adaptive NIZK proofs for linear subspaces. In K. Sako and P. Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 1–20. Springer, Heidelberg, Dec. 2013. 2
30. C. S. Jutla and A. Roy. Switching lemma for bilinear tests and constant-size NIZK proofs for linear subspaces. In J. A. Garay and R. Gennaro, editors, *CRYPTO 2014, Part II*, volume 8617 of *LNCS*, pages 295–312. Springer, Heidelberg, Aug. 2014. 1, 2
31. Y. Kalai, O. Paneth, and L. Yang. On publicly verifiable delegation from standard assumptions. Cryptology ePrint Archive, Report 2018/776, 2018. `https://eprint.iacr.org/2018/776`. 6
32. Y. T. Kalai, R. Raz, and R. D. Rothblum. How to delegate computations: the power of no-signaling proofs. In D. B. Shmoys, editor, *46th ACM STOC*, pages 485–494. ACM Press, May / June 2014. 6
33. J. Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *24th ACM STOC*, pages 723–732. ACM Press, May 1992. 1
34. E. Kiltz and H. Wee. Quasi-adaptive NIZK for linear subspaces revisited. In E. Oswald and M. Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 101–128. Springer, Heidelberg, Apr. 2015. 2, 25
35. B. Libert, T. Peters, M. Joye, and M. Yung. Linearly homomorphic structure-preserving signatures and their applications. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 289–307. Springer, Heidelberg, Aug. 2013. 2
36. M. Maller, M. Kohlweiss, S. Bowe, and S. Meiklejohn. Sonic: Zero-knowledge snarks from linear-size universal and updateable structured reference string. Cryptology ePrint Archive, Report 2019/099, 2019. `http://eprint.iacr.org/2019/099`. 2, 3, 4, 6, 8
37. S. Micali. The knowledge complexity of interactive proofs. In *SIAM Journal on Computing 30 (4)*, pages 1253–1298, 2000. 1
38. P. Morillo, C. Ràfols, and J. L. Villar. The kernel matrix Diffie-Hellman assumption. In J. H. Cheon and T. Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 729–758. Springer, Heidelberg, Dec. 2016. 4, 8
39. O. Paneth and G. N. Rothblum. On zero-testable homomorphic encryption and publicly verifiable non-interactive arguments. In Y. Kalai and L. Reyzin, editors, *TCC 2017, Part II*, volume 10678 of *LNCS*, pages 283–315. Springer, Heidelberg, Nov. 2017. 6
40. B. Parno, J. Howell, C. Gentry, and M. Raykova. Pinocchio: Nearly practical verifiable computation. In *2013 IEEE Symposium on Security and Privacy*, pages 238–252. IEEE Computer Society Press, May 2013. 1, 3, 6, 13

A Security Definitions

A.1 Dual-mode commitments and Groth-Sahai proofs [26].

Groth-Sahai proofs allow to prove satisfiability of quadratic equations in bilinear groups in the non-interactive setting. More precisely, Groth-Sahai proofs deal with equations of the form

$$\sum_{j=1}^{m_y} a_j \mathsf{y}_j + \sum_{i=1}^{m_x} b_i \mathsf{x}_i + \sum_{i,j=1}^{m_x,m_y} \gamma_{i,j} \mathsf{x}_i \mathsf{y}_j = t,$$

in which the set of variables is divided into two disjoint subsets $\mathsf{X} = \{\mathsf{x}_1, \dots, \mathsf{x}_{m_x}\}$ and $\mathsf{Y} = \{\mathsf{y}_1, \dots, \mathsf{y}_{m_y}\}$, and depending on the type of equation $\mathsf{X}, \mathsf{Y} \subset \mathbb{Z}_p$ (quadratic equations in $\mathbb{Z}_p$), $\mathsf{X} \subset \mathbb{Z}_p, \mathsf{Y} \subset \mathbb{G}_\gamma$ (multi-exponentiation equations in $\mathbb{G}_\gamma$) for $\gamma \in \{1, 2\}$ or $\mathsf{X} \subset \mathbb{G}_1$ and $\mathsf{Y} \subset \mathbb{G}_2$ (pairing product equations).

The scheme can be seen as a commit-and-prove scheme [12], where in the first step the prover gives commitments to the solutions, and in the second provides a proof that these commitments verify the corresponding equation. In particular, the commitments used are *dual-mode commitments*, that is, commitments that can be either perfectly binding or perfectly hiding, and we can move from one to the other with an indistinguishable change of security game. More precisely, Groth-Sahai commitments to field elements $z \in \mathbb{Z}_p$ and group elements $[z]_s \in \mathbb{G}$ are, respectively:

$$\mathsf{Com}(z; w) = z\,[\boldsymbol{u}]_s + w[\boldsymbol{u}_1]_s, \qquad \mathsf{Com}([z]_s; w_1, w_2) = \begin{bmatrix} 0 \\ z \end{bmatrix}_s + w_1[\boldsymbol{u}_1]_s + w_2[\boldsymbol{u}_2]_s,$$

where $[\boldsymbol{u}]_s, [\boldsymbol{u}_1]_s, [\boldsymbol{u}_2]_s$ are vectors in $\mathbb{G}^2$ given in the commitment key, and their definitions depend on whether we want the commitments to be perfectly binding or perfectly hiding.

Groth-Sahai proofs are sound, witness-indistinguishable and, in many cases, zero-knowledge. More precisely, the proof is always zero-knowledge for quadratic equations in $\mathbb{Z}_p$ and multi-exponentiation equations, and also for pairing product equations provided that $t = 1$.

A.2 Quasi-Adaptive NIZK proofs

QANIZK proofs consider a language defined by a relation $\mathcal{R}_\rho$, which in turn is completely determined by some parameter ρ sampled from a distribution $\mathcal{D}_{gk}$. We say that $\mathcal{D}_{gk}$ is *witness samplable* if there exists an efficient algorithm that samples (ρ, ω) from a distribution $\mathcal{D}_{gk}^{\mathsf{par}}$ such that ρ is distributed according to $\mathcal{D}_{gk}$, and membership of ρ in the *parameter language* $\mathcal{L}_{\mathsf{par}}$ can be efficiently verified with ω. While the Common Reference String can be set based on ρ, the zero-knowledge simulator is required to be a single probabilistic polynomial time algorithm that works for the whole collection of relations $\mathcal{R}_{gk}$.

A tuple of algorithms $(\mathsf{K}_0, \mathsf{K}_1, \mathsf{P}, \mathsf{V})$ is called a QA-NIZK proof system for witness-relations $\mathcal{R}_{gk} = \{\mathcal{R}_\rho\}_{\rho \in \sup(\mathcal{D}_{gk})}$ with parameters sampled from a distribution $\mathcal{D}_{gk}$ over associated parameter language $\mathcal{L}_{\mathsf{par}}$, if there exists a probabilistic polynomial time simulator $(\mathsf{S}_1, \mathsf{S}_2)$, such that for all non-uniform PPT adversaries $\mathcal{A}_1$, $\mathcal{A}_2$, $\mathcal{A}_3$ we have:

Quasi-Adaptive Completeness:

$$\Pr\begin{bmatrix} gk \leftarrow \mathsf{K}_0(1^\lambda); \rho \leftarrow \mathcal{D}_{gk}; \psi \leftarrow \mathsf{K}_1(gk, \rho); (x, w) \leftarrow \mathcal{A}_1(gk, \psi); \\ \pi \leftarrow \mathsf{P}(\psi, x, w) : \mathsf{V}(\psi, x, \pi) = 1 \text{ if } \mathcal{R}_\rho(x, w) \end{bmatrix} = 1.$$

Computational Quasi-Adaptive Soundness:

$$\Pr\begin{bmatrix} gk \leftarrow \mathsf{K}_0(1^\lambda); \rho \leftarrow \mathcal{D}_{gk}; \psi \leftarrow \mathsf{K}_1(gk, \rho); \\ (x, \pi) \leftarrow \mathcal{A}_2(gk, \psi) : \mathsf{V}(\psi, x, \pi) = 1 \text{ and } \neg(\exists w : \mathcal{R}_\rho(x, w)) \end{bmatrix} \approx 0.$$

Perfect Quasi-Adaptive Zero-Knowledge:

$$\Pr[gk \leftarrow \mathsf{K}_0(1^\lambda); \rho \leftarrow \mathcal{D}_{gk}; \psi \leftarrow \mathsf{K}_1(gk,\rho) : \mathcal{A}_3^{\mathsf{P}(\psi,\cdot,\cdot)}(gk,\psi) = 1] =$$
$$\Pr[gk \leftarrow \mathsf{K}_0(1^\lambda); \rho \leftarrow \mathcal{D}_{gk}; (\psi,\tau) \leftarrow \mathsf{S}_1(gk,\rho) : \mathcal{A}_3^{\mathsf{S}(\psi,\tau,\cdot,\cdot)}(gk,\psi) = 1]$$

where

- $\mathsf{P}(\psi,\cdot,\cdot)$ emulates the actual prover. It takes input (x,w) and outputs a proof π if $(x,w) \in \mathcal{R}_\rho$. Otherwise, it outputs $\perp$.
- $\mathsf{S}(\psi,\tau,\cdot,\cdot)$ is an oracle that takes input (x,w). It outputs a simulated proof $\mathsf{S}_2(\psi,\tau,x)$ if $(x,w) \in \mathcal{R}_\rho$ and $\perp$ if $(x,w) \notin \mathcal{R}_\rho$.

Note that ψ is the CRS in the above definitions. We assume that ψ contains an encoding of ρ, which is thus available to V.

In this work algorithm K_0 always samples the group key for an asymmetric bilinear group. For this reason we will always omit K_0.

B An Example

We illustrate how our encoding for circuit satisfiability which divides the linear constraints into different levels works.

Example 1. $\phi : \mathbb{Z}_p^4 \to \mathbb{Z}_p$, $\phi(x_1,x_2,x_3,x_4) = (((x_1+2x_2)(x_3+x_4))(3+4x_2))((x_2+x_4)x_1)$. If we set $C_{0,j} = x_j$, $j = 1,2,3,4$, then $C(x_1,x_2,x_3,x_4) = c$ and $C_{i,j}$ is a valid assignment of the jth multiplication gate at level i if and only if the following equations are satisfied:

- Level 1: • $C_{1,1} = A_{1,1}B_{1,1}$ $A_{1,1} = (C_{0,1} + 2C_{0,2})$, $B_{1,1} = (C_{0,3} + C_{0,4})$ • $C_{1,2} = A_{1,2}B_{1,2}$, $A_{1,2} = (C_{0,2} + C_{0,4})$, $B_{1,1} = C_{0,1}$.
- Level 2: • $C_{2,1} = A_{2,1}B_{2,1}$, $A_{2,1} = C_{1,1}$ $B_{2,1} = (3 + 4C_{0,2})$.
- Level 3: • $C_{3,1} = A_{3,1}B_{3,1}$, $A_{3,1} = C_{2,1}$, $B_{3,1} = C_{1,2}$.
- Correct output: $C_{3,1} = y$.

The Lagrangian Pedersen commitments for each level and each side are defined as:

- Level 1: • $L_1 = (C_{0,1} + 2C_{0,2})\lambda_1 + (C_{0,2} + C_{0,4})\lambda_2$ • $R_1 = (C_{0,3} + C_{0,4})\lambda_1 + C_{0,1}\lambda_2$.
- Level 2: • $L_2 = C_{1,1}\lambda_1$ • $R_2 = 4C_{0,2}\lambda_1$.
- Level 3: •$L_3 = C_{2,1}\lambda_1$ • $R_3 = C_{1,2}\lambda_1$,

and the affine term $(\hat{L}_1, \hat{R}_1, \hat{L}_2, \hat{R}_2, \hat{L}_3, \hat{R}_3) = (0,0,0,3\lambda_1,0,0)$. In matrix form,

$$\begin{pmatrix} c_{0,1} \\ c_{0,2} \\ c_{0,3} \\ c_{0,4} \\ L_1 \\ R_1 \\ L_2 \\ R_2 \\ L_3 \\ R_3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ \lambda_1 & 2\lambda_1 & 0 & \lambda_2 & 0 & 0 & 0 \\ \lambda_2 & 0 & \lambda_1 & \lambda_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \lambda_1 & 0 & 0 \\ 0 & 4\lambda_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \lambda_1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \lambda_1 \end{pmatrix} \begin{pmatrix} c_{0,1} \\ c_{0,2} \\ c_{0,3} \\ c_{0,4} \\ c_{1,1} \\ c_{1,2} \\ c_{2,1} \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 3\lambda_1 \\ 0 \end{pmatrix}$$

C Boolean Circuits

For boolean circuits, we let $\mathcal{R}$ be a relation generator which on input 1^λ outputs a family of polynomial time decidable relations $R_\Phi = \{R_\phi : \ \phi \in \Phi\}$, where,

- for some N, Φ is the set of all boolean circuits such that the total number of inputs plus gates is at most N,
- $R_\phi = \{(\phi, (\boldsymbol{x}_{pub}, \boldsymbol{x}_{sec}), \boldsymbol{y}) : \phi(\boldsymbol{x}_{pub}, \boldsymbol{x}_{sec}) = \boldsymbol{y}\}$.

Further, the associated language is $\mathcal{L}_\phi = \{(\phi, \boldsymbol{x}_{pub}, \boldsymbol{y}) : \exists \boldsymbol{x}_{sec}, (\boldsymbol{x}_{pub}, \boldsymbol{x}_{sec}, \boldsymbol{y}) \in \mathcal{R}_\phi\}$. The circuit ϕ computes a function $\{0,1\}^n \to \{0,1\}^{n'}$ for some $n, n' \in \mathbb{N}$. The gates of ϕ are arbitrary fan-in two gates. Again, the statement is $u = (\phi, \boldsymbol{x}_{pub}, \boldsymbol{y})$ and the witness is $w = (\boldsymbol{x}_{sec})$ or any information which can be efficiently computed from (u, w).

C.1 Preliminaries on Boolean Circuits

We extend our results to any boolean circuit $\phi : \{0,1\}^n \to \{0,1\}^{n'}$ with m multiplication gates and where all the gates have fan-in two. The gates of ϕ can be of any type (excluding non-interesting or trivial gate types).

We give two constructions which have different performance. The first one characterizes boolean gates as quadratic operations on the input. The second one uses the gate linearization approach of [10], which observe that if the left, right and output wire a, b, c of a certain gate are boolean, then correct gate evaluation can be reduced to proving that a linear combination of the three wires is in $\{0, 2\}$. Note that despite of the name, "gate linearization" is still a quadratic condition, as $x \in \{0, 2\}$ is equivalent to saying that x is the solution to a quadratic equation. The first characterization is closer to a a Quadratic Span Program, while the second one is closer to Square Span programs.

We list below the 10 gate types allowed for the circuit ϕ, along with both characterizations. The list of gates is taken from [10], which observe that the last remaining 6 gate types depend mostly on one input and are not used often.

- $\mathsf{AND}(a,b,c)$: 1) $ab = c$, 2) $a + b - 2c \in \{0,1\}$.
- $\mathsf{NAND}(a,b,c)$: 1) $1 - ab = c$, 2) $a + b - 2(1 - c) \in \{0,1\}$.
- $\mathsf{OR}(a,b,c)$: 1) $1 - (1 - a)(1 - b) = c$, 2) $(1 - a) + (1 - b) - 2(1 - c) \in \{0,1\}$.
- $\mathsf{NOR}(a,b,c)$: 1) $(1 - a)(1 - b) = c$, 2) $(1 - a) + (1 - b) - 2c \in \{0,1\}$.
- $\mathsf{XOR}(a,b,c)$: 1) $b(1 - a) + a(1 - b) = c$, 2) $a + b + c \in \{0,2\}$.
- $\mathsf{XNOR}(a,b,c)$: 1) $1 - a(1 - b) - b(1 - a) = c$, 2) $a + b + (1 - c) \in \{0,2\}$.
- $\mathsf{G}_1(a,b,c) = (c = \overline{a} \wedge b)$: 1) $(1 - a)b = c$, 2) $(1 - a) + b - 2c \in \{0,1\}$.
- $\mathsf{G}_2(a,b,c) = (c = \overline{\overline{a} \wedge b})$: 1) $1 - (1 - a)b = c$, 2) $(1 - a) + b - 2(1 - c) \in \{0,1\}$.
- $\mathsf{G}_3(a,b,c) = (c = a \wedge \overline{b})$: 1) $a(1 - b) = c$, 2) $a + (1 - b) - 2c \in \{0,1\}$.
- $\mathsf{G}_4(a,b,c) = (c = \overline{a \wedge \overline{b}})$: 1)$1 - a(1 - b) = c$, 2) $a + (1 - b) - 2(1 - c) \in \{0,1\}$.

From each one of the two characterizations, we derive a lemma which expresses boolean circuit satisfiability in terms of different sets of equations. The first one uses the expression of boolean gates in terms of quadratic equations. The constants $f_{i,j}, g_{i,j}$ establish consistency of the wires and the constants $\alpha_i, \beta_i, \gamma_i$ depend on the type of gate and make sure that the right gate is evaluated.

Lemma 11. *Let $\phi : \{0,1\}^n \to \{0,1\}^{n'}$, be a circuit with m boolean gates. There exist*

a) variables A_i, B_i, C_i, $i = 1, \ldots, n + m$,
b) constants $f_{i,j}, g_{i,j} \in \{0,1\}$, $i = 1, \ldots, n + m$, $j = 1, \ldots, n + m$,
c) constants $\beta_i, \gamma_i, \epsilon_i \in \{0,1\}$, $i = n + 1, \ldots, n + m$,

such that, for every $(x_1, \ldots, x_n) \in \mathbb{Z}_p^n$, if we set $C_j = x_j$, for all $j = 1, \ldots, n$, then $\phi(\boldsymbol{x}) = \boldsymbol{y}$ and A_{n+i}, B_{n+i} and C_{n+i} are evaluated to the left, right and output of the ith gate if and only if the following equations are satisfied:

1. *(Boolean input) For each $i = 1, \dots, n$,*
$$A_i B_i - C_i = 0. \tag{18}$$
2. *(Correct Gate Evaluation) For each $i = n+1, \dots, n+m$,*
$$C_i = A_i B_i + \beta_i A_i + \gamma_i B_i + \epsilon_i. \tag{19}$$
3. *(Affine constraints) For each $i = 1, \dots, n+m$,*
$$A_i = \sum_{j=1}^{n+m} f_{i,j} C_j \qquad\qquad B_i = \sum_{j=1}^{n+m} g_{i,j} C_j.$$
4. *(Correct Output) For all $j = 1, \dots, n'$, $C_{n+m-n'+j} = y_j$.*

Proof. For $i = 1, \dots, n$, the constraints $f_{i,j}, g_{i,j}$ should be defined as 0 if $i \neq j$ and 1 otherwise. Then, for any $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$ satisfying the constraints, $a_i = c_i$ and $b_i = c_i$, and the equation expresses the fact that the input is boolean, as $a_i b_i - c_i = c_i^2 - c_i$ is satisfied if and only if $c_i \in \{0, 1\}$.

On the other hand, the circuit ϕ specifies, for the *ith* circuit gate, a pair of indexes j_L, j_R which indicate the left and right wire, and also a type of gate. That is, from the quadratic expression for boolean circuit satisfiability, correct evaluation of this gate is expressed as:

$$C_{n+i} = C_{j_L} C_{j_R} \alpha_i + C_{j_L} \beta_i + C_{j_R} \hat{\gamma}_i + \epsilon_i,$$

for some constants $\alpha, \beta_i, \hat{\gamma}_i, \epsilon_i$ which depend on the gate type and such that $\alpha_i \in \{\pm 1\}$. This can be rewritten as:

$$C_{n+i} = C_{j_L}(C_{j_R}\alpha_i) + C_{j_L}\beta_i + (C_{j_R}\alpha_i)(\alpha_i \hat{\gamma}_i) + \epsilon_i, \tag{20}$$

where we use the fact that $\alpha_i^2 = \alpha_i$. For $i = n+1, \dots, n+m$, we define the constant $f_{i,j}$ and $g_{i,j}$ to be 0 everywhere except for $f_{i,j_L} = 1$ and $g_{i,j_R} = \alpha_i$. Therefore, $A_i = \sum_{j=1}^{n+m} f_{i,j} C_j = C_{j_L}$, $B_i = \sum_{j=1}^{n+m} g_{i,j} C_j = C_{j_R} \alpha_i$ and equation 21 can be rewritten as:

$$C_i = A_i B_i + A_i \beta_i + B_i \gamma_i + \epsilon_i, \tag{21}$$

where $\gamma_i = \alpha_i \hat{\gamma}_i$. Obviously, this implies that if $C_j = x_j$, and the linear constraints are satisfied, then the rest of the output wires are also consistent with this definition. Together with the conditions which guarantee that the input is boolean, we conclude that, for $j = n+m-n'+1, \dots, n+m$, C_j is the output corresponding to this input. Therefore, if these values are consistent with $\boldsymbol{y}$, we can conclude that $\phi(\boldsymbol{x}) = \boldsymbol{y}$.

For succintness, we will express all the quadratic equations (boolean input and correct gate evaluation) as a divisibility relation with the usual polynomial aggregation technique.

Lemma 12. *Let $\mathcal{R} \subset \mathbb{Z}_p$ be a set of cardinal $n+m$ and let $\lambda_i(X)$ be the associated Lagrangian polynomials and $t(X)$ the polynomial whose roots are the elements of $\mathcal{R}$. Let $\phi : \{0,1\}^n \to \{0,1\}^{n'}$, be any circuit with m boolean gates. There exist some unique polynomials $u_L(X), u_R(X), u_0(X)$ of degree at most $n+m-1$ which are efficiently computable from the circuit description and such that for any tuple $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}) \in (\mathbb{Z}_p^{n+m})^3$, if*

$$\ell(X) = \sum_{i=1}^{n+m} a_i \lambda_i(X), \qquad r(X) = \sum_{i=1}^{m} b_i \lambda_i(X), \qquad o(X) = \sum_{i=1}^{n+m} c_i \lambda_i(X),$$

it holds that $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$ satisfy equations (18) and (19) if and only if $t(X)$ divides $p(X)$, where

$$p(X) = \ell(X) r(X) + \ell(X) u_L(X) + r(X) u_R(X) + u_0(X) - o(X).$$

Proof. The proof is a direct consequence of Lemma 11. Indeed, for $i = 1, \dots, n$, it suffices to define $u_L(r_i) = 0$, $u_R(r_i) = 0$ and $u_0(r_i) = 0$. Therefore,

$$p(r_i) = \ell(r_i)r(r_i) - o(r_i) = a_i b_i - c_i.$$

On the other hand, for $i = n+1, \dots, n+m$, it suffices to define $u_L(X), u_R(X), u_0(X)$ to take the values $u_L(r_i) = \beta_i$, $u_R(r_i) = \gamma_i$ and $u_0(r_i) = \epsilon_i$. Therefore, $p(r_i) = a_i b_i + a_i\beta_i + b_i\gamma_i + \epsilon_i - c_i$.

These two facts together prove that if equations (18) and (19) are satisfied then $p(X)$ is divisible by $t(X)$, since it is 0 in all of its roots.

Finally, the polynomials $u_L(X), u_R(X), u_0(X)$ can be efficiently computed from the circuit description, as they depend only on n, m and the type of each gate.

We now state a similar lemma which uses the other characterization of correct gate evaluation.

Lemma 13. *Let $\phi : \{0,1\}^n \to \{0,1\}^{n'}$, be a boolean circuit with m gates. There exist*

a) variables C_i, $i = 1, \dots, n+m$
b) variables D_i, $i = 1, \dots, m$,
c) constants $f_i, f_{i,j} \in \mathbb{Z}_p$, $i = 1, \dots, m$, $j = 1, 1, \dots, n+m$

such that, for every $(x_1, \dots, x_n) \in \mathbb{Z}_p^n$, if we set $C_j = 2x_j$, for all $j = 1, \dots, n$, then $\phi(\boldsymbol{x}) = \boldsymbol{y}$ and C_{n+i} is evaluated to the two times the output of the ith gate if and only if the following equations are satisfied:

1. *(Boolean wires) For each $i = 1, \dots, n+m$,*

$$(C_i - 1)^2 = 1.$$

2. *(Correct evaluation of gates) For each $i = 1, \dots, m$,*

$$(D_i - 1)^2 - 1 = 0.$$

3. *(Affine constraints)*

$$D_i = f_i + \sum_{j=1}^{n+m} f_{i,j} C_j.$$

4. *(Correct Output) For all $j = 1, \dots, n'$, $C_{n+m-n'+j} = 2y_j$.*

Proof. The lemma is a direct characterization of gate linearization. Multiplying by two when necessary, write all gate linearization constraints as proving that a linear combination of a, b, c is in $\{0, 2\}$. The constants $f_i, f_{i,j}$ should be defined as encoding the gate linearization constraint. For instance, if the ith gate is a NAND gate which takes as left and right input the wires C_{j_L}, C_{j_R} and outputs C_{j_O}, $D_i = 2C_{j_L} + 2C_{j_R} + 2C_{j_O} - 4$, i.e. $f_i = -4$, $f_{i,j_L} = f_{i,j_R} = f_{i,j_O} = 2$.

C.2 Circuit Slicing for Boolean Circuits

We partition the set of gates $\mathcal{G}$ of a given circuit ϕ into d different subsets $\mathcal{G}_1, \dots, \mathcal{G}_d$. The subset $\mathcal{G}_i$ consists of all the gates at level i, that is, each gate in $\mathcal{G}_i$, takes as input a left wire w_L and a right wire w_R and both wires have been obtained from the input wires by evaluating at most $i-1$ gates and at least one of w_L or w_R is the result of evaluating exactly $i-1$ gates. Let n_i be the cardinal of the gates at level i. We assume gates at each level are ordered in some way and they are denoted as $G_{i,1}, \dots, G_{i,n_i}$.

We encode boolean circuit satisfiability as d sets of equations in a very similar way as we did for the arithmetic case. For each level i, define variables $C_{i,j}$, $j = 1, \dots, n_i$. The equations will be defined so that each of these variables encodes the output of gate j at level i. The gate $G_{i,j}$ will be correctly evaluated if

$$C_{i,j} = G_{i,j}(A_{i,j}, B_{i,j}),$$

where $A_{i,j} = C_{k_L,\ell_L}$ and $B_{i,j} = C_{k_R,\ell_R}$ for some indexes $0 \leq k_L, k_R < i$, $\ell_L \in \{1, \ldots, n_{k_L}\}$ and $\ell_R \in \{1, \ldots, n_{k_R}\}$, which depend on i, j and which are specified by the circuit description. That is, the left wire of $G_{i,j}$ should be evaluated to the output of the ℓ_L gate at level k_L and the right wire to the output of the ℓ_R gate at level k_R. The following lemma is obvious from this discussion and the expression of boolean gates as a quadratic equation.

Lemma 14. *Let $\phi : \{0,1\}^{n_0} \to \{0,1\}^{n_d}$, be a circuit of multiplicative depth d with n_i gates at level i. There exist*

a) variables $C_{i,j}$, $i = 0, \ldots, d$, $j = 1, \ldots, n_i$,
b) variables $A_{i,j}, B_{i,j}$, $i = 1, \ldots, d$, $j = 1, \ldots, n_i$,
c) constants $f_{i,j,k,\ell}, g_{i,j,k,\ell} \in \{0,1\}$, $i = 1, \ldots, d$, $k = 0, \ldots, i-1$, $j = 1, \ldots, n_i$, $\ell = 1, \ldots, n_k$,
d) constants $\beta_{i,j}, \gamma_{i,j}, \epsilon_{i,j}, \delta_{i,j} \in \{0,1\}$, $i = 1, \ldots, d$, $j = 1, \ldots, n_i$, which depend on the gate type:

such that, for every $(x_1, \ldots, x_{n_0}) \in \mathbb{Z}_p^{n_0}$, if we set $C_{0,j} = x_j$, for all $j = 1, \ldots, n_0$, then $\boldsymbol{x} \in \{0,1\}^{n_0}$, $\phi(\boldsymbol{x}) = \boldsymbol{y}$, and $A_{i,j}$, $B_{i,j}$ and $C_{i,j}$ are evaluated to the output of the jth gate at level i if and only if the following equations are satisfied:

1. *(Quadratic constraints). For each $i = 1, \ldots, d$, for all $j = 1, \ldots, n_i$,*

$$C_{i,j} = A_{i,j}B_{i,j} + A_{i,j}\beta_{i,j} + B_{i,j}\gamma_{i,j} + \epsilon_{i,j}, \tag{22}$$

where

$$A_{i,j} = \sum_{k=0}^{i-1}\sum_{\ell=1}^{n_k} f_{i,j,k,\ell}C_{k,\ell} \qquad \textit{and} \tag{23}$$

$$B_{i,j} = \sum_{k=0}^{i-1}\sum_{\ell=1}^{n_k} g_{i,j,k,\ell}C_{k,\ell}. \tag{24}$$

2. *(Correct Output) For all $j = 1, \ldots, n_d$, $C_{d,j} = y_j$.*

We note that for each i, j the constants $f_{i,j,k,\ell}$ are zero everywhere except for $f_{i,j,k_L,\ell_L} = 1$ and $g_{i,j,k,\ell}$ also except for $g_{i,j,k_R,\ell_R} \in \{\pm 1\}$.

C.3 First NIZK Argument in the Generic Group Model

From the first characterization of boolean circuits, given in Lemma 11 and Lemma 12, we can construct a SNARK with the same universal and updatability properties discussed before. This characterization also separates quadratic and linear constraints in such a way that the quadratic constraints are universal.

To ease readability, we omit the algorithms of the scheme which only refer to updates and verification of the common reference string, as well as the description of the proofs of correctness and update trapdoors, which are the same as in the argument for arithmetic circuit satisfiability of Sect. 5.

More formally, we let $\mathcal{R}$ be a relation generator which on input 1^λ outputs a family of polynomial time decidable relations $R_\Phi = \{R_\phi :\ \phi \in \Phi\}$, where,

– for some N, Φ is the set of all boolean circuits such that if the number of inputs is n, the number of gates is m, then $m + n \leq N$,
– $R_\phi = \{(\phi, (\boldsymbol{x}_{pub}, \boldsymbol{x}_{sec}), \boldsymbol{y}) : \phi(\boldsymbol{x}_{pub}, \boldsymbol{x}_{sec}) = \boldsymbol{y}\}$.

Further, the associated language is $\mathcal{L}_\phi = \{(\phi, \boldsymbol{x}_{pub}, \boldsymbol{y}) : \exists \boldsymbol{x}_{sec}, (\boldsymbol{x}_{pub}, \boldsymbol{x}_{sec}, \boldsymbol{y}) \in \mathcal{R}_\phi\}$. The circuit ϕ computes a function $\{0,1\}^n \to \{0,1\}^{n'}$ for some $n, n' \in \mathbb{N}$. The gates of ϕ are arbitrary (non-trivial) fan-in two gates.

$\mathsf{Setup}(R_\Phi)$**:** This algorithm samples $s \leftarrow \mathbb{Z}_p^*$ and publishes

$$\Sigma_\Phi = \left(gk, \{[\lambda_i(s)]_{1,2}\}_{i=1}^N, \{[s^i]_{1,2}\}_{i=1}^N, [t(s)]_{1,2}\right),$$

where $\lambda_i(X)$ are the Lagrangian polynomials associated to some set $\mathcal{R} = \{r_1, \ldots, r_N\} \subset \mathbb{Z}_p$. The simulation trapdoor is $\tau_\Sigma = s$.

$\mathsf{Drv.Setup}(\phi, n_{pub}, \mathit{\Sigma_\Phi})$**:** On input a circuit $\phi : \mathbb{Z}_p^n \to \mathbb{Z}_p^{n'}$ with m multiplication gates, $n_{pub} \leq n$ public inputs and such that $n + m \leq N$ and $\mathit{\Sigma_\Phi}$, this algorithm computes $([\mathbf{Q}]_1, [\boldsymbol{q}_0]_1) \in \mathbb{G}_1^{4 \times n+m+3} \times \mathbb{G}_1^4$, defined as:

$$\mathbf{Q} = \begin{pmatrix} \mathbf{\Lambda}_{pub} & \mathbf{0} & \mathbf{\Lambda}_{opt} & 0 & 0 & 0 \\ \mathbf{\Lambda}_{pub} & \mathbf{\Lambda}_{mid} & \mathbf{\Lambda}_{opt} & t(s) & 0 & 0 \\ \mathbf{V}_{pub} & \mathbf{V}_{mid} & \mathbf{0} & 0 & t(s) & 0 \\ \mathbf{W}_{pub} & \mathbf{W}_{mid} & \mathbf{0} & 0 & 0 & t(s) \end{pmatrix}, \tag{25}$$

where $\mathbf{\Lambda}_{pub} = (\lambda_1(s), \ldots, \lambda_{n_{pub}}(s))$, $\mathbf{\Lambda}_{mid} = (\lambda_{n_{pub}+1}(s), \ldots, \lambda_{n+m-n'}(s))$, $\mathbf{\Lambda}_{opt} = (\lambda_{n+m-n'+1}(s), \ldots, \lambda_{n+m}(s))$, $\mathbf{V}_{pub} = (v_1(s), \ldots, v_{n_{pub}}(s))$, $\mathbf{V}_{mid} = (v_{n_{pub}+1}(s), \ldots, v_{n+m-n'}(s))$, $\mathbf{W}_{pub} = (w_1(s), \ldots, w_{n_{pub}}(s))$, $\mathbf{W}_{mid} = (w_{n_{pub}+1}(s), \ldots, w_{n+m-n'}(s))$, and the sets of polynomials $\mathcal{V} = \{v_i(X)\}_{i=1}^{n+m}, \mathcal{W} = \{w_i(X)\}_{i=1}^{n+m}$ are the ones associated to the circuit, which satisfy that $v_j(r_i) = f_{i,j}$, $w_j(r_i) = g_{i,j}$ for the constants of Lemma 11. It samples $\boldsymbol{k} \leftarrow \mathbb{Z}_p^4$ and publishes $\sigma_{\phi,\mathsf{lin}} = ([\mathbf{Q}^\top \boldsymbol{k}]_1, [\boldsymbol{k}]_2)$. It outputs the final common reference string:

$$\sigma_\phi = (\mathit{\Sigma_\Phi}, [\mathbf{Q}]_1, [u_L(s)]_2, [u_R(s)]_1, [u_0(s)]_T, \phi, n_{pub}, \sigma_{\phi,\mathsf{lin}}).$$

$\mathsf{Prove}(\phi, \sigma_\phi, (\boldsymbol{x}_{pub}, \boldsymbol{x}_{sec}, \boldsymbol{y}))$**:** From $(\boldsymbol{x}_{pub}, \boldsymbol{x}_{sec}, \boldsymbol{y})$ the prover generates a (redundant) satisfiability witness $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$ which satisfies the constraints of Lemma 12.

1. Sample $\delta_1, \delta_2, \delta_3 \leftarrow \mathbb{Z}_p$ and commit to $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$ a witness for satisfiability as:

$$[O]_1 = \sum_{i=1}^{n+m} c_i[\lambda_i(s)]_1 + \delta_1[t(s)]_1 \qquad [L]_1 = \sum_{i=1}^{n+m} a_i[\lambda_i(s)]_1 + \delta_2[t(s)]_1$$

$$[R]_\gamma = \sum_{i=1}^{n+m} b_i[\lambda_i(s)]_\gamma + \delta_3[t(s)]_\gamma, \qquad \gamma \in \{1, 2\}.$$

2. Let

$$\ell(X) = \left(\sum_{i=n+1}^{n+m} a_i\lambda_i(X) + \delta_2 t(X)\right) \qquad r(X) = \left(\sum_{i=n+1}^{n+m} b_i\lambda_i(X) + \delta_3 t(X)\right),$$

$$o(X) = \left(\sum_{i=n+1}^{n+m} c_i\lambda_i(X) + \delta_1 t(X)\right).$$

Let $u_L(X), u_R(X), u_0(X)$ be the polynomials associated to ϕ as described in Lemma 12 of degree at most $n + m - 1$. Define

$$p(X) = \ell(X)r(X) + \ell(X)u_L(X) + r(X)u_R(X) + u_0(X) - o(X).$$

Compute the polynomial $h(X)$ such that $p(X) = h(X)t(X)$. Compute $[H]_1 = [h(s)]_1$ with the powers $\{[s^i]_1\}_{i=0}^{N-2}$ given in $\mathit{\Sigma_\Phi}$.

3. Let $[O_{pub}]_1 = \sum_{i=1}^{n_{pub}} c_i[\lambda_i(s)]_1 + \sum_{i=n+m-n'+1}^{n+m} c_i[\lambda_i(s)]_1$. Define $\boldsymbol{z}^\top = \big(O_{pub}, O, L, R\big)$, and compute a proof that $[\boldsymbol{z}]_1$ is in the column span of $\mathbf{Q}$ as $[\pi]_1 = (\boldsymbol{c}^\top, \boldsymbol{\delta}^\top)[\mathbf{Q}^\top \boldsymbol{k}]_1$. The output of the algorithm is

$$\mathit{\Pi} = ([L]_1, [R]_1, [R]_2, [O]_1, [H]_1, [\pi]_1).$$

$\mathsf{Vrfy}((\phi, \boldsymbol{x}_{pub}, \boldsymbol{y}), \sigma_\phi, \mathit{\Pi})$**:** On input the proof $\mathit{\Pi} = ([L]_1, [R]_1, [\hat{R}]_2, [O]_1, [H]_1, [\pi]_1)$ for some instance $(\phi, \boldsymbol{x}_{pub}, \boldsymbol{y})$, output 1 if the following checks are successful and 0 otherwise:

1. $e([L]_1, [R]_2) + e([L]_1, [u_L(s)]_2) + e([u_R(s)]_1, [R]_2) + [u_0(s)]_T - e([O]_1, [1]_2) = e([H]_1, [t(s)]_2)$.
2. Compute $[O_{pub}]_1 = \sum_{i=1}^{n_{pub}} c_i[\lambda_i(s)]_1 + \sum_{i=n+m-n'+1}^{n+m} c_i[\lambda_i(s)]_1$, $[\boldsymbol{z}^\top]_1 = [O_{pub}, O, L, R]_1$ and verify if $e([\pi]_1, [1]_2) = e([\boldsymbol{z}^\top]_1, [\boldsymbol{k}]_2)$.
3. $e([R]_1, [1]_2) = e([1]_1, [\hat{R}]_2)$.

For zero-knowledge, the simulator works analogously as in the argument for arithmetic circuit satisfiability of Sect. 5. The proof of soundness is also analogous the same as in The proof of knowledge soundness is the same as the matrix $\mathbf{Q}$ is the same except for the definition of the two last rows (which depend on the polynomials defined by the affine constraint). The only point where the definition of $\mathbf{Q}$ comes into play is the point in which one argues that $\boldsymbol{q}_j(X)^\top k(\boldsymbol{X})$ are linearly independent polynomials. In this case, this is obvious as the second row $\mathbf{Q}$ are the Lagrangian polynomials evaluated at s. Lagrangian polynomials are linearly independent, as one can see by the fact that there exists a set of evaluation points (the set $\mathcal{R}$) such that for each point in this set only one of the polynomials is non-zero. We conclude that for this matrix $\mathbf{Q}$, the QANIZK argument is also an argument of knowledge and it is possible to extract the coefficients of $p(X)$ from any generic adversary. On the other hand, if the adversary breaks soundness, $t(X)$ does not divide $p(X)$ and the adversary has negligible probability of computing $[H]_1 = [p(s)/t(s)]_1$.

Efficiency. The cost of the proof is $(5, 1)$ as in the arithmetic case. Naive verification of the equations requires to compute 11 pairings, but with batching the cost reduces to 6 pairings.

C.4 Second NIZK Argument in the Generic Group Model

We now give an alternative construction base on the second characterization of boolean circuits, inspired on the "Square Span Program" construction of [10]. In this subsection, we construct a SNARK for proving satisfiability of any circuit on n inputs and m boolean gates such that $n + 2m \leq N$.

That is, in this case we let $\mathcal{R}$ be a relation generator which on input 1^λ outputs a family of polynomial time decidable relations $R_\Phi = \{R_\phi : \phi \in \Phi\}$, where,

- for some N, Φ is the set of all boolean circuits such that if the number of inputs is n, the number of gates is m, then $m + 2n \leq N$,
- $R_\phi = \{(\phi, (\boldsymbol{x}_{pub}, \boldsymbol{x}_{sec}), \boldsymbol{y}) : \phi(\boldsymbol{x}_{pub}, \boldsymbol{x}_{sec}) = \boldsymbol{y}\}$.

Further, the associated language is $\mathcal{L}_\phi = \{(\phi, \boldsymbol{x}_{pub}, \boldsymbol{y}) : \exists \boldsymbol{x}_{sec}, (\boldsymbol{x}_{pub}, \boldsymbol{x}_{sec}, \boldsymbol{y}) \in \mathcal{R}_\phi\}$. The circuit ϕ computes a function $\{0,1\}^n \to \{0,1\}^{n'}$ for some $n, n' \in \mathbb{N}$. The gates of ϕ are arbitrary (non-trivial) fan-in two gates.

$\mathsf{Setup}(R_\Phi)$**:** This algorithm samples $s \leftarrow \mathbb{Z}_p^*$ and publishes

$$\Sigma_\Phi = \left(gk, \{[\lambda_i(s)]_{1,2}\}_{i=1}^N, \{[s^i]_{1,2}\}_{i=1}^N, [t(s)]_{1,2}\right),$$

where $\lambda_i(X)$ are the Lagrangian polynomials associated to some set $\mathcal{R} = \{r_1, \ldots, r_N\} \subset \mathbb{Z}_p$. The simulation trapdoor is $\tau_\Sigma = s$.

$\mathsf{Drv.Setup}(\phi, n_{pub}, \Sigma_\Phi)$**:** On input a circuit $\phi : \mathbb{Z}_p^n \to \mathbb{Z}_p^{n'}$ with m multiplication gates, $n_{pub} \leq n$ public inputs and such that $n + m \leq N$ and Σ_Φ, this algorithm computes $([\mathbf{Q}]_1, [\boldsymbol{q}_0]_1) \in \mathbb{G}_1^{3 \times n+2m+2} \times \mathbb{G}_1^3$, defined as:

$$\mathbf{Q} = \begin{pmatrix} \boldsymbol{\Lambda}_{pub} & \mathbf{0} & \boldsymbol{\Lambda}_{opt} & 0 & 0 \\ \boldsymbol{\Lambda}_{pub} & \boldsymbol{\Lambda}_{mid} & \boldsymbol{\Lambda}_{opt} & t(s) & 0 \\ \mathbf{V}_{pub} & \mathbf{V}_{mid} & \mathbf{0} & 0 & t(s) \end{pmatrix}, \qquad \boldsymbol{q}_0 = \begin{pmatrix} 0 \\ 0 \\ v_0(s) \end{pmatrix}, \tag{26}$$

where $\boldsymbol{\Lambda}_{pub} = (\lambda_1(s), \ldots, \lambda_{n_{pub}}(s))$, $\boldsymbol{\Lambda}_{mid} = (\lambda_{n_{pub}+1}(s), \ldots, \lambda_{n+m-n'}(s))$, $\boldsymbol{\Lambda}_{opt} = (\lambda_{n+m-n'+1}(s), \ldots, \lambda_{n+m}(s))$, $\mathbf{V}_{pub} = (v_1(s), \ldots, v_{n_{pub}}(s))$, $\mathbf{V}_{mid} = (v_{n_{pub}+1}(s), \ldots, v_{n+m-n'}(s))$, and the set of polynomials $\mathcal{V} = \{v_i(X)\}_{i=0}^{n+m}$ is the one which satisfies $v_0(r_i) = f_i$, $v_j(r_i) = f_{i,j}$, for the constants of Lemma 13. It samples $\boldsymbol{k} \leftarrow \mathbb{Z}_p^3$ and publishes $\sigma_{\phi,\mathsf{lin}} = ([\mathbf{Q}^\top \boldsymbol{k}]_1, [\boldsymbol{k}]_2)$. It outputs the final common reference string:

$$\sigma_\phi = (\Sigma_\Phi, [\mathbf{Q}]_1, [\boldsymbol{q}_0]_1, \phi, n_{pub}, \sigma_{\phi,\mathsf{lin}}).$$

$\mathsf{Prove}(\phi, \sigma_\phi, (\boldsymbol{x}_{pub}, \boldsymbol{x}_{sec}, \boldsymbol{y}))$**:** From $(\boldsymbol{x}_{pub}, \boldsymbol{x}_{sec}, \boldsymbol{y})$ the prover generates a satisfiability witness $(\boldsymbol{c}, \boldsymbol{d})$ which satisfies the constraints of Lemma 13.

1. Commit to $(\boldsymbol{c}, \boldsymbol{d})$ in $\mathbb{G}_1$ and $\mathbb{G}_2$ as $[O]_{1,2} = \sum_{i=1}^{n+m} c_i[\lambda_i(s)]_1 + \sum_{i=1}^{m} d_i[\lambda_{n+m+i}(s)]_1 + \delta[t(s)]_1$, $\delta \leftarrow \mathbb{Z}_p$.

2. Define
$$p(X) = \left(\sum_{i=1}^{n+m} c_i\lambda_i(X) + \sum_{i=1}^{m} d_i\lambda_{n+m+i}(X) + \delta t(X) - 1\right)^2 - 1.$$
Compute the polynomial $h(X)$ such that $p(X) = h(X)t(X)$ and $[H]_1 = [h(s)]_1$ with the powers $\{[s^i]_1\}_{i=0}^N$ given in Σ_Φ.
3. Let $[O_{pub}]_1 = \sum_{i=1}^{n_{pub}} c_i[\lambda_i(s)]_1 + \sum_{i=n+m-n'+1}^{n+m} c_i[\lambda_i(s)]_1$.
Define $\boldsymbol{z}^\top = \big(O_{pub}, O, O\big)$, and compute a proof that $[\boldsymbol{z}]_1$ is in the column span of $\mathbf{Q}$ as $[\pi]_1 = (\boldsymbol{c}^\top, \boldsymbol{d}^\top, \boldsymbol{\delta}^\top)[\mathbf{Q}^\top\boldsymbol{k}]_1$. The output of the algorithm is
$$\Pi = ([O]_1, [O]_2, [H]_1, [\pi]_1).$$

$\mathsf{Vrfy}((\phi, \boldsymbol{x}_{pub}, \boldsymbol{y}), \sigma_\phi, \Pi)$**:** On input the proof $\Pi = ([O]_1, [\hat{O}]_2, [H]_1, [\pi]_1)$ for some instance $(\phi, \boldsymbol{x}_{pub}, \boldsymbol{y})$, output 1 if the following checks are successful and 0 otherwise:
1. $e([O]_1 - [1]_2, [O]_2 - [1]_2) - [1]_T = e([H]_1, [t(s)]_2)$.
2. Compute $[O_{pub}]_1 = \sum_{i=1}^{n_{pub}} c_i[\lambda_i(s)]_1 + \sum_{i=n+m-n'+1}^{n+m} c_i[\lambda_i(s)]_1$, $[\boldsymbol{z}^\top]_1 = [O_{pub}, O, O]_1$ and verify if $e([\pi]_1, [1]_2) = e([\boldsymbol{z}^\top]_1, [\boldsymbol{k}]_2)$.
3. $e([O]_1, [1]_2) = e([1]_1, [\hat{O}]_2)$.

Efficiency. The cost of the proof is $(3, 1)$ as in the square span program construction of [10], but our scheme has the updatable and universal properties discussed before. Naive verification of the equations requires to compute 8 pairings, but with batching the cost reduces to 4 pairings, by grouping together all terms of the form $e([O]_1, \cdot)$ and all terms of the form $e([1]_1, \cdot)$.

C.5 A New Argument based on Weaker Assumptions

From Lemma 14, we can design an argument for boolean circuit satisfiability based on weaker assumptions, similar as in Sect. 6. The argument is based on a quadratic and a linear "knowledge transfer" subarguments. The linear argument is identical to the arithmetic case.

For the quadratic argument, now the prover needs to show (aggregating the proof at each level i for $j = 1, \ldots, n_i$) that the quadratic equations $C_{i,j} = A_{i,j}B_{i,j} + A_{i,j}\beta_{i,j} + B_{i,j}\gamma_{i,j} + \epsilon_{i,j}$ are satisfied, whereas before the equations were $C_{ij} = A_{ij}B_{ij}$. The technique to aggregate them, inspired by the quadratic span programs of [16] as well as the proof, follows exactly the same steps. Security also reduces to the $(\mathcal{R}, \boldsymbol{m})$-Rational Strong Diffie-Hellman Assumption, where $m = \max_{i=0,\ldots,d} n_i$.

Indeed, the verification equation of the quadratic argument is adapted to the new equation type. For each level $i = 1, \ldots, d$, given commitments $[L_i]_1, [R_i]_2, [O_i]_1$, and some value $[H_i]_1$ the quadratic argument checks if
$$e([L_i]_1, [R_i]_2) + e([L_i]_1, [u_{L,i}(s)]_2) + e([u_{R,i}(s)]_1, [R_i]_2) + e([u_{0,i}(s)]_1, [1]_2) - e([O_i]_1, [1]_2) = e([H]_1, [T]_2),$$
where $u_{L,i}(X), u_{R,i}(X), u_{0,i}(X)$ are the polynomials associated to the gate constants at level i. To prove soundness, given an opening of $[L_i]_1$ and $[R_i]_2$ which is not consistent with $[O_i]$, it suffices to compute $[O']_1, [H']_1$ consistent with these openings and subtract the two verification equations to find a solution to the $(\mathcal{R}, \boldsymbol{m})$-Rational Strong Diffie-Hellman Assumption.

Zero-Knowledge . The argument can be made zero-knowledge for the middle wires by proving with the GS proof system that the argument for correct circuit evaluation is satisfied, as discussed in Sect. 6.6 for the arithmetic case. The input can also be hidden provided it is encrypted with an extractable commitment. In the boolean case this can be done in a relatively efficient way under falsifiable assumptions. In particular, a GS commitment to a boolean value is extractable. The cost of giving the committed inputs and a proof that they open to $\{0, 1\}$ using the GS proof system is $(6n_0, 6n_0)$ group elements and can be reduced to $(2n_0 + 10, 10)$ group elements under standard assumptions using the results of [20], but at the price of having a common reference string quadratic in n_0 and to $(2n_0 + 4, 6)$ under a non-standard (falsifiable) q-assumption using the results of [11].