Sumcheck Arguments and their Applications

Jonathan Bootle
jbt@zurich.ibm.com
IBM Research – Zurich

Alessandro Chiesa
alexch@berkeley.edu
UC Berkeley

Katerina Sotiraki
katesot@berkeley.edu
UC Berkeley

March 14, 2021

Abstract

We introduce a class of interactive protocols, which we call *sumcheck arguments*, that establishes a novel connection between the sumcheck protocol (Lund et al. JACM 1992) and folding techniques for Pedersen commitments (Bootle et al. EUROCRYPT 2016).

Informally, we consider a general notion of bilinear commitment over modules, and show that the sumcheck protocol applied to a certain polynomial associated with the commitment scheme yields a succinct argument of knowledge for openings of the commitment. Building on this, we additionally obtain succinct arguments for the NP-complete language R1CS over certain rings.

Sumcheck arguments enable us to recover as a special case numerous prior works in disparate cryptographic settings (such as discrete logarithms, pairings, groups of unknown order, lattices), providing one abstract framework to understand them all. Further, we answer open questions raised in prior works, such as obtaining a lattice-based succinct argument from the SIS assumption for satisfiability problems over rings.

Keywords: sumcheck protocol; succinct arguments; scalar-product protocol

Contents

1 Introduction

Sumcheck protocols. The sumcheck protocol is an interactive proof introduced in [LFKN92] that has played a fundamental role in the theory of probabilistic proofs in complexity theory (e.g., [BFL91; BFLS91; GKR08]) and, more recently, in cryptography. The sumcheck protocol has been used widely in a line of works on *succinct arguments* [CMT12; VSBW13; Wah+17; ZGKPP17; WTSTW18; XZZPS19; Set20]. One of the main benefits of the sumcheck protocol is that, in certain settings, the prover can be implemented in a linear number of operations [Tha13] or as a streaming algorithm [CMT12]; this avoids operations such as the Fast Fourier Transform (common in other succinct arguments) that are costly in time and in memory. The sumcheck protocol also satisfies strong soundness properties that facilitate proving the security of the Fiat–Shamir transformation in the plain model [CCHLRR18], which is notoriously hard to analyze for many other interactive proofs. Moreover, variants of the sumcheck protocol have spawned new lines of research: the univariate sumcheck of [BCRSVW19] was used in numerous succinct arguments [BCGGRS19; ZXZS; CHMMVW20; COS20; CFFQR20; BFHVXZ20]; and the sumcheck protocol for tensor codes of [Mei13] was used to obtain probabilistic proofs with rate 1 [RR20] and with linear-time provers [BCG20; BCL20].

Folding techniques. Separately, a line of works starting with [BCCGP16] constructs succinct arguments based on "folding techniques" for Pedersen commitments in the discrete logarithm setting. Informally, to prove knowledge of a long message opening of a given Pedersen commitment, the prover engages with the verifier in a reduction that halves the message length by folding the message "around" a verifier challenge. This can be repeatedly applied until the message length is small enough to send the message directly. Beyond commitment openings, [BCCGP16] give protocols for scalar-product relations, which lead to succinct arguments for NP languages such as arithmetic circuit satisfiability. These succinct arguments can be realized via a linear number of group scalar multiplications, or alternatively as streaming algorithms [BHRRS20].

Folding techniques, subsequently improved in [BBBPWM18], have been deployed as part of cryptocurrencies (Monero [Mon] and PIVX [Piv]) and are widely used thanks to popular open-source libraries [dalek18; Adj]. These practical applications have motivated careful analyses of concrete security [JT20], which facilitates setting security parameters in applications.

Folding techniques have been adapted to work in other cryptographic settings, such as pairings [LMR19], groups of unknown order (GUO) [BFS20], and lattices [BLNS20]. They have also been formulated in more abstract settings: [BMMTV19] study sufficient properties of commitment schemes which enable folding techniques; and [AC20; ACF20; ACR20] have applied folding techniques to general group homomorphisms.

Folding techniques for Pedersen (and related) commitments are arguably not fully understood, despite the numerous works and applications mentioned above. For example, they are typically used as non-interactive arguments after the Fiat–Shamir transformation is applied to the (public-coin) interactive argument. Yet the security of this non-interactive argument, even in the random oracle model, has only been proven via a superpolynomial-time extractor [BMMTV19] or in the algebraic group model [GT20]. Moreover, almost all succinct arguments are obtained via some type of probabilistic proof (and there are settings where this is inherent [RV09; CY20]) but no such probabilistic proof is evident in folding techniques.

A connection? The sumcheck protocol and folding techniques seem rather different protocols but they share several common features. Both protocols have a prover that can be realized via a linear number of operations [Tha13; BCCGP16], or alternatively as a streaming algorithm [CMT12; BHRRS20]; moreover, both protocols satisfy similar notions of strong soundness [CCHLRR18; GT20], which facilitate proving useful security properties. Are these similarities mere coincidences?

1.1 Our results

We introduce a class of interactive protocols, which we call *sumcheck arguments*, that establishes a novel connection between the sumcheck protocol and folding techniques for Pedersen commitments. This provides a single framework to understand numerous prior works in disparate cryptographic settings (such as discrete logarithms, pairings, GUO, lattices) and also enables us to answer open questions raised in prior works. We elaborate on these contributions below, and summarize the technical ideas underlying them in Section 2.

(1) Sumcheck arguments. Recall that the sumcheck protocol is an interactive proof for statements of the form $\sum_{\vec{\omega}\in H^\ell} p(\vec{\omega}) = \tau$ for a given summation domain H, ℓ-variate polynomial p, and claimed sum τ. While typically stated for polynomials over finite fields, the sumcheck protocol works for polynomials over any module M over a ring R (given certain mild conditions).[1] Let $\Sigma[R, M, H, \ell, p, \tau]$ denote the sumcheck protocol for the statement $\sum_{\vec{\omega}\in H^\ell} p(\vec{\omega}) = \tau$ when $H \subseteq R$, $p \in M[X_1, \ldots, X_\ell]$, and $\tau \in M$.

A *sumcheck argument* is, informally, a sumcheck protocol used to succinctly prove knowledge of openings for certain commitments. Let CM be a commitment scheme that is *bilinear*, which informally means that (a) the message space $\mathbb{M}^n$, commitment key space $\mathbb{K}^n$, and commitment space $\mathbb{C}$ are all modules over a ring R; and (b) the commitment function is linear in the "first-half" of the message and commitment key, and is also linear in the "second-half" of the message and commitment key. We observe that we can transform the statement "I know m such that $\mathsf{CM.Commit}(\mathsf{ck}, \mathsf{m}) = \mathsf{cm}$" into the sumcheck-like statement "I know m such that $\sum_{\vec{\omega}\in\{-1,1\}^{\log n}} f_{\mathsf{CM}}(p_{\mathsf{ck}}(\vec{\omega}), p_{\mathsf{m}}(\vec{\omega})) = \mathsf{cm}$" for certain low-degree polynomials $f_{\mathsf{CM}}, p_{\mathsf{ck}}, p_{\mathsf{m}}$ that depend on the scheme CM, commitment key ck, and message m, respectively. Our main technical result is to construct a knowledge extractor for the sumcheck protocol applied to such statements.

Theorem 1 (informal)**.** *Let* CM *be a bilinear commitment scheme satisfying certain properties. Let* $\mathsf{cm} \in \mathbb{C}$ *be a commitment to a message* $\mathsf{m} \in \mathbb{M}^n$ *using a commitment key* $\mathsf{ck} \in \mathbb{K}^n$*. Then*

$$\Sigma\big[R, M = \mathbb{C}, H = \{-1, 1\}, \ell = \log n, p = f_{\mathsf{CM}}(p_{\mathsf{ck}}, p_{\mathsf{m}}), \tau = \mathsf{cm}\big]$$

is an argument of knowledge for an opening to cm *with respect to* ck*. The round complexity is* $O(\log n)$*, the communication complexity is* $O(\log n)$ *elements in* $\mathbb{C}$*, and the prover and verifier complexity is* $O(n)$*.*

The above informal statement omits many technical details, such as commitment randomness and relaxed notions of commitment opening necessary to express settings over lattices or GUO.

As we elaborate in Section 2, well-known folding techniques from prior works can be viewed, perhaps surprisingly, as special cases of a sumcheck argument. We remark that while the usual security notion of the sumcheck protocol is an unconditional soundness guarantee, the security notion that we establish for a sumcheck argument is a knowledge guarantee, proved under certain properties of CM. In turn these properties may hold unconditionally or under certain hardness assumptions (we give examples of this in Section 2).[2]

(2) Succinct arguments for R1CS over rings. Building on sumcheck arguments, we obtain zero-knowledge succinct arguments for satisfiability problems defined over *rings*. This is in contrast to most prior succinct arguments, which support satisfiability problems defined over prime-order fields (which are the "scalar fields" associated to underlying cryptographic prime-order groups). This extension is motivated by the fact that certain computations are more efficiently expressed over certain rings (e.g., approximate arithmetic [CCKP19]), and parallels analogous lines of work for secret-sharing schemes and multiparty computation protocols [CDESX18; ACDEY19; Abs+20] for supporting computations defined over rings.

[1]A module is a mathematical structure the extends a vector space by allowing scalars to be from a ring rather than a field.

[2]Thus sumcheck arguments are distinct from direct algebraic generalizations of the sumcheck protocol to rings [CCKP19].

In more detail, we focus on the ring variant of the NP-complete problem known as *rank-1 constraint satisfiability* (R1CS), which is a widely used generalization of arithmetic circuit satisfiability. We obtain a zero-knowledge succinct argument for R1CS over any ring R_* with suitable algebraic properties, assuming the hardness of the *bilinear relation assumption* over a related ring, which is a natural generalization of assumptions such as the DL assumption, the SIS assumption, and others.

Definition 1 (informal)**.** *The R1CS problem asks: given a ring R_*, coefficient matrices $A, B, C \in R_*^{N \times N}$ each containing at most $M = \Omega(N)$ non-zero entries, and an instance vector x over R_*, is there a witness vector $\vec{w}$ over R_* such that $\vec{z} := (x, \vec{w}) \in R_*^N$ and $A\vec{z} \circ B\vec{z} = C\vec{z}$? (Here "$\circ$" denotes the entry-wise product of vectors over R_*.)*

Theorem 2. *Let R_* be a ring with suitable algebraic properties. Assuming hardness of the bilinear relation assumption over a related ring, there is a zero-knowledge succinct argument of knowledge for the R1CS problem over R_*. For $N \times N$ coefficient matrices with at most M non-zero entries, the argument has round complexity $O(\log N)$, communication complexity $O(\log N)$, and prover and verifier complexity $O(M)$.*

One immediate application of our result is to lattice cryptography. Prior work used folding techniques to obtain (zero-knowledge) succinct arguments of knowledge for lattice commitments [BLNS20], but left open the question of obtaining succinct arguments for NP-complete problems relevant to lattices.[3] Our Theorem 2 directly implies a solution to this open question. This may seem surprising, as sumcheck arguments, like many other lattice-based arguments of knowledge, may only provide relaxed openings, which means that the knowledge extractor may only be able to find an opening of a multiple of a bilinear commitment. This notwithstanding we still derive from it a knowledge extractor for the R1CS problem.

Corollary 1. *Let $R_* := \mathbb{Z}_p[X]/(X^k + 1)$ for a prime p and k a power of 2. Assuming hardness of the SIS problem over a related ring, there is an argument of knowledge for R1CS over R_* with round complexity $O(\log N)$, communication complexity $O(\log N)$, and prover and verifier complexity $O(M)$.*

Our new lattice-based argument system shows that one can efficiently prove general relations over rings pertinent to lattice cryptography with the same asymptotics as in other settings, despite the fact that most lattice-based proofs of knowledge suffer from relaxed soundness properties. This allows users to prove statements about lattice-based encryption and signature schemes directly over their native rings rather than having to convert them into statements tractable for other proof systems, which often leads to computational overheads in practical schemes [BCOS20].

Moreover, Corollary 1 contributes a new succinct argument that is plausibly post-quantum, adding to a surprisingly short list of such candidates. (Prior constructions of plausibly post-quantum succinct arguments are from hash functions [CMS19] or lattice knowledge assumptions [BISW17; BISW18; GMNO18].) An intriguing question left open by our work is whether the *security reduction* of the construction in Corollary 1 can be carried out against an efficient quantum adversary.

Finally, returning to Theorem 2, having a single construction of a zero-knowledge succinct argument over general rings may simplify future practical applications. Our theorem enables having a single abstract implementation that can be debugged and audited once and for all, and can then be instantiated over disparate algebraic settings depending on an application's needs, by simply specifying the desired ring.

(3) On instantiations. By instantiating the bilinear commitment CM in Theorem 1 we obtain succinct arguments of knowledge for different relations of interest, as we now explain.

[3]This differs from using lattices to instantiate the collision-resistant hash function in Kilian's PCP-based protocol [Kil92], because this would not lead to a succinct argument for computations expressed over relevant rings.

As a simple example, the Pedersen commitment scheme can be formulated in an abstract setting where messages and group generators are replaced by elements of appropriate rings or modules. This *generalised Pedersen commitment scheme* satisfies the conditions in Theorem 1, either unconditionally or under the same assumptions that imply its binding properties. Our sumcheck argument for the generalised Pedersen commitment scheme thus yields succinct protocols for opening Pedersen commitments in different settings, such as discrete logarithms, pairings, GUO, and lattices.

Another example is a *generalised scalar product commitment*, which is a commitment scheme that includes a commitment to the scalar product of two parts of the message. This draws inspiration from [BCCGP16; BBBPWM18; BMMTV19] which consider bilinear commitment schemes for discrete logarithms or pairings. Proving knowledge of an opening implies that the commitment was correctly computed, and therefore in this case that a scalar-product relation is satisfied. These scalar-product commitments in fact underlie our proof of Theorem 2 based on Theorem 1.

In Figure 1 we provide a comparison between succinct arguments with comparable efficiency in prior works, classified by type of relation and algebraic setting. The table demonstrates that our sumcheck arguments recover *all prior types of relations and all algebraic settings* as special cases, and additionally contribute *new combinations that were not achieved before*.

	DL	pairings	GUO	ideal lattices
basic commitment				[BLNS20]
linear-function commitment or polynomial commitment	[ACR20; AC20]		[BFS20]	previously open
scalar-product commitment	[BCCGP16]	[LMR19]	previously open	
bilinear commitment	[BMMTV19]		previously open	
	sumcheck arguments from this work			

Figure 1: Comparison of prior works that use folding techniques to achieve succinct arguments of knowledge, and also our sumcheck arguments. The rows from top to bottom indicate increasingly more general types of commitment (and so a result in a row directly implies a result in all rows above it). The columns indicate different cryptographic settings in which the commitments are constructed. Results spanning multiple columns indicate an abstraction that simultaneously captures all those settings. We see that our work captures all prior settings and types of commitments, and also achieves functionalities that were left open by prior works.

1.2 New connections and new opportunities

The novel connection between folding techniques and the sumcheck protocol, captured by our sumcheck arguments, casts many aspects of prior works in a new light. Here are several examples.

- [BCCGP16] describes folding techniques for splitting a long vector into more than two pieces before folding, to allow trading argument size for round complexity. This corresponds to running a sumcheck argument using *polynomials of fewer variables and a higher individual degree*.
- [BBBPWM18] improves the efficiency of folding techniques via a more complicated use of verifier challenges. This corresponds to running a sumcheck argument using a *different evaluation domain*, and using polynomials described using a *different monomial basis*.
- [PLS19] gives a zero-knowledge version of folding techniques that achieves better concrete efficiency by using less prover randomness. This relates to *derandomization of zero-knowledge sumcheck arguments*.

- [CHJKS20] gives weighted inner product arguments to improve concrete efficiency of satisfiability arguments. This corresponds to running a sumcheck argument for a *weighted sum of polynomial evaluations*.
- [BMMTV19] and [BFS20] consider subprotocols for delegating expensive verifier computation to the prover. This corresponds to *delegating polynomial evaluation*, to help the verifier outsource the expensive task of evaluating polynomials of commitment keys. Sumcheck arguments neatly conceptualize the role of polynomials in folding protocols and simplify the task of applying delegation protocols in other settings.

We expect that other folding techniques such as [ACR20; Lee20] can also be viewed as sumcheck arguments.

Looking forward, many design options for sumcheck protocols (such as the use of the Lagrange basis) have not yet been tested with particular folding techniques, and may improve concrete efficiency. Further, the successful analysis of sumcheck protocols provides a roadmap for proving strong soundness properties of folding protocols in the random oracle model and in quantum settings.

1.3 Related work

Figure 1 summarizes the main relationship between sumcheck arguments for bilinear commitments and prior work that uses folding techniques. Below we additionally discuss the prior works that have studied folding techniques for abstract commitment schemes and homomorphisms.

Bünz et al. [BMMTV19] present folding techniques for doubly-homomorphic commitments over prime-order groups, which are both key-homomorphic and message homomorphic. These can capture non-linear relations such as scalar-product relations under computational assumptions.

Attema et al. [ACF20] present folding techniques for pre-images of general group homomorphisms over prime-order groups. These were extended further from prime-order groups to $\mathbb{Z}$-modules in [BDFG20], who also noted that a $\mathbb{Z}$-module homomorphism could be phrased as a Pedersen-like function. These techniques give *proofs* for homomorphisms and linear relations, without relying on any computational assumptions.

Both general group homomorphisms and doubly-homomorphic commitment schemes are special cases of bilinear commitment schemes. Our work also finds the same distinction between proofs and arguments; our sumcheck protocols for "linear" commitment schemes such as the generalised Pedersen commitment scheme do not require computational assumptions, whereas our sumchecks for "quadratic relations" do require computational assumptions. We note that according to results on interactive proofs [GH98], one cannot expect efficient folding techniques for quadratic relations without any computational assumptions, as proofs or arguments for quadratic relations imply the same for NP-complete languages.

We also note that while NP-complete relations can be interactively reduced to linear relations (under computational assumptions) as in [ACF20], the reduction uses secret-sharing techniques implemented through interpolation. This is likely to use Fast Fourier Transforms, which require linear space-complexity for the prover. By contrast, we expect that by reducing to sumcheck arguments for bilinear relations, one can benefit from the streaming properties of the sumcheck protocol, leading to a logarithmic-space prover.

In defining our sumcheck polynomials, we show that by exploiting homomorphic properties, ordinary commitment schemes can be made to take polynomials as input. In the case of ordinary (singly) homomorphic commitments, this corresponds to creating a polynomial commitment scheme by committing to each *coefficient* of a polynomial individually. Other work [ACR20] uses a similar idea to show that "bilinear gates" over pairing groups may act on polynomials, whereas we consider the more general setting of polynomials over more general modules. We also highlight an interesting duality with [BDFG20], who create commitment schemes by using polynomial commitments to each *monomial* of a polynomial individually.

Like [BMMTV19; ACF20; BDFG20], sumcheck arguments easily capture optimisations of folding techniques which compress several target commitment values into one (such as the optimisation from

[BCCGP16] to [BBBPWM18]) as folding techniques applied to alternative commitment schemes or group homomorphisms.

However, none of these works directly prove relations over more complicated rings and modules, nor capture the folding techniques of [BLNS20] over ideal lattices, which use public-coin challenges from a special set in line with other practically-efficient lattice-based protocols.

2 Techniques

We summarize the main ideas behind our results. The first few subsections are dedicated to explaining sumcheck arguments (Theorem 1) in several steps of progressive generality. In Section 2.1 we describe the connection between folding techniques and the sumcheck protocol in the simplest possible setting: succinct arguments of knowledge for Pedersen commitments. Then in Section 2.2 we show how to lift this connection to any bilinear commitment, but still in the context of prime-order groups. Finally in Section 2.3 we explain the main considerations in generalizing further to consider bilinearity over rings, and in Section 2.4 we give an example of how commitments can be formulated in this framework. After that we turn our attention to our other contributions. In Section 2.5 we discuss a generic scalar-product protocol built from sumcheck arguments, and then in Section 2.6 explain how it enables us to obtain zero-knowledge succinct arguments for R1CS over rings (Theorem 2). Finally, in Section 2.7 we discuss how we also obtain polynomial commitment schemes over rings from sumcheck arguments.

2.1 From split-and-fold to sumcheck

2.1.1 Split-and-fold techniques for Pedersen commitments

We review a simple recursive technique for proving knowledge of an opening of a given Pedersen commitment. For now we ignore the goal of zero knowledge, and instead focus on achieving communication complexity that is much smaller than (indeed, logarithmic in) the message whose knowledge is being proved. The technique, introduced in [BCCGP16], relies on a "split-and-fold" interactive reduction that halves the message length and, when applied recursively, yields the desired protocol.

The construction below, which we refer to as Π_{F} (for "folding"), illustrates this reduction. Let $\mathbb{G}$ be a group of prime order q and let $\mathbb{F}$ be the finite field of size q.

Protocol 1: split-and-fold for Pedersen commitments (Π_{F})

For $n = 2^\ell$, the prover and verifier receive as input a commitment key $\vec{\mathsf{G}} \in \mathbb{G}^n$ and commitment $\mathsf{C} \in \mathbb{G}$. The prover also receives as input an opening $\vec{a} \in \mathbb{F}^n$ such that $\mathsf{C} = \langle \vec{a}, \vec{\mathsf{G}} \rangle$.

If $n = 1$ then the prover sends $a \in \mathbb{F}$ to the verifier, and the verifier checks if $a \cdot \mathsf{G} = \mathsf{C}$ as claimed. If $n > 1$, the interactive reduction works as follows.

1. Parse $\vec{\mathsf{G}} \in \mathbb{G}^n$ as $(\vec{\mathsf{G}}_0, \vec{\mathsf{G}}_1) \in \mathbb{G}^{n/2} \times \mathbb{G}^{n/2}$, and $\vec{a} \in \mathbb{F}^n$ as $(\vec{a}_0, \vec{a}_1) \in \mathbb{F}^{n/2} \times \mathbb{F}^{n/2}$.
2. The prover computes "cross terms" $\mathsf{C}_- := \langle \vec{a}_0, \vec{\mathsf{G}}_1 \rangle$ and $\mathsf{C}_+ := \langle \vec{a}_1, \vec{\mathsf{G}}_0 \rangle$ and sends them to the verifier.
3. The verifier samples $r \leftarrow \mathbb{F}$ and sends r to the prover.
4. The verifier outputs the new commitment key $\vec{\mathsf{G}}' := r \cdot \vec{\mathsf{G}}_0 + \vec{\mathsf{G}}_1 \in \mathbb{G}^{n/2}$ and the new commitment $\mathsf{C}' := \mathsf{C}_- + r \cdot \mathsf{C} + r^2 \cdot \mathsf{C}_+$. The prover outputs the new opening $\vec{a}' := \vec{a}_0 + r \cdot \vec{a}_1 \in \mathbb{F}^{n/2}$.

The reduction preserves completeness because if we expand the new commitment C' then the original commitment C appears as the middle coefficient of the polynomial in r with C_+ and C_- as the other terms:

$$\langle \vec{a}', \vec{\mathsf{G}}' \rangle = \langle \vec{a}_0 + r \cdot \vec{a}_1, r \cdot \vec{\mathsf{G}}_0 + \vec{\mathsf{G}}_1 \rangle = \langle \vec{a}_0, \vec{\mathsf{G}}_1 \rangle + r \cdot \langle \vec{a}, \vec{\mathsf{G}} \rangle + r^2 \cdot \langle \vec{a}_1, \vec{\mathsf{G}}_0 \rangle = \mathsf{C}_- + r \cdot \mathsf{C} + r^2 \cdot \mathsf{C}_+ = \mathsf{C}' \ .$$

So $\vec{a}'$ is a new opening for the new commitment C' under the new commitment key $\vec{\mathsf{G}}'$. Intuitively, the reduction is secure because the prover sends the cross terms C_+ and C_- before receiving the challenge r. (Turning this intuition into a proof requires an extraction argument, which is discussed in Section 2.1.4.)

After the reduction, the prover may send the new opening $\vec{a}'$ to the verifier, who checks that $\mathsf{C}' = \langle \vec{a}', \vec{\mathsf{G}}' \rangle$. Alternatively, the interactive reduction can be applied recursively until the final opening is a single field element a, the final Pedersen commitment key is a single group element G, and the verifier checks that the final commitment C^* satisfies $\mathsf{C}^* = a \cdot \mathsf{G}$. (The total number of recursions is $\ell = \log_2 n$.)

2.1.2 A sumcheck protocol for Pedersen commitments

We describe a sumcheck-like protocol that, as we shall see, proves knowledge of an opening for a given Pedersen commitment — we call that a *sumcheck argument*. After this, in Section 2.1.3, we show that the sumcheck argument is *equivalent* to Π_{F} (Protocol 1). The protocol below is written as an interactive reduction to be repeated a logarithmic number of times. We review the original sumcheck protocol [LFKN92] in Appendix A, and refer to it as Π_{SC} from now on.

We now describe the sumcheck argument. Let $\vec{v}$ be a vector of length $n = 2^\ell$ over $\mathbb{F}$ or $\mathbb{G}$, whose entries we index via binary strings $(i_1, \ldots, i_\ell) \in \{0,1\}^\ell$; we use $\mathrm{rv}(\vec{v})$ to denote $\vec{v}$ in reverse order. We define the ℓ-variate polynomial $p_{\vec{v}}$ over $\mathbb{F}$ or $\mathbb{G}$ by

$$p_{\vec{v}}(X_1, \ldots, X_\ell) := \sum_{i_1, \ldots, i_\ell \in \{0,1\}} v_{i_1, \ldots, i_\ell} X_1^{i_1} \cdots X_\ell^{i_\ell} \ .$$

Protocol 2: sumcheck argument (Π_{SA})

For $n = 2^\ell$, the prover and verifier receive as input a commitment key $\vec{\mathsf{G}} \in \mathbb{G}^n$ and commitment $\mathsf{C} \in \mathbb{G}$. The prover also receives as input an opening $\vec{a} \in \mathbb{F}^n$ such that $\mathsf{C} = \langle \vec{a}, \vec{\mathsf{G}} \rangle$.

For $H := \{-1, 1\} \subseteq \mathbb{F}$, the prover and verifier engage in a sumcheck round for the claim

$$\sum_{\omega_1, \ldots, \omega_\ell \in H} p_{\vec{a}}(\omega_1, \ldots, \omega_\ell) \cdot p_{\mathrm{rv}(\vec{\mathsf{G}})}(\omega_1, \ldots, \omega_\ell) \cdot \frac{\omega_1}{2} \cdots \frac{\omega_\ell}{2} = \mathsf{C} \ .$$

In other words, if $\ell = 0$ then the prover sends $p_a \in \mathbb{F}$ to the verifier, and the verifier uses $p_{\mathrm{rv}(\mathsf{G})} \in \mathbb{G}$ to check if $p_a \cdot p_{\mathrm{rv}(\mathsf{G})} = \mathsf{C}$ as claimed. If $\ell > 0$ then the interactive reduction works as follows.

- The prover $\mathbf{P}$ sends the polynomial $q(X) \in \mathbb{G}[X]$ to the verifier, computed as follows:

$$q(X) := \sum_{\omega_2, \ldots, \omega_\ell \in H} p_{\vec{a}}(X, \omega_2, \ldots, \omega_\ell) \cdot p_{\mathrm{rv}(\vec{\mathsf{G}})}(X, \omega_2, \ldots, \omega_\ell) \cdot \frac{\omega_2}{2} \cdots \frac{\omega_\ell}{2} \ . \tag{1}$$

- The verifier $\mathbf{V}$ samples $r \leftarrow \mathbb{F}$ and sends r to the prover.
- The verifier checks that $\mathsf{C} = \sum_{\omega \in H} q(\omega) \cdot \frac{\omega}{2}$. (If not, it rejects.)
- The verifier outputs the new commitment key $\vec{\mathsf{G}}' \in \mathbb{G}^{n/2}$ that is the coefficients of $p_{\mathrm{rv}(\vec{\mathsf{G}})}(r, X_2, \ldots, X_\ell) \in \mathbb{G}[X_2, \ldots, X_\ell]$ in reverse order, and the new commitment $\mathsf{C}' := q(r) \in \mathbb{G}$. The prover outputs the new opening $\vec{a}' \in \mathbb{F}^{n/2}$ that is the coefficients of $p_{\vec{a}}(r, X_2, \ldots, X_\ell) \in \mathbb{F}[X_2, \ldots, X_\ell]$. The new sumcheck claim about these is:

$$\sum_{\omega_2, \ldots, \omega_\ell \in H} p_{\vec{a}'}(\omega_2, \ldots, \omega_\ell) \cdot p_{\mathrm{rv}(\vec{\mathsf{G}}')}(\omega_2, \ldots, \omega_\ell) \cdot \frac{\omega_2}{2} \cdots \frac{\omega_\ell}{2} = \mathsf{C}' \ .$$

The interactive reduction can be applied recursively until the final opening is a single field element a, the final Pedersen commitment key is a single group element G, and the verifier checks that the final commitment C^* satisfies $\mathsf{C}^* = p_a \cdot p_{\mathrm{rv}(\mathsf{G})}$. The sumcheck argument Π_{SA} is remarkably similar to the sumcheck protocol Π_{SC}. The two protocols, however, also have differences: (a) in Π_{SA} the summation is over a particular polynomial p derived from the commitment key and opening; (b) when $\ell = 0$, in Π_{SA} the prover sends the witness to the verifier to allow them to check the final verification equation, whereas in Π_{SC} the verifier checks the final verification equation alone; (c) as we elaborate in Section 2.1.4, Π_{SA} and Π_{SC} have different security guarantees.

Before we proceed, however, we wish to clarify that Π_{SA} is mathematically well-defined. The "multiplication" operation implicit in the expression $p_{\vec{a}} \cdot p_{\mathrm{rv}(\vec{\mathsf{G}})}$ which maps $\mathbb{F}[X_1, \ldots, X_\ell] \times \mathbb{G}[X_1, \ldots, X_\ell] \to \mathbb{G}[X_1, \ldots, X_\ell]$, is a natural extension of the scalar multiplication operation $a \cdot \mathsf{G}$ which maps $\mathbb{F} \times \mathbb{G} \to \mathbb{G}$.

Consider the polynomials $P(X) = a + a' \cdot X \in \mathbb{F}[X]$ and $Q(X) = \mathsf{G} + X \cdot \mathsf{G}' \in \mathbb{G}[X]$, and let $r \in \mathbb{F}$. The product of $P(r)$ and $Q(r)$ can be written as follows:

$$\begin{aligned} P(r) \cdot Q(r) &= (a + a'r) \cdot (\mathsf{G} + r \cdot \mathsf{G}') \\ &= a \cdot (\mathsf{G} + r \cdot \mathsf{G}') + a'r \cdot (\mathsf{G} + r \cdot \mathsf{G}') \\ &= a \cdot \mathsf{G} + ar \cdot \mathsf{G}' + a'r \cdot \mathsf{G} + a'r^2 \cdot \mathsf{G}' \\ &= a \cdot \mathsf{G} + r \cdot (a \cdot \mathsf{G}' + a' \cdot \mathsf{G}) + r^2 \cdot (a' \cdot \mathsf{G}') \ , \end{aligned}$$

where the second and third equalities follow from the *bilinear properties* of scalar multiplication.[4] This holds for any $r \in \mathbb{F}$, and so it makes sense to define the "scalar multiplication" of $P(X)$ and $Q(X)$:

$$P(X) \cdot Q(X) = (a + a'X) \cdot (\mathsf{G} + X \cdot \mathsf{G}') := a \cdot \mathsf{G} + X \cdot (a \cdot \mathsf{G}' + a' \cdot \mathsf{G}) + X^2 \cdot (a' \cdot \mathsf{G}') \ .$$

The polynomial $p_{\vec{a}}(X_1, \ldots, X_\ell) \cdot p_{\mathrm{rv}(\vec{\mathsf{G}})}(X_1, \ldots, X_\ell)$, whose coefficients lie in $\mathbb{G}$, is defined this way.

Remark 2.1 (sumcheck protocol for rings)**.** Chen et al. [CCKP19] study the sumcheck protocol (Protocol 6) for polynomials defined over certain rings. This is distinct from our sumcheck argument (Protocol 2), which provides a knowledge guarantee for certain commitments rather than a soundness guarantee for a given polynomial (we elaborate on this in Section 2.1.4).

2.1.3 Equivalence between sumcheck argument and split-and-fold

We explain how Π_{SA} (Protocol 2) is essentially mathematically equivalent to Π_{F} (Protocol 1). The equivalence of the two protocols is established through a series of technical components, but is independent of the analysis of Π_{SA} and our other results. Hence, skipping this section does not affect the understanding of the rest of the discussion.

(1) Pedersen commitment as polynomial summation. One can express the Pedersen commitment $\mathsf{C} = \langle \vec{a}, \vec{\mathsf{G}} \rangle \in \mathbb{G}$ (the scalar product of a vector over $\mathbb{F}$ and a vector over $\mathbb{G}$) as a sum of evaluations of $p_{\vec{a}} \cdot p_{\mathrm{rv}(\vec{\mathsf{G}})}$:

$$\sum_{\omega_1, \ldots, \omega_\ell \in \{-1,1\}} p_{\vec{a}}(\omega_1, \ldots, \omega_\ell) \cdot p_{\mathrm{rv}(\vec{\mathsf{G}})}(\omega_1, \ldots, \omega_\ell) \cdot \frac{\omega_1}{2} \cdots \frac{\omega_\ell}{2} = \langle \vec{a}, \vec{\mathsf{G}} \rangle \ .$$

Below we follow [BCG20] which explains the same for the scalar product of two vectors over $\mathbb{F}$.

[4]For any $a, a' \in \mathbb{F}$ and $\mathsf{G}, \mathsf{G}' \in \mathbb{G}$ we have $(a + a') \cdot \mathsf{G} = a \cdot \mathsf{G} + a' \cdot \mathsf{G}$ and $a \cdot (\mathsf{G} + \mathsf{G}') = a \cdot \mathsf{G} + a \cdot \mathsf{G}'$.

Each contribution to the coefficient of $X_1 \cdots X_\ell$ in $p_{\vec{a}} \cdot p_{\mathrm{rv}(\vec{\mathsf{G}})}$ arises from a multiplication of the monomials in the terms $a_{i_1,\dots,i_\ell} X_1^{i_1} \cdots X_\ell^{i_\ell}$ and $\mathsf{G}_{i_1,\dots,i_\ell} X_1^{1-i_1} \cdots X_\ell^{1-i_\ell}$, which multiply to give $a_{i_1,\dots,i_\ell} \cdot \mathsf{G}_{i_1,\dots,i_\ell} \cdot X_1 \cdots X_\ell$. Thus, the coefficient of $X_1 \cdots X_\ell$ in $p_{\vec{a}} \cdot p_{\mathrm{rv}(\vec{\mathsf{G}})}$ is equal to $\langle \vec{a}, \vec{\mathsf{G}} \rangle = \mathsf{C}$.

Next, observe that for any univariate polynomial $P(X)$, summing $P(X) \cdot X/2$ over its evaluations at 1 and -1 gives the sum of the odd coefficients of P. Applying the same idea to $p(\vec{X}) = p_{\vec{a}}(\vec{X}) \cdot p_{\mathrm{rv}(\vec{\mathsf{G}})}(\vec{X})$ and summing $p(\vec{X}) \cdot X_1/2 \cdots X_\ell/2$ over each variable returns the sum of the coefficients $p_{\vec{i}}$ such that $\vec{i} \equiv 1 \bmod 2$. The only such non-zero coefficient is the coefficient of $X_1 \cdots X_\ell$, which is $\langle \vec{a}, \vec{\mathsf{G}} \rangle = \mathsf{C}$. This yields the claimed summation equation.

(2) New commitment key and new opening. Π_{F} and Π_{SA} produce the *same* new commitment key and new opening in each reduction. Label the entries of $\vec{\mathsf{G}}$ and $\vec{a}$ as $(\mathsf{G}_{\vec{i}})_{\vec{i} \in \{0,1\}^\ell}$ and $(a_{\vec{i}})_{\vec{i} \in \{0,1\}^\ell}$, in order of their binary representation.

- The new commitment key in Π_{F} is $\vec{\mathsf{G}}' = r \cdot \vec{\mathsf{G}}_0 + \vec{\mathsf{G}}_1 \in \mathbb{G}^{n/2}$ and its entries are $\mathsf{G}'_{i_2,\dots,i_\ell} = r \cdot \mathsf{G}_{0,\dots,i_\ell} + \mathsf{G}_{1,\dots,i_\ell} = \sum_{i_1 \in \{0,1\}} \mathsf{G}_{i_1,\dots,i_\ell} r^{1-i_1}$. In Π_{SA}, we have

$$p_{\mathrm{rv}(\vec{\mathsf{G}})}(r, X_2, \dots, X_\ell) = \sum_{i_1,\dots,i_\ell \in \{0,1\}} \mathsf{G}_{1-i_1,\dots,1-i_\ell} r^{i_1} X_2^{i_2} \cdots X_\ell^{i_\ell}$$

  and so $\mathrm{rv}(\vec{\mathsf{G}}')$ are the coefficients of $p_{\mathrm{rv}(\vec{\mathsf{G}})(r,X_2,\dots,X_\ell)}$.

- The new opening in Π_{F} is $\vec{a}' = \vec{a}_0 + r \cdot \vec{a}_1 \in \mathbb{F}_q^{n/2}$ and its entries are $a'_{i_2,\dots,i_\ell} = a_{0,\dots,i_\ell} + r \cdot a_{1,\dots,i_\ell} = \sum_{i_1 \in \{0,1\}} a_{i_1,\dots,i_\ell} r^{i_1}$. In Π_{SA}, we have

$$p_{\vec{a}}(r, X_2, \dots, X_\ell) = \sum_{i_1,\dots,i_\ell \in \{0,1\}} a_{i_1,\dots,i_\ell} r^{i_1} X_2^{i_2} \cdots X_\ell^{i_\ell}$$

  and so $\vec{a}'$ are the coefficients of $p_{\vec{a}}(r, X_2, \dots, X_\ell)$.

(3) The cross terms as coefficients of *q*. The cross terms C_+ and C_- in Π_{F} are the coefficients of the polynomial $q(X)$ in Π_{SA}. The coefficients of $q(X)$ in Equation (1) are equal to the coefficients of $X_2 \cdots X_\ell$, $X_1 X_2 \cdots X_\ell$ and $X_1^2 X_2 \cdots X_\ell$ of the polynomial $p_{\vec{a}}(\vec{X}) \cdot p_{\mathrm{rv}(\vec{\mathsf{G}})}(\vec{X})$. We have already established that the coefficient of $X_1 X_2 \cdots X_\ell$ is $\mathsf{C} = \langle \vec{a}, \vec{\mathsf{G}} \rangle$.

Next, each contribution to the $X_2 \cdots X_\ell$ term of $p_{\vec{a}} \cdot p_{\mathrm{rv}(\vec{\mathsf{G}})}$ arises from a multiplication of the monomials in the terms $a_{0,i_2,\dots,i_\ell} X_2^{i_2} \cdots X_\ell^{i_\ell}$ and $\mathsf{G}_{1,i_2,\dots,i_\ell} X_2^{1-i_2} \cdots X_\ell^{1-i_\ell}$, which multiply to give $a_{0,i_2,\dots,i_\ell} \cdot \mathsf{G}_{1,i_2,\dots,i_\ell} \cdot X_1 \cdots X_\ell$. Thus, the coefficient of $X_2 \cdots X_\ell$ in $p_{\vec{a}} \cdot p_{\mathrm{rv}(\vec{\mathsf{G}})}$ is equal to $\langle \vec{a}_0, \vec{\mathsf{G}}_1 \rangle$, which is equal to C_-. Similar reasoning applies to C_+.

(4) The verification equations. The verification equations in Π_{SA} simplify to give the calculation of the new commitment key in Π_{F}. Note that in Π_{SA} the prover sends $q(X) = q_0 + X \cdot q_1 + X^2 \cdot q_2$ to the verifier; the verifier checks that $\mathsf{C} = 1/2 \cdot q(1) - 1/2 \cdot q(-1)$. The new commitment C' is $q(r) = q_0 + r \cdot q_1 + r^2 \cdot q_2$.

However, we have already seen that an honest prover will use $(q_0, q_1, q_2) = (\mathsf{C}_-, \mathsf{C}, \mathsf{C}_+)$. Therefore, instead of performing the verification check, which amounts to checking that $\mathsf{C} = q_1$, and then computing the new instance $q(r)$, the prover can send only q_0 and q_2 to the verifier, who will use C as q_1, since the two are supposed to be equal. Thus, Π_{F} incorporates a simple optimization to Π_{SA}.

2.1.4 Soundness of the sumcheck argument

The security guarantee offered by a sumcheck argument is different from that of the sumcheck protocol. The sumcheck protocol (Protocol 6) has a soundness guarantee: if the polynomial p does not have the claimed sum τ then the verifier accepts with small probability. In contrast, the sumcheck argument (Protocol 2) has a **knowledge soundness** guarantee: there exists an extractor that, given a suitable collection of accepting transcripts for a given commitment C, efficiently finds an opening $\vec{a}$ such that $\mathsf{C} = \langle \vec{a}, \vec{\mathsf{G}} \rangle$. This difference makes sense: any given Pedersen commitment C can *always* be expressed as a scalar product of some opening $\vec{a}$ and the commitment key generators $\vec{\mathsf{G}}$; in fact, there are many different possible openings $\vec{a}$ for which this is true! Therefore, soundness is not a meaningful notion for the sumcheck argument.

The knowledge extractor for the sumcheck argument is a translation of the knowledge extractor for the split-and-fold protocol (Protocol 1) studied in prior works. Interestingly, as noted in [ACF20; BDFG20], in this case knowledge soundness can be established without relying on any computational assumptions.

Assume that the interactive reduction is applied recursively ℓ times. The knowledge extractor takes as input 3^ℓ accepting protocol executions for strings $r_1, \dots, r_\ell \in \mathbb{F}$ of verifier challenges which are arranged in a 3-ary tree format. In more detail, the root of the tree is labelled with a value of the first challenge $r_1 \in \mathbb{F}$. Each non-leaf node at level i is labelled with some challenge value r_i, and has three children, each labelled with a distinct value of r_{i+1}.

The extractor works inductively. Let $q_i(X_i)$ be a polynomial produced in the i-th reduction, for some path $(r_1, \dots, r_{i-1})$ in the tree. Given suitable openings of length $2^{\ell-i}$ to the commitment $q_i(r_i)$, for three different values of r_i (enough to specify the quadratic polynomial q_i), the extractor can find an opening of $1/2 \cdot q_i(1) - 1/2 \cdot q_i(-1)$ of length $2^{\ell-(i-1)}$. Since $1/2 \cdot q_i(1) - 1/2 \cdot q_i(-1) = q_{i-1}(r_{i-1})$ according to the verifier's checks, we can apply the same technique at the next level.

For example, at level 1 in the tree, the knowledge extractor for the split-and-fold protocol is given three different vectors $\vec{a}'_1, \vec{a}'_2, \vec{a}'_3 \in \mathbb{F}^{n/2}$, for three different challenge values $r^{(1)}, r^{(2)}, r^{(3)} \in \mathbb{F}$, such that

$$\begin{aligned}
\langle \vec{a}'_1, \vec{\mathsf{G}}'_1 \rangle &= \mathsf{C}_- + r^{(1)} \cdot \mathsf{C} + (r^{(1)})^2 \cdot \mathsf{C}_+ \,, \\
\langle \vec{a}'_2, \vec{\mathsf{G}}'_2 \rangle &= \mathsf{C}_- + r^{(2)} \cdot \mathsf{C} + (r^{(2)})^2 \cdot \mathsf{C}_+ \,, \\
\langle \vec{a}'_3, \vec{\mathsf{G}}'_3 \rangle &= \mathsf{C}_- + r^{(3)} \cdot \mathsf{C} + (r^{(3)})^2 \cdot \mathsf{C}_+ \,.
\end{aligned}$$

Above $\vec{\mathsf{G}}'_i = r^{(i)} \cdot \vec{\mathsf{G}}_0 + \vec{\mathsf{G}}_1$ is the commitment key $\vec{\mathsf{G}} = (\vec{\mathsf{G}}_0, \vec{\mathsf{G}}_1) \in \mathbb{G}^n$ folded with respect to $r^{(i)}$. The knowledge extractor can then use linear algebra to find $\vec{a} \in \mathbb{F}^n$ such that $\langle \vec{a}, \vec{\mathsf{G}} \rangle = \mathsf{C}$. This idea can then be applied recursively starting from the last round and going backwards to the first round.

Similarly, the knowledge extractor for the sumcheck argument is given an opening $\vec{a}'_i$ of length $n/2$ to the commitment $q(r^{(i)})$, with respect to commitment key $\vec{\mathsf{G}}'_i$ for $i = 1, 2, 3$, and uses these openings to find an opening of length n to the middle coefficient of $q(X)$, which is equal to $1/2 \cdot q(1) - 1/2 \cdot q(-1)$. According to the verifier's check, we know that $1/2 \cdot q(1) - 1/2 \cdot q(-1) = \mathsf{C}$, so the extractor obtained an opening to C.

A key ingredient of the knowledge extractor is the ability to double the length of known openings by manipulating multiple transcripts for a given recursion round. The Pedersen commitment, being a homomorphism into $\mathbb{G}$, allows this unconditionally. Jumping ahead, this property of a commitment scheme, which we call **invertibility**, may require computational assumptions, and is a central component of our sumcheck argument stated for the general setting of bilinear commitments (see Sections 2.2 and 2.3).

2.2 Sumcheck arguments for bilinear commitments

We explain how to formulate a sumcheck argument for proving knowledge of an opening for any *bilinear commitment* that satisfies certain functionality and security properties. We proceed in two steps: in Section 2.2.1 we focus on the special case of scalar product protocols under Pedersen commitments to gain intuition, and then in Section 2.2.2 we extend this to apply to a bilinear commitment.

2.2.1 The case of scalar-product protocols under Pedersen commitments

We have seen how to re-express the split-and-fold protocol for Pedersen commitments as a sumcheck argument. The split-and-fold techniques in [BCCGP16] actually achieve more: proving knowledge of openings of *two* Pedersen commitments such that the scalar product of the two openings is a publicly-known value. We recall that protocol below and then explain how to re-express it as a sumcheck argument too.

Protocol 3: scalar product protocol for Pedersen commitments

For $n = 2^\ell$, the prover and verifier receive as input commitment keys $\vec{\mathsf{G}}, \vec{\mathsf{H}} \in \mathbb{G}^n$, commitments $\mathsf{C}_0, \mathsf{C}_1 \in \mathbb{G}$, and target value $t \in \mathbb{F}$. The prover also receives as input openings $\vec{a}, \vec{b} \in \mathbb{F}^n$ such that $\mathsf{C}_0 = \langle \vec{a}, \vec{\mathsf{G}} \rangle$, $\mathsf{C}_1 = \langle \vec{b}, \vec{\mathsf{H}} \rangle$, and $t = \langle \vec{a}, \vec{b} \rangle$.

If $n = 1$ then the prover sends $a, b \in \mathbb{F}$ to the verifier, and the verifier checks that $a \cdot \mathsf{G} = \mathsf{C}_0$, $b \cdot \mathsf{H} = \mathsf{C}_1$, and $a \cdot b = t$ as claimed. If $n > 1$, the interactive reduction works as follows.

1. The prover and the verifier parse each vector as two parts half its size: $\vec{\mathsf{G}} = (\vec{\mathsf{G}}_0, \vec{\mathsf{G}}_1) \in \mathbb{G}^{n/2} \times \mathbb{G}^{n/2}$, $\vec{\mathsf{H}} = (\vec{\mathsf{H}}_0, \vec{\mathsf{H}}_1) \in \mathbb{G}^{n/2} \times \mathbb{G}^{n/2}$, $\vec{a} = (\vec{a}_0, \vec{a}_1) \in \mathbb{F}^{n/2} \times \mathbb{F}^{n/2}$, and $\vec{b} = (\vec{b}_0, \vec{b}_1) \in \mathbb{F}^{n/2} \times \mathbb{F}^{n/2}$.
2. The prover computes "cross terms"
$$\begin{aligned} &\mathsf{C}_{a,-} := \langle \vec{a}_0, \vec{\mathsf{G}}_1 \rangle \qquad &&\mathsf{C}_{b,-} := \langle \vec{b}_0, \vec{\mathsf{H}}_1 \rangle \qquad &&t_+ := \langle \vec{a}_0, \vec{b}_1 \rangle \\ &\mathsf{C}_{a,+} := \langle \vec{a}_1, \vec{\mathsf{G}}_0 \rangle &&\mathsf{C}_{b,+} := \langle \vec{b}_1, \vec{\mathsf{H}}_0 \rangle &&t_- := \langle \vec{a}_1, \vec{b}_0 \rangle \end{aligned}$$
and sends these commitments and values to the verifier.
3. The verifier samples $r \leftarrow \mathbb{F}$ and sends r to the prover.
4. The verifier outputs new commitment keys, commitments, and target value:
$$\begin{aligned} &\vec{\mathsf{G}}' := r \cdot \vec{\mathsf{G}}_0 + \vec{\mathsf{G}}_1 \in \mathbb{G}^{n/2} \qquad &&\mathsf{C}'_0 := \mathsf{C}_{a,-} + r \cdot \mathsf{C}_0 + r^2 \cdot \mathsf{C}_{a,+} \qquad &&t' := t_+ + r \cdot t + r^2 \cdot t_- \\ &\vec{\mathsf{H}}' := \vec{\mathsf{G}}_0 + r \cdot \vec{\mathsf{G}}_1 \in \mathbb{G}^{n/2} &&\mathsf{C}'_1 := r^2 \cdot \mathsf{C}_{b,-} + r \cdot \mathsf{C}_1 + \mathsf{C}_{b,+} && \end{aligned}$$
The prover outputs new openings $\vec{a}' := \vec{a}_0 + r \cdot \vec{a}_1$ and $\vec{b}' := r \cdot \vec{b}_0 + \vec{b}_1 \in \mathbb{F}_q^{n/2}$, which are such that $\mathsf{C}'_0 = \langle \vec{a}', \vec{\mathsf{G}}' \rangle$, $\mathsf{C}'_1 = \langle \vec{b}', \vec{\mathsf{H}}' \rangle$, and $t' = \langle \vec{a}', \vec{b}' \rangle$.

The scalar product $\mathsf{C}_0 = \langle \vec{a}, \vec{\mathsf{G}} \rangle$ can be expressed as a weighted sum of evaluations of $p_{\vec{a}} \cdot p_{\mathrm{rv}(\vec{\mathsf{G}})}$. Similarly, we can express $\mathsf{C}_1 = \langle \vec{b}, \vec{\mathsf{H}} \rangle$ via a weighted sum of $p_{\mathrm{rv}(\vec{b})} \cdot p_{\vec{\mathsf{H}}}$ and $t = \langle \vec{a}, \vec{b} \rangle$ via a weighted sum of $p_{\vec{a}} \cdot p_{\mathrm{rv}(\vec{b})}$. Hence Protocol 3 can be phrased as a sumcheck on the polynomial:
$$p := (p_{\vec{a}} \cdot p_{\mathrm{rv}(\vec{\mathsf{G}})}, p_{\mathrm{rv}(\vec{b})} \cdot p_{\vec{\mathsf{H}}}, p_{\vec{a}} \cdot p_{\mathrm{rv}(\vec{b})}) \in (\mathbb{G} \times \mathbb{G} \times \mathbb{F})[X_1, \ldots, X_\ell] \ .$$
The first two components of polynomial p are well-defined because of the implicit multiplication which maps $\mathbb{F}[X_1, \ldots, X_\ell] \times \mathbb{G}[X_1, \ldots, X_\ell] \to \mathbb{G}[X_1, \ldots, X_\ell]$ as in Section 2.1.2. The multiplication in the third

component $p_{\vec{a}} \cdot p_{\mathrm{rv}(\vec{b})}$ is a natural extension of the group operation. Viewed another way, we obtain an opening protocol for the commitment scheme that, given a commitment key $(\vec{\mathsf{G}}, \vec{\mathsf{H}})$, maps a message $(\vec{a}, \vec{b})$ to

$$\mathsf{Commit}(\vec{\mathsf{G}}, \vec{\mathsf{H}}; \vec{a}, \vec{b}) := (\langle \vec{a}, \vec{\mathsf{G}} \rangle, \langle \vec{b}, \vec{\mathsf{H}} \rangle, \langle \vec{a}, \vec{b} \rangle) \ .$$

The protocol is a sumcheck argument on a polynomial which we denote $\mathsf{Commit}(p_{\mathrm{rv}(\vec{\mathsf{G}})}, p_{\vec{\mathsf{H}}}; p_{\vec{a}}, p_{\mathrm{rv}(\vec{b})})$. We provide a more formal definition of this polynomial in Section 2.2.2. As before, the cross terms of Protocol 3 correspond exactly to the coefficients of the polynomial q in the sumcheck argument.

The above sumcheck argument can be shown to satisfy the following knowledge soundness property: there exists an extractor that, given a suitable collection of accepting transcripts for a given commitment C, efficiently finds an opening $(\vec{a}, \vec{b})$ such that $\mathsf{C} = \mathsf{Commit}(\vec{\mathsf{G}}, \vec{\mathsf{H}}; \vec{a}, \vec{b})$, assuming that the discrete logarithm problem is hard over $\mathbb{G}$. Proving knowledge soundness follows a similar approach to that for Π_{F} (Protocol 1) sketched in Section 2.1.4. The main difference is that "inverting" from a given round to the previous one involves not only solving linear equations to find openings $\vec{a}$ and $\vec{b}$ of Pedersen commitments C_0 and C_1, but also arguing that $\vec{a}$ and $\vec{b}$ are a preimage of $(\mathsf{C}_0, \mathsf{C}_1, t)$ under Commit; in other words, that $\langle \vec{a}, \vec{b} \rangle = t$. Here this requires assuming the hardness of the discrete logarithm over $\mathbb{G}$ (which one may have assumed anyway to make the commitment binding), unlike for Protocol 1 where no assumptions were necessary.[5]

In sum, we have seen that the scalar-product protocol of [BCCGP16] can be rephrased as a sumcheck argument for a specific commitment scheme. (The subsequent more optimized protocol from [BBBPWM18] can also be viewed via a related sumcheck argument.)

2.2.2 Extending to any bilinear commitment

We have seen that Protocol 1 and Protocol 3 can be phrased as sumcheck arguments on different commitment schemes. Both commitment schemes are examples of a *bilinear* commitment scheme, defined next.

Definition 2. *Let $\mathbb{M}, \mathbb{K}, \mathbb{C}$ be $\mathbb{F}_q$-vector spaces. A commitment scheme* CM *is* **bilinear** *if (i) the message* $\mathsf{m} \in \mathbb{M}^n$ *can be split into left and right messages* $(\mathsf{mL}, \mathsf{mR}) \in \mathbb{M}^{n/2} \times \mathbb{M}^{n/2}$*; (ii) the commitment key* ck *can be split into neutral, left, right commitment keys* $(\mathsf{ck}_0, \mathsf{ckL}, \mathsf{ckR}) \in \mathbb{K}_0 \times \mathbb{K}^{n/2} \times \mathbb{K}^{n/2}$*, where $\mathbb{K}_0$ might not be an $\mathbb{F}_q$-vector space; (iii) the commitment function is linear with respect to the left parts and the right parts.*

The third condition implies that for any keys $\mathsf{ckL}, \mathsf{ckL}', \mathsf{ckR}, \mathsf{ckR}'$ and messages $\mathsf{mL}, \mathsf{mL}', \mathsf{mR}, \mathsf{mR}'$,

$$\begin{aligned} &\mathsf{CM.Commit}\,(\mathsf{ck}_0, \mathsf{ckL} + \mathsf{ckL}', \mathsf{ckR}; \mathsf{mL} + \mathsf{mL}', \mathsf{mR}) \\ &= \mathsf{CM.Commit}\,(\mathsf{ck}_0, \mathsf{ckL}, \mathsf{ckR}; \mathsf{mL}, \mathsf{mR}) \\ &\quad + \mathsf{CM.Commit}\,(\mathsf{ck}_0, \mathsf{ckL}', \mathsf{ckR}; \mathsf{mL}', \mathsf{mR}) \end{aligned} \qquad \begin{aligned} &\mathsf{CM.Commit}\,(\mathsf{ck}_0, \mathsf{ckL}, \mathsf{ckR} + \mathsf{ckR}'; \mathsf{mL}, \mathsf{mR} + \mathsf{mR}') \\ &= \mathsf{CM.Commit}\,(\mathsf{ck}_0, \mathsf{ckL}, \mathsf{ckR}; \mathsf{mL}, \mathsf{mR}) \\ &\quad + \mathsf{CM.Commit}\,(\mathsf{ck}_0, \mathsf{ckL}, \mathsf{ckR}'; \mathsf{mL}, \mathsf{mR}') \end{aligned}$$

Definition 2 naturally extends to apply to hiding commitment schemes CM with commitment randomness.

Since the key space $\mathbb{K}$ and the message space $\mathbb{M}$ are $\mathbb{F}_q$*-vector spaces*, multilinear polynomials such as $p_{\mathsf{ckL}}(\vec{X})$ and $p_{\mathsf{mL}}(\vec{X})$ over $\mathbb{K}$ and $\mathbb{M}$ can be evaluated over points in $\mathbb{F}_q$ as part of Protocol 4. Then, as we shall explain shortly, bilinearity allows us to "lift" the commitment function to polynomials, which enables us to define the following polynomial

$$p(X_1, \ldots, X_\ell) = \mathsf{CM.Commit}\left(\mathsf{ck}_0, p_{\mathsf{ckL}}(\vec{X}), p_{\mathsf{ckR}}(\vec{X}); p_{\mathsf{mL}}(\vec{X}), p_{\mathsf{mR}}(\vec{X})\right) \ . \tag{2}$$

We can then obtain an opening protocol for CM via a sumcheck argument on this polynomial.

[5]Using a cryptographic assumption to prove security in this setting is not surprising because the commitment scheme is bilinear, and as we shall see directly implies succinct arguments for NP, which can only be computationally sound.

Protocol 4: sumcheck argument for a bilinear commitment

For $n = 2^\ell$, the prover and verifier receive as input a commitment key $(\mathsf{ck}_0, \mathsf{ckL}, \mathsf{ckR}) \in \mathbb{K}_0 \times \mathbb{K}^n \times \mathbb{K}^n$ and commitment $\mathsf{cm} \in \mathbb{C}$. The prover also receives as input an opening $(\mathsf{mL}, \mathsf{mR}) \in \mathbb{M}^n \times \mathbb{M}^n$ such that $\mathsf{cm} = \mathsf{CM.Commit}\left((\mathsf{ck}_0, \mathsf{ckL}, \mathsf{ckR}), (\mathsf{mL}, \mathsf{mR})\right)$.

For $H := \{-1, 1\} \subseteq \mathbb{F}$, the prover and verifier engage in a sumcheck round for the claim

$$\sum_{\vec{\omega} \in H^\ell} \mathsf{CM.Commit}\left(\mathsf{ck}_0, p_{\mathsf{ckL}}(\vec{\omega}), p_{\mathsf{ckR}}(\vec{\omega}), p_{\mathsf{mL}}(\vec{\omega}), p_{\mathsf{mR}}(\vec{\omega})\right) = 2^\ell \cdot \mathsf{cm} \ .$$

In other words, if $\ell = 0$ then the prover sends $p_{\mathsf{mL}}, p_{\mathsf{mR}} \in \mathbb{M}$ to the verifier, and the verifier uses $p_{\mathsf{ckL}}, p_{\mathsf{ckR}} \in \mathbb{K}$ to check if $\mathsf{CM.Commit}\left(\mathsf{ck}_0, p_{\mathsf{ckL}}, p_{\mathsf{ckR}}, p_{\mathsf{mL}}, p_{\mathsf{mR}}\right) = \mathsf{C}$ as claimed. If $\ell > 0$ then the prover and verifier engage in an interactive reduction, using polynomial $q(X)$ produced by summing $p(X_1, \omega_2, \ldots, \omega_\ell)$ over $\omega_2, \ldots, \omega_\ell \in H$.

After a reduction using randomness r, the verifier outputs new commitment keys $(\mathsf{ckL}', \mathsf{ckR}') \in \mathbb{K}^{n/2} \times \mathbb{K}^{n/2}$ and the new commitment $\mathsf{cm}' := q(r) \in \mathbb{C}$. The prover outputs the new opening $(\mathsf{mL}', \mathsf{mR}') \in \mathbb{F}^{n/2} \times \mathbb{F}^{n/2}$.

The above opening protocol for the bilinear commitment scheme CM has perfect completeness if CM is *extendable* and is knowledge sound if CM is *invertible*; we discuss both of these properties shortly.

Theorem 3. *Protocol 4 is an opening protocol for any bilinear commitment scheme* CM *that is extendable and invertible. (There exists an extractor that given a commitment* cm*, commitment key* $\mathsf{ck} = (\mathsf{ck}_0, \mathsf{ckL}, \mathsf{ckR})$*, and suitable tree of accepting transcripts for Protocol 4, finds an opening* $\mathsf{m} = (\mathsf{mL}, \mathsf{mR})$ *such that* $\mathsf{cm} = \mathsf{CM.Commit}(\mathsf{ck}, \mathsf{m})$*.)*

A generic sumcheck argument. Protocol 4 captures as a special case the split-and-fold techniques of Protocol 1 and Protocol 3. (In particular, Pedersen commitments are a special case of bilinear commitments where the left message input and right commitment key input are simply ignored.) Moreover, Protocol 4 contributes a powerful generic understanding for describing even more split-and-fold techniques: for the commitment scheme used in the Bulletproofs protocol [BBBPWM18], and for pairing-based commitment schemes appearing in works such as [LMR19; BMMTV19],[6] in which the message space and key space are $\mathbb{F}_q$-vector spaces, and the commitment schemes are bilinear, extendable, and have a suitable invertibility property.

Defining a sumcheck polynomial. We explain why the polynomial in Equation (2) is well-defined.

The bilinear property of the commitment scheme makes it possible to extend the commitment scheme to take polynomials as input, just as Protocol 1 and Protocol 3 could ultimately be written as sumcheck arguments because the scalar-multiplication map is bilinear.

For example, if $\mathsf{CM.Commit}(\ldots)$ is bilinear, then for messages $\mathsf{mL}, \mathsf{mL}'$, $\mathsf{mR}, \mathsf{mR}'$ and commitment keys $\mathsf{ckL}, \mathsf{ckL}'$, $\mathsf{ckR}, \mathsf{ckR}'$ and ck_0, we have

$$\begin{aligned}
&\mathsf{CM.Commit}\left(\mathsf{ck}_0, \mathsf{ckL} + r \cdot \mathsf{ckL}', \mathsf{ckR} + r \cdot \mathsf{ckR}'; \mathsf{mL} + r \cdot \mathsf{mL}', \mathsf{mR} + r \cdot \mathsf{mR}'\right) \\
&= \mathsf{CM.Commit}\left(\mathsf{ck}_0, \mathsf{ckL}, \mathsf{ckR} + r \cdot \mathsf{ckR}'; \mathsf{mL}, \mathsf{mR} + r \cdot \mathsf{mR}'\right)
\end{aligned}$$

[6] The key-homomorphic message-homomorphic commitment schemes in [BMMTV19] are a special case of bilinear commitments, where the left message input and right commitment key input are simply ignored.

$$
\begin{aligned}
&+ r \cdot \mathsf{CM.Commit}\left(\mathsf{ck}_0, \mathsf{ckL}', \mathsf{ckR} + r \cdot \mathsf{ckR}'; \mathsf{mL}', \mathsf{mR} + r \cdot \mathsf{mR}'\right) \\
&= \mathsf{CM.Commit}\left(\mathsf{ck}_0, \mathsf{ckL}, \mathsf{ckR}; \mathsf{mL}, \mathsf{mR}\right) \\
&\;+ r \cdot \left(\mathsf{CM.Commit}\left(\mathsf{ck}_0, \mathsf{ckL}, \mathsf{ckR}'; \mathsf{mL}, \mathsf{mR}'\right) + \mathsf{CM.Commit}\left(\mathsf{ck}_0, \mathsf{ckL}', \mathsf{ckR}; \mathsf{mL}', \mathsf{mR}\right)\right) \\
&\;+ r^2 \cdot \mathsf{CM.Commit}\left(\mathsf{ck}_0, \mathsf{ckL}', \mathsf{ckR}'; \mathsf{mL}', \mathsf{mR}'\right) \quad .
\end{aligned}
$$

Thus, it makes sense to write

$$
\begin{aligned}
&\mathsf{CM.Commit}\left(\mathsf{ck}_0, \mathsf{ckL} + X \cdot \mathsf{ckL}', \mathsf{ckR} + X \cdot \mathsf{ckR}'; \mathsf{mL} + X \cdot \mathsf{mL}', \mathsf{mR} + X \cdot \mathsf{mR}'\right) \\
&= \mathsf{CM.Commit}\left(\mathsf{ck}_0, \mathsf{ckL}, \mathsf{ckR}; \mathsf{mL}, \mathsf{mR}\right) \\
&\;+ X \cdot \left(\mathsf{CM.Commit}\left(\mathsf{ck}_0, \mathsf{ckL}, \mathsf{ckR}'; \mathsf{mL}, \mathsf{mR}'\right) + \mathsf{CM.Commit}\left(\mathsf{ck}_0, \mathsf{ckL}', \mathsf{ckR}; \mathsf{mL}', \mathsf{mR}\right)\right) \\
&\;+ X^2 \cdot \mathsf{CM.Commit}\left(\mathsf{ck}_0, \mathsf{ckL}', \mathsf{ckR}'; \mathsf{mL}', \mathsf{mR}'\right) \quad .
\end{aligned}
$$

With this in mind, the sumcheck polynomial is defined to be

$$
\begin{aligned}
p(X_1, \ldots, X_\ell) &= \mathsf{CM.Commit}\left(\mathsf{ck}_0, p_{\mathsf{ckL}}(\vec{X}), p_{\mathsf{ckR}}(\vec{X}); p_{\mathsf{mL}}(\vec{X}), p_{\mathsf{mR}}(\vec{X})\right) \\
&= \sum_{\vec{i} \in \{0,1\}^\ell} \mathsf{CM.Commit}\left(\mathsf{ck}_0, \mathsf{ckL}_{\vec{i}}, p_{\mathsf{ckR}}(\vec{X}); \mathsf{mL}_{\vec{i}}, p_{\mathsf{mR}}(\vec{X})\right) \cdot X_1^{i_1} \cdots X_\ell^{i_\ell} \\
&= \sum_{\vec{i},\vec{j} \in \{0,1\}^\ell} \mathsf{CM.Commit}\left(\mathsf{ck}_0, \mathsf{ckL}_{\vec{i}}, \mathsf{ckR}_{\vec{j}}; \mathsf{mL}_{\vec{i}}, \mathsf{mR}_{\vec{j}}\right) \cdot X_1^{i_1+j_1} \cdots X_\ell^{i_\ell+j_\ell} \quad .
\end{aligned}
$$

The second and third equalities above are justified by the bilinearity of the commitment scheme.

Remark 2.2. When discussing previous examples, such as Protocol 2 and Protocol 3, we used multilinear polynomials such as "$p_{\mathrm{rv}(\vec{\mathsf{G}})}$", with the entries of $\vec{\mathsf{G}}$ appearing in reverse order, in order to recover protocols already appearing in the literature as closely as possible. However, having understood the connection to sumcheck protocols, it will be sufficient, and notationally simpler, to avoid vectors in reverse order.

Completeness. We say that CM is *extendable* if for all commitment keys $(\mathsf{ck}_0, \mathsf{ck})$ (where ck includes a left and right key), and $\mathsf{m} \in \mathbb{M}^{n+n'}$, we have

$$
\begin{aligned}
\mathsf{CM.Commit}\left(\mathsf{ck}_0, \mathsf{ck}; \mathsf{m}\right) = &\mathsf{CM.Commit}\left(\mathsf{ck}_0, \mathsf{ck}[:n]; \mathsf{m}[:n]\right) \\
&+ \mathsf{CM.Commit}\left(\mathsf{ck}_0, \mathsf{ck}[n:]; \mathsf{m}[n:]\right)
\end{aligned}
$$

We sketch why if CM is extendable then Protocol 4 has perfect completeness. It suffices to express $\mathsf{cm} = \mathsf{CM.Commit}\left(\mathsf{ck}_0, \mathsf{ckL}, \mathsf{ckR}; \mathsf{mL}, \mathsf{mR}\right)$ as a sum of evaluations of the sumcheck polynomial p in Equation (2).

Consider $P(X) = (a_0 + a_1 X)(b_0 + b_1 X)$. We can obtain the scalar product $a_0 b_0 + a_1 b_1$ from $P(X)$ using $1/2 \cdot P(1) + 1/2 \cdot P(-1)$. Applying a similar idea to $p(\vec{X})$ gives

$$
\begin{aligned}
\sum_{\vec{\omega} \in \{-1,1\}^\ell} p(\vec{\omega}) &= 2^\ell \cdot \sum_{\vec{i} \equiv \vec{0} \bmod 2} p_{\vec{i}} \\
&= 2^\ell \cdot \sum_{\vec{i} \in \{0,1\}^\ell} \mathsf{CM.Commit}\left(\mathsf{ck}_0, \mathsf{ckL}_{\vec{i}}, \mathsf{ckR}_{\vec{i}}; \mathsf{mL}_{\vec{i}}, \mathsf{ckR}_{\vec{i}}\right) \\
&= 2^\ell \cdot \mathsf{CM.Commit}\left(\mathsf{ck}_0, \mathsf{ckL}, \mathsf{ckR}; \mathsf{mL}, \mathsf{mR}\right) = 2^\ell \cdot \mathsf{cm} \quad .
\end{aligned}
$$

The equality between the penultimate and final lines follows from the extendability of the commitment scheme.

Knowledge soundness. If CM is invertible then Protocol 4 is knowledge sound. Informally, a commitment scheme is invertible if there exists an efficient algorithm that, given a commitment $\mathsf{cm} \in \mathbb{C}$ after i-th level of recursion, and a suitable number of openings $(\mathsf{mL}', \mathsf{mR}') \in \mathbb{M}^{n/2^{\ell}} \times \mathbb{M}^{n/2^{\ell}}$ of the commitment cm' at the $(i+1)$-th level, corresponding to distinct challenges r, can find an opening of cm with respect to the commitment key of the i-th level. The proof of Theorem 3 relies on a similar strategy to that described in Section 2.1.4, but may use computational assumptions to prove invertibility. We remark that invertibility is incomparable with the binding property of the bilinear commitment scheme, since in Section 2.1.4, the Pedersen commitment scheme can be inverted without any computational assumptions, whereas inverting the commitment scheme of Section 2.2.1 requires the hardness of the discrete logarithm problem.

2.3 Extending sumcheck arguments to modules over rings

We have so far discussed sumcheck arguments for bilinear commitments where the message space, key space, and commitment space are vector spaces over a finite field $\mathbb{F}$. However, split-and-fold techniques have appeared in other cryptographic settings, such as groups of unknown order [BFS20] and lattices [BLNS20]. Sumcheck arguments can be generalized to capture these settings as well. We explain this extension in several steps. First, we extend the definition of bilinear commitments to more general algebraic structures and to valid openings of particular form. Second we show how this definition suffices for capturing the additional cryptographic settings. Third we show the necessary changes in the sumcheck argument and argue for its correctness and soundness.

The most natural algebraic structures for sumcheck arguments that suffice for our results are rings and modules. More specifically, the message space $\mathbb{M}$, key space $\mathbb{K}$, and commitment space $\mathbb{C}$ are modules over a ring R, and challenges in the protocol are sampled from a subset $\mathcal{C} \subseteq R$. The properties that this subset needs to satisfy are related to knowledge soundness and are explained below. Additionally, valid openings to a commitment are only "small" elements, so the underlying ring and message space must be equipped with a norm. The commit function is a mapping $\mathsf{CM.Commit}\colon \mathbb{M}^n \times \mathbb{K}^n \to \mathbb{C}$.

We highlight the required changes using the example of Pedersen commitments, which we show how to define in general settings in Section 2.4. The same changes apply to the sumcheck argument for any bilinear commitment. In the case of Pedersen commitments, $\mathsf{CM.Commit}\left(\vec{a}, \vec{\mathsf{G}}\right) = \langle \vec{a}, \vec{\mathsf{G}} \rangle$, where $\vec{a}$ is the message vector, $\vec{\mathsf{G}}$ is the key vector, and $\mathbb{M}$ and $\mathbb{K}$ are such that the mapping $\langle \cdot, \cdot \rangle$ is well-defined. To motivate the changes in sumcheck arguments, let us start by reviewing the Pedersen commitment scheme in the RSA group and lattice setting.

- In the RSA setting, $\vec{a}$ is a vector in $\mathbb{Z}^n_{<q}$ for some prime q and $\vec{\mathsf{G}} = (\mathsf{G}, \mathsf{G}^q, \dots, \mathsf{G}^{q^{n-1}})$ where G is a random element of the RSA group. The underlying ring in this case is $\mathbb{Z}$, the message space is also a subset of $\mathbb{Z}$, and the commitment and key spaces are equal to a group $\mathbb{G}$. Valid openings consist of elements with norm less than q.

- In the lattice setting, $\vec{a}$ is a vector of "short" ring elements and $\vec{\mathsf{G}}$ is a matrix of random ring elements. The most widely-used ring is $R = \mathbb{Z}_q[X]/\langle X^d + 1 \rangle$ and "short" ring elements belong to the set $\mathbb{Z}_{\leq B_{\mathsf{SIS}}}[X]/\langle X^d + 1 \rangle$ for some bound B_{SIS}. In this case, message, key, and commitment spaces are equal to the ring R. Valid openings consist of "short" ring elements with norm less than B_{SIS}.

We rewrite Protocol 4 for the case of Pedersen commitments incorporating the properties required for capturing disparate cryptographic settings.

Protocol 5: generalised sumcheck argument for Pedersen commitments (Π'_{SA})

For $n = 2^\ell$, the prover and verifier receive as input a commitment key $\vec{\mathsf{G}} \in \mathbb{K}^n$ and commitment $\mathsf{C} \in \mathbb{C}$. The prover also receives as input an opening $\vec{a} \in \mathbb{M}^n$ such that $\mathsf{C} = \langle \vec{a}, \vec{\mathsf{G}} \rangle$.

For $H := \{-1, 1\} \subseteq R$, the prover and verifier engage in a sumcheck round for the claim

$$\sum_{\omega_1,\ldots,\omega_\ell \in H} p_{\vec{a}}(\omega_1, \ldots, \omega_\ell) \cdot p_{\vec{\mathsf{G}}}(\omega_1, \ldots, \omega_\ell) = \mathsf{C} \ .$$

In other words, if $\ell = 0$ then the prover sends $p_a \in \mathbb{M}$ to the verifier, and the verifier checks that $\|p_a\| \leq B_{\mathsf{SA}}$ and uses $p_{\mathsf{G}} \in \mathbb{K}$ to check if $p_a \cdot p_{\mathsf{G}} = \mathsf{C}$ as claimed. If $\ell > 0$ then the interactive reduction works as follows.

- The prover $\mathbf{P}$ sends the polynomial $q(X) \in \mathbb{K}[X]$ to the verifier, computed as follows:

$$q(X) := \sum_{\omega_2,\ldots,\omega_\ell \in H} p_{\vec{a}}(X, \omega_2, \ldots, \omega_\ell) \cdot p_{\vec{\mathsf{G}}}(X, \omega_2, \ldots, \omega_\ell) \ . \tag{3}$$

- The verifier $\mathbf{V}$ samples $r \leftarrow \mathcal{C}$ and sends r to the prover.
- The verifier checks that $\mathsf{C} = \sum_{\omega \in H} q(\omega)$. (If not, it rejects.)
- The verifier outputs the new commitment key $\vec{\mathsf{G}}' \in \mathbb{K}^{n/2}$, which consists of the coefficients of $p_{\vec{\mathsf{G}}}(r, X_2, \ldots, X_\ell) \in \mathbb{K}[X_2, \ldots, X_\ell]$, and the new commitment $\mathsf{C}' := q(r) \in \mathbb{C}$. The prover outputs the new opening $\vec{a}' \in \mathbb{M}^{n/2}$ that as the coefficients of $p_{\vec{a}}(r, X_2, \ldots, X_\ell) \in \mathbb{M}[X_2, \ldots, X_\ell]$. The new sumcheck claim about these is:

$$\sum_{\omega_2,\ldots,\omega_\ell \in H} p_{\vec{a}'}(\omega_2, \ldots, \omega_\ell) \cdot p_{\vec{\mathsf{G}}'}(\omega_2, \ldots, \omega_\ell) = \mathsf{C}' \ .$$

Remark 2.3. Note that Protocol 5 follows the syntax of Protocol 4 instead of Protocol 2. The extension to general bilinear commitments is direct. However, the updates related to spaces and norm checks can be easily incorporated directly in Protocol 2 as well. This results in a sumcheck argument for generalised Pedersen commitments analogous to [BLNS20] in different cryptographic settings.

Completeness. The check $\|p_a\| \leq B_{\mathsf{SA}}$ is satisfied if the constant B_{SA} is set to be an upper bound on the norm of $p_{\vec{a}}(\vec{r})$. The constant B_{SA} can be computed based on n, the maximum norm of elements of $\mathcal{C}$, and the message space $\mathbb{M}$. For the other checks, the proof of completeness of Section 2.3 is still applicable.

Knowledge soundness. Similarly to the discrete logarithm case, given enough accepting transcripts it is possible to extract an opening of the commitment. In contrast to the previous protocols, random challenges need to be sampled from a set of low-norm elements which additionally satisfy a special invertibility property. We call sets which satisfy this invertibility property *sampling sets*. This allows us to recover a "relaxed" opening of the commitment, which differs from a regular opening in two ways: (a) the extracted opening might have larger norm than the committed value, (b) it might not satisfy the commitment equation, but only a closely related equation. In the case of Pedersen commitments, the extracted value is an opening for a small multiple of the original commitment. Note that this opening is not equivalent to an original opening because of the low-norm requirement. Theorem 3 is updated to include these changes as follows.

Theorem 4. *Protocol 5 is an opening protocol for the Pedersen commitment scheme* CM *in cryptographic settings where* CM *is invertible. Specifically, there exists an extractor that given a commitment* C *for a message with norm less than* B_{C}*, a commitment key* $\mathsf{ck} = \vec{\mathsf{G}}$*, a suitable tree of accepting transcripts, finds a relaxed valid opening* $\mathsf{m} = \vec{a} \in \mathbb{M}$ *such that* $c \cdot \mathsf{C} = \langle \vec{a}, \vec{\mathsf{G}} \rangle$*. The norm of* m *can be larger than* B_{C}*, but since it is a valid opening it has to be smaller than the norm bound specified in the commitment scheme.*

The slackness factor c depends on the setting. Out of all the settings that we consider, $c \neq 1$ only in the lattice setting, where we recover the result of [BLNS20]. Similarly to Section 2.2.2, we argue knowledge soundness through invertibility. As before, a commitment scheme is invertible if there exists an efficient algorithm that, given the commitment $\mathsf{C} \in \mathbb{C}$ of level i and a suitable collection of *relaxed* openings $(\vec{a}', \vec{\mathsf{G}}') \in \mathbb{M}^{n/2^\ell} \times \mathbb{K}^{n/2^\ell}$ of the commitment C' of the $i+1$-level of recursion for distinct challenges r, can find a *relaxed* opening (with possibly different slackness) of C with respect to the key of the i-th level. The opening of C might have larger norm than the relaxed openings $\vec{a}'$; in particular, invertibility is parametrized by a constant D such that if $\|\vec{a}'\| \leq B$, then the opening of level i has norm less than $D \cdot B$. Also, the slackness factor of the i-th level opening might be larger; specifically, invertibility is parametrized by a constant C such that if the slackness in level $i+1$ is c, then the slackness in level i is $C \cdot c$.

2.4 Instantiations of bilinear commitments for sumcheck arguments

Our main theorem on sumcheck arguments (Theorem 1) applies to any bilinear commitment that is invertible. In this section we explain how to define a *generalized Pedersen commitment* over any *bilinear module*, and how this commitment can be shown to satisfy the required properties for several cryptographic settings: (i) discrete logarithms; (ii) pairings; (iii) GUO; and (iv) lattices. In this paper we also consider generalized commitments over bilinear modules that capture scalar products, which we discuss in Section 2.5. Details on all instantiations of bilinear commitments that we study can be found in Section 5.

A bilinear module $\mathcal{M} = (R, M_L, M_R, M_T, e)$ consists of a normed ring R, three modules M_L, M_R, M_T over R, and a non-degenerate bilinear map $e\colon M_L \times M_R \to M_T$; the modules M_L, M_R and M_T are related to the message space, key space, and commitment space of the Pedersen commitments scheme. A bilinear module $\mathcal{M}$ is *argument-friendly* if M_L is equipped with a norm, and $\mathcal{M}$ is also associated with a sampling set $\mathcal{C} \subseteq R$ and a constant $B_{\max}$ (which indicates the norm bound for valid commitment openings).

Definition 2.4 (informal)**.** *The* **generalized Pedersen commitment scheme** *for messages of length* n *has messages of the form* $\vec{a} \in M_L^n$ *such that* $\|\vec{a}\| \leq B_{\max}$*, and commitment keys of the form* $\vec{\mathsf{G}} \in M_R^n$*. A commitment is computed as* $e(\vec{a}, \vec{\mathsf{G}})$*, which for notational simplicity we denote as* $\langle \vec{a}, \vec{\mathsf{G}} \rangle$*.*

In the hiding version, the commitment key has an extra component $\vec{\mathsf{G}}_0 \in M_R^{r_{\mathsf{Ped}}}$ *and the commitment is computed as* $\langle \vec{a}, \vec{\mathsf{G}} \rangle + \langle \vec{\rho}, \vec{\mathsf{G}}_0 \rangle$*, where* $\vec{\rho} \in M_L^{r_{\mathsf{Ped}}}$ *is sampled such that* $\|\vec{\rho}\| \leq B_{\max}$*, for a parameter* r_{Ped} *that depends on the setting. An opening with slackness* $c \in R$ *for a commitment* $\mathsf{C} \in M_T$ *under the commitment key* $(\vec{\mathsf{G}}, \vec{\mathsf{G}}_0) \in M_R^n \times M_R^{r_{\mathsf{Ped}}}$ *is a vector* $(\vec{a}, \vec{\rho}) \in M_L^n \times M_L^{r_{\mathsf{Ped}}}$ *such that* $c \cdot \mathsf{C} = \langle \vec{a}, \vec{\mathsf{G}} \rangle + \langle \vec{\rho}, \vec{\mathsf{G}}_0 \rangle$*.*

The generalized Pedersen commitment scheme is binding under the *bilinear relation assumption*, a natural extension of the discrete logarithm assumption. Moreover, its bilinearity holds unconditionally, and can be argued in a similar way as for the usual Pedersen commitment scheme (over prime-order groups).

Establishing invertibility, however, is more involved because we cannot establish it "once and for all" for all bilinear modules; rather, we establish it individually for each cryptographic setting of interest. One intuitive reason for why invertibility is more involved to establish is that it is incomparable to the binding property of a bilinear commitment scheme. In some cases it holds unconditionally (e.g., for the discrete logarithm setting as described in Section 2.1.4) and in other cases it follows from specific hardness assumptions (as previewed in Section 2.3).

To highlight the complications of each cryptographic setting, rather than specifically discussing invertibility, in this high level overview we describe the extraction algorithm for the Schnorr protocol for each cryptographic setting. This protocol is a simple zero-knowledge argument of knowledge for a commitment opening, and the extractor is asked to produce a (possibly relaxed) opening for the commitment given two accepting transcripts sharing the same first message.

Definition 2.5 (informal)**.** *In the* **Schnorr protocol** *for bilinear modules, the prover and verifier have a commitment* $\mathsf{C} \in M_T$ *and commitment key* $\mathsf{G} \in M_R$*, while the prover additionally has a witness* $a \in M_L$ *with* $\|a\| \leq B_{\max}$ *such that* $\mathsf{C} = a \cdot \mathsf{G}$*. The prover and verifier then interact as follows:*

- *the prover sends* $t = b \cdot \mathsf{G} \in M_T$ *for random* $b \in M_L$ *with* $\|b\| \leq B_b$ *(here* B_b *depends on* $B_{\max}$*);*
- *the verifier sends a random challenge* $r \in \mathcal{C}$*;*
- *the prover sends the response* $s := r \cdot a + b \in M_T$*;*
- *the verifier accepts if* $s \cdot \mathsf{G} = r \cdot \mathsf{C} + t$*.*

DL setting. The extractor recovers openings of C and t given two accepting transcripts (t, r_1, s_1) and (t, r_2, s_2) sharing the same first message t but with distinct challenges r_1 and r_2. First, subtracting one copy of the verification equation from the other shows that $(s_1 - s_2) \cdot \mathsf{G} = (r_1 - r_2) \cdot \mathsf{C}$. Then, since the challenges r_1 and r_2 are distinct elements of $\mathbb{F}_q$, we can multiply by $(r_1 - r_2)^{-1}$ to recover an opening $a' := \frac{(s_1 - s_2)}{(r_1 - r_2)}$ of C. The opening of t is then computed as $b' := s_1 - a' \cdot r_1$. In this setting, all $\mathbb{F}_q$-vectors are valid openings, so there is no special consideration regarding norms.

Pairing setting. Invertibility of the commitment scheme is similar to the discrete logarithm setting, in that all pairwise differences of distinct challenges are invertible.

GUO setting. Here invertibility will rely on cryptographic assumptions, similar to [BFS20]. Because the order of the group is unknown it is not possible to compute $(r_1 - r_2)^{-1}$. However, finding $(s_1 - s_2)$ that is not divisible by $(r_1 - r_2)$ breaks a cryptographic assumption known as the *Fractional Root Assumption*. Then, if $\frac{(s_1 - s_2)}{(r_1 - r_2)} \in \mathbb{Z}$, we can compute $\frac{(s_1 - s_2)}{(r_1 - r_2)} \cdot \mathsf{G}$. It must be that $\mathsf{C} = \frac{s_1 - s_2}{r_1 - r_2} \cdot \mathsf{G}$, since otherwise $(r_1 - r_2)(\mathsf{C} - \frac{s_1 - s_2}{r_1 - r_2} \cdot \mathsf{G}) = 0$, which would break the *Order Assumption*.

Another consideration in this setting is the choice of the sampling set and how it affects the size of the extracted commitment openings. The challenges are sampled from the set $\mathcal{C} = [-\frac{p-1}{2}, \frac{p-1}{2}]$ for a small prime p. Then, if $\|s_1\|, \|s_2\| \leq B$, the extracted opening of C, which is equal to $a' = \frac{(s_1 - s_2)}{(r_1 - r_2)}$, has norm less than $2 \cdot B$, and the extracted opening of t, which is equal to $b' = s_1 - a' \cdot r_1 = \frac{s_1 r_2 - s_2 r_1}{r_1 - r_2}$, has norm less than $(p-1) \cdot B$. This means that p must be chosen appropriately, because openings are valid only if their norm is less than q.

Lattice setting. Similar considerations relating to extracted openings exist in the lattice setting. In this setting there is no a priori bound on the norm of $(r_1 - r_2)^{-1}$ even if r_1 and r_2 have small norm. Because valid openings must be ring elements with small norm, we cannot recover an opening of C by multiplying by $(r_1 - r_2)^{-1}$. Instead we use a special sampling set introduced in [BCKLN14], which contains ring elements with norm 1 and has norm bounds for "relaxed" inverses. In particular, for $r_1, r_2 \in \mathcal{C}$ it holds that $2(r_1 - r_2)^{-1}$ is an element of norm 1. This relaxation factor translates into a slackness of the commitment opening. Namely, instead of extracting a' and b' such that $\mathsf{C} = a' \cdot \mathsf{G}$ and $t = b' \cdot \mathsf{G}$, we have that $2 \cdot \mathsf{C} = a' \cdot \mathsf{G}$ and $2 \cdot t = b' \cdot \mathsf{G}$.

The norm bounds on the extracted openings are easily recovered. If $\|s_1\|, \|s_2\| \leq B$, the extracted opening of C, which is equal to $a' = 2\frac{s_1 - s_2}{r_1 - r_2}$, has norm less than $2 \cdot B$, and the extracted opening of t, which is equal to $b' = 2s_1 - a' \cdot r_1 = 2\frac{s_1 r_2 - s_2 r_1}{r_1 - r_2}$, has norm less than $2 \cdot B$.

2.5 Succinct argument for scalar products over rings

We explain how to use sumcheck arguments to obtain zero-knowledge succinct arguments of knowledge for scalar-product relations over rings. Informally, this involves choosing a specific bilinear commitment to plug in to Theorem 1, and also carefully using randomness to achieve zero knowledge (which is not a guarantee of Theorem 1). Afterwards, in Section 2.6 we explain how to build on this to prove Theorem 2.

Fix a bilinear module $\mathcal{M} = (R, M_L, M_R, M_T, e)$ where M_L is not merely an R-module but also a ring itself (so that scalar products over M_L are defined).[7] The commitment scheme that we consider has two-part messages and includes a commitment to the scalar-product of these two parts; it is the natural extension of the scalar-product commitment from Section 2.2.1 to bilinear modules.

Definition 2.6 (informal)**.** *The* **generalized scalar-product commitment scheme** *for messages of length n has messages of the form $(\vec{a}, \vec{b}) \in M_L^n \times M_L^n$ such that $\|\vec{a}\|, \|\vec{b}\| \leq B_{\max}$, and commitment keys of the form $(\vec{\mathsf{G}}, \vec{\mathsf{G}}_0, \vec{\mathsf{H}}, \vec{\mathsf{H}}_0, \mathsf{U}, \vec{\mathsf{U}}_0) \in M_R^{n+r_{\mathsf{Ped}}} \times M_R^{n+r_{\mathsf{Ped}}} \times M_R^{1+r_{\mathsf{Ped}}}$. A commitment is computed as*

$$\Big(\langle \vec{a}, \vec{\mathsf{G}} \rangle + \langle \vec{\rho}_a, \vec{\mathsf{G}}_0 \rangle,\ \langle \vec{b}, \vec{\mathsf{H}} \rangle + \langle \vec{\rho}_b, \vec{\mathsf{H}}_0 \rangle,\ \langle \vec{a}, \vec{b} \rangle \cdot \mathsf{U} + \langle \vec{\rho}_t, \vec{\mathsf{U}}_0 \rangle \Big) \quad ;$$

where $\vec{\rho}_a, \vec{\rho}_b$ and $\vec{\rho}_t \in M_L^{r_{\mathsf{Ped}}}$ are sampled with norms at most $B_{\max}$, for a parameter r_{Ped} that depends on the setting. In other words, a commitment is the sum of three generalized Pedersen commitments: for the first part of the message $\vec{a}$, for the second part of the message $\vec{b}$, and for their scalar product $\langle \vec{a}, \vec{b} \rangle \in M_L$.

A valid opening for a commitment $\mathsf{C} \in M_T$ with keys $(\vec{\mathsf{G}}, \vec{\mathsf{G}}_0, \vec{\mathsf{H}}, \vec{\mathsf{H}}_0 \mathsf{U}, \vec{\mathsf{U}}_0) \in M_R^{n+r_{\mathsf{Ped}}} \times M_R^{n+r_{\mathsf{Ped}}} \times M_R^{1+r_{\mathsf{Ped}}}$ and slackness $c \in R$ is a vector $(\vec{a}, \vec{b}, \vec{\rho}_a, \vec{\rho}_b, \vec{\rho}_t) \in M_L^n \times M_L^n \times M_L^{3r_{\mathsf{Ped}}}$ such that

$$c^2 \cdot \mathsf{C} = \Big(c \cdot \langle \vec{a}, \vec{\mathsf{G}} \rangle + c \cdot \langle \vec{\rho}_a, \vec{\mathsf{G}}_0 \rangle,\ c \cdot \langle \vec{b}, \vec{\mathsf{H}} \rangle + c \cdot \langle \vec{\rho}_b, \vec{\mathsf{H}}_0 \rangle,\ \langle \vec{a}, \vec{b} \rangle \cdot \mathsf{U} + \langle \vec{\rho}_t, \vec{\mathsf{U}}_0 \rangle \Big) \quad .$$

The generalized scalar-product commitment scheme is binding under the bilinear relation assumption. Moreover, its bilinearity holds unconditionally. The proof of invertibility for the different cryptographic settings follows from algebraic manipulations analogous to the case of generalized Pedersen commitments discussed in Section 2.4. Witness extraction in this case requires computational assumptions even in the discrete logarithm setting (as discussed in Section 2.2.1).

We give a zero-knowledge succinct argument of knowledge for the following relation related to the scalar-product of committed messages, which we denote by $\mathcal{R}_{\text{comSP}}$.

Definition 2.7 (informal)**.** *The committed scalar-product relation $\mathcal{R}_{\text{comSP}}(c, B_{\mathsf{C}})$ consists of instance-witness pairs $(\mathbb{x}, \mathbb{w})$ where the instance $\mathbb{x}$ contains*

- *an argument-friendly bilinear module $\mathcal{M} = (R, M_L, M_R, M_T, e, \mathcal{C}, B_{\mathsf{C}}, B_{\max})$, where $\mathcal{C}$ is a sampling set,*
- *an ideal $I \subseteq M_L$,*[8]
- *commitment keys $(\vec{\mathsf{G}}, \vec{\mathsf{G}}_0, \vec{\mathsf{H}}, \vec{\mathsf{H}}_0, \mathsf{U}, \vec{\mathsf{U}}_0) \in M_R^n \times M_R^{r_{\mathsf{Ped}}} \times M_R^n \times M_R^{r_{\mathsf{Ped}}} \times M_R \times M_R^{r_{\mathsf{Ped}}}$, and*
- *commitments $\mathsf{C}_0, \mathsf{C}_1, \mathsf{C}_{\text{SP}} \in M_T$*

[7] In the pairing setting where M_L is not a ring, we define scalar-product commitments slightly differently. We refer to Section 5 for more details.

[8] An ideal $I \subseteq R$ is a subset which is a group under addition, and closed under multiplication by scalars in R; in other words, a subset of R which is also a submodule.

and the witness $\mathbb{w} = (\vec{a}, \vec{\rho}_0, \vec{b}, \vec{\rho}_1, \vec{\rho}) \in M_L^{2n+3r_{\mathsf{Ped}}}$ *is such that* $\|\vec{a}\|, \|\vec{\rho}_0\|, \|\vec{b}\|, \|\vec{\rho}_1\|, \|\vec{\rho}\| \le B_{\mathsf{C}}$ *and*

- $(\vec{a}, \vec{\rho}_0)$ *is a valid opening of the Pedersen commitments* C_0 *with slackness c,*
- $(\vec{b}, \vec{\rho}_1)$ *is a valid opening of the Pedersen commitments* C_1 *with slackness c, and*
- $(\langle \vec{a}, \vec{b} \rangle \bmod I, \vec{\rho})$ *is a valid opening of the Pedersen commitment* C_{SP} *with slackness* c^2*.*

The relation reasons about scalar-product relations over the quotient ring $R_* = M_L/I$, or modulo I, for some *ideal* $I \subseteq M_L$. In certain settings, such as the lattice setting, where it is only possible to extract openings to commitments with slackness c, we will choose I so that we can "cancel out" the slackness c modulo I as part of knowledge extraction algorithms and prove exact scalar-product relations over R_*.

Now we explain how the argument works. The prover begins by making a commitment to $\langle \vec{a}, \vec{b} \rangle$. The argument consists of: (i) a consistency check that the committed values $\langle \vec{a}, \vec{b} \rangle$ and $\langle \vec{a}, \vec{b} \rangle \bmod I$ (which may not be equal) are equivalent modulo I; (ii) an interactive reduction masking the three Pedersen commitments to $\vec{a}$, $\vec{b}$ and $\langle \vec{a}, \vec{b} \rangle$, and then converting them into a single scalar-product commitment by using various random linear combinations; (iii) a sumcheck argument to prove knowledge of an opening to the scalar-product commitment.

Checking consistency modulo *I*. To check consistency modulo I, we run a Schnorr-like protocol on the commitments to $\langle \vec{a}, \vec{b} \rangle$ and t. As part of the Schnorr-protocol, the prover will commit to a masking value y_C and send the commitment and the value $y_C \bmod I$ to the verifier, and after receiving a random challenge $\gamma \in \mathcal{C}$ from the verifier, the prover will send a value $e_C = \gamma(\langle \vec{a}, \vec{b} \rangle - t) + y_C$ to the verifier. The verifier may then check that $e_C = y_C \bmod I$. Intuitively, if $\langle \vec{a}, \vec{b} \rangle$ is not equal to t modulo I, then there is only a small probability that $\gamma(\langle \vec{a}, \vec{b} \rangle - t) + y_C$ can be equal to $y_C \bmod I$, which was fixed before the verifier sent the random challenge γ. To prove this intuition, we require that the challenge space $\mathcal{C}$ must remain a sampling set modulo I and the relaxation factor in the soundness guarantee for opening commitments must be invertible modulo I, which holds for all of our instantiations.

The interactive reduction. To make the argument zero-knowledge, the prover and verifier first rerandomise the commitments to $\vec{a}$, $\vec{b}$ and $\langle \vec{a}, \vec{b} \rangle$ to $\vec{z}_a := \gamma\vec{a} + \vec{y}_a$, $\vec{z}_b := \gamma\vec{b} + \vec{y}_b$ and $\langle \vec{z}_a, \vec{z}_b \rangle = \gamma^2 \langle \vec{a}, \vec{b} \rangle + \gamma v_1 + v_0$ using masking values $\vec{y}_a$ and $\vec{y}_b$. The prover commits to $\vec{y}_a$, $\vec{y}_b$ and to v_1 and v_0 (which depend only on $\vec{a}, \vec{b}, \vec{y}_a$ and $\vec{y}_b$), in advance. The openings of the rerandomised commitments do not leak any information about $\vec{a}$ or $\vec{b}$, and so the prover can safely send the commitment randomness to the verifier.

Next, the verifier uses the commitments to $\vec{z}_a$, $\vec{z}_b$ and $\langle \vec{z}_a, \vec{z}_b \rangle$ and the randomness to compute a single scalar-product commitment without any commitment randomness. Finally, the verifier runs a sumcheck argument on this scalar-product commitment.

The full details of the protocol are described in Section 6.

2.6 Succinct argument for R1CS over rings

We explain the main ideas behind Theorem 2, which provides a zero-knowledge succinct argument of knowledge for R1CS over rings. Recall that the R1CS problem over a ring R_* asks: given coefficient matrices $A, B, C \in R_*^{N \times N}$ and an instance vector x over R_*, is there a witness vector $\vec{w}$ over R_* such that $\vec{z} := (x, \vec{w}) \in R_*^N$ satisfies $A\vec{z} \circ B\vec{z} = C\vec{z}$? To a first order, Theorem 2 is proved by reducing the R1CS problem over R_* to several scalar-product sub-problems over R_*, and then relying on the zero-knowledge succinct argument for scalar products in Section 2.5. This means that we can ultimately support R1CS over the rings supported in that section: $R_* = M_L/I$, where M_L is the left module of an argument-friendly bilinear module, and $I \subseteq M_L$ is an ideal. Below we summarize the reduction from R1CS to scalar products.

The prover $\mathbf{P}$ sends commitments to the full assignment $\vec{z} \in R_*^N$ and to its linear combinations $\vec{z}_A, \vec{z}_B \in$

R_*^N. Then $\mathbf{P}$ is left to convince the verifier $\mathbf{V}$ that the committed information satisfies these conditions:

$$\vec{z}_A = A\vec{z} \quad , \quad \vec{z}_B = B\vec{z} \quad , \quad \vec{z}_A \circ \vec{z}_B = C\vec{z} \quad , \quad x \text{ is a prefix of } \vec{z} \ .$$

To reduce the first three conditions, the verifier $\mathbf{V}$ sends a structured challenge vector $\vec{r}$. Multiplying on the left by $\vec{r}^\intercal$ reduces the first three conditions to $\langle \vec{r}, \vec{z}_A \rangle = \langle \vec{r}_A, \vec{z} \rangle$, $\langle \vec{r}, \vec{z}_B \rangle = \langle \vec{r}_B, \vec{z} \rangle$, $\langle \vec{r} \circ \vec{z}_A, \vec{z}_B \rangle = \langle \vec{r}_C, \vec{z} \rangle$; here we defined $\vec{r}_A := \vec{r}^\intercal A$, $\vec{r}_B := \vec{r}^\intercal B$, and $\vec{r}_C := \vec{r}^\intercal C$. Moreover, to reduce the last condition, the verifier sends $\mathbf{V}$ a random challenge vector $\vec{s}$ of the same length as x; padding $\vec{s}$ with zeroes to get $\vec{s}'$ of the same length as $\vec{z}$, we have $\langle \vec{s}', \vec{z} \rangle = \langle \vec{s}, x \rangle$. Note that both parties can each individually compute $\vec{r}_A, \vec{r}_B, \vec{r}_C$ by right-multiplying $\vec{r}$ by A, B, C respectively, and also both parties can each individually compute $\langle \vec{s}, x \rangle$.

Next, the prover $\mathbf{P}$ sends a commitment to $\vec{z}_A' := \vec{r} \circ \vec{z}_A$, and also commitments to $\alpha := \langle \vec{r}_A, \vec{z} \rangle$, $\beta := \langle \vec{r}_B, \vec{z} \rangle$, and $\gamma := \langle \vec{r}_C, \vec{z} \rangle$. Then the prover and verifier engage in scalar-product sub-protocols (described in Section 2.5) to verify these 7 scalar products (recall each party can compute $\langle \vec{s}, x \rangle$):

$$\begin{matrix} \langle \vec{r}, \vec{z}_A \rangle = \alpha \\ \langle \vec{r}_A, \vec{z} \rangle = \alpha \end{matrix} \quad , \quad \begin{matrix} \langle \vec{r}, \vec{z}_B \rangle = \beta \\ \langle \vec{r}_B, \vec{z} \rangle = \beta \end{matrix} \quad , \quad \begin{matrix} \langle \vec{z}_A', \vec{z}_B \rangle = \gamma \\ \langle \vec{r}_C, \vec{z} \rangle = \gamma \end{matrix} \quad , \quad \langle \vec{s}', \vec{z} \rangle = \langle \vec{s}, x \rangle \ .$$

The prover and verifier use an additional challenge vector $\vec{y}$ and 2 further scalar-product sub-protocols to check that $\langle \vec{z}_A', \vec{y} \rangle = \langle \vec{z}_A, \vec{r} \circ \vec{y} \rangle$, which shows that $\vec{z}_A'$ was correctly computed from $\vec{z}_A$ and $\vec{r}$.

All commitments in the protocol are hiding, and hence do not leak any information about the witness vector $\vec{w}$. Hence the zero-knowledge property of the above protocol directly reduces to the zero-knowledge property of the scalar-product sub-protocols.

We conclude by noting that if we instantiate the bilinear module with lattices then Theorem 2 gives Corollary 1: a zero-knowledge succinct argument of knowledge for R1CS based on the SIS assumption.

2.7 Polynomial commitments over rings

As another direct application of the scalar-product protocol over rings, we construct a polynomial commitment scheme over rings. In a polynomial commitment, the prover commits to a polynomial p and then later proves the correct evaluation of the polynomial at a desired point. In our case, the committed polynomial is over a ring R and the evaluation is performed modulo I for an ideal $I \subseteq R$ as in Section 2.5.

Assume that the committed polynomial p is univariate in monomial basis, then p is represented as the vector of coefficients $\vec{a} = (a_0, \ldots, a_d)$ and $p(z) = \langle \vec{a}, \vec{b} \rangle$ where $\vec{b} = (1, z, z^2, \ldots, z^d)$. The polynomial commitment scheme consists of a binding commitment scheme for a message space $R[X]$ of polynomials over some ring R, which in our construction is the generalised Pedersen commitment over an argument-friendly bilinear module, and an interactive public-coin protocol between a prover and a verifier. The protocol convinces the verifier that for some values z and v it holds that $p(z) = v \bmod I$ for some ideal $I \subseteq R$. Based on the observation that the polynomial evaluation can be implemented as a scalar-product, the interactive evaluation protocol is the sumcheck protocol for generalised scalar-product relations.

Theorem 2.8 (informal)**.** *The generalized Pedersen commitment defines a polynomial commitment where the evaluation protocol reduces to a sumcheck argument for the scalar-product relation over rings. The evaluation protocol has round complexity $O(\log d)$, communication $O(\log d)$, and prover and verifier complexity $O(d)$, where d is the degree of the committed polynomial.*

Similar constructions apply to both univariate and multivariate polynomials which are represented in monomial or Lagrange basis.

3 Preliminaries

Notation. We denote rings with R, modules with M, fields with $\mathbb{F}$, and groups with $\mathbb{G}$. We use subscripts to define various rings, e.g. R_1, R_2. We use $\vec{x}$ to denote a vector.

3.1 Rings and modules

Rings are mathematical structures that generalise fields. A ring is equipped with addition and multiplication operations, but unlike fields, multiplicative inverses need not exist. In this paper, we will use only commutative rings, where the multiplication operation commutes. A module over a ring extends the notion of vector space over a field, where the scalars are elements of a ring.

Normed rings and modules. We define rings and modules equipped with norms. The following definitions are slightly different than the ones in standard algebra textbooks due to the expansion factors used, but are more convenient for our purposes.

Definition 3.1. *Let R be a ring. A* **norm for** *R is a map $\|\cdot\|_R : R \to \mathbb{R}_{\geq 0}$ that satisfies the following properties: (i) $\|0\|_R = 0$ and $\|1\|_R = 1$; (ii) for all $r \in R$, $\|r\|_R = \|-r\|_R$; (iii) for all $a, b \in R$, $\|a+b\|_R \leq \|a\|_R + \|b\|_R$; (iv) there exists a constant "expansion factor" $\gamma_R \in \mathbb{R}_{>0}$ such that, for all $a, b \in R$, $\|ab\|_R \leq \gamma_R \|a\|_R \|b\|_R$.*

Definition 3.2. *Let R be a ring with norm $\|\cdot\|_R$, and let M be an R-module. A* **norm for** *M is a map $\|\cdot\|_M : R \to \mathbb{R}_{\geq 0}$ that satisfies the following properties: (i) $\|0\|_M = 0$; (ii) for all $r \in R$, $\|r\|_M = \|-r\|_M$; (iii) for all $a, b \in M$, $\|a+b\|_M \leq \|a\|_M + \|b\|_M$; (iv) there exists a constant "expansion factor" $\gamma_M \in \mathbb{R}_{>0}$ such that, for all $a \in R$ and $b \in M$, $\|ab\|_M \leq \gamma_M \|a\|_R \|b\|_M$.*

Definition 3.3. *For a ring R with norm $\|\cdot\|_R$, $R(B)$ denotes the set of ring elements with norm at most B:*

$$R(B) := \{r \in R \ : \ \|r\|_R \leq B\} \ .$$

and similarly for a module M and set $M(B)$.

For a set $\mathcal{C} \subseteq R$, $m(\mathcal{C}) = \max_{x \in \mathcal{C}} \|x\|$.

Polynomials and challenge spaces over rings and modules. Let R be a ring and M an R-module. We denote by $M[X_1, \ldots, X_\ell]$ the set of polynomials in variables $X_1, \ldots, X_\ell$ with coefficients in M. A polynomial $p \in M[X_1, \ldots, X_\ell]$ defines a function from R^ℓ to M.

We consider sumcheck protocols for polynomials defined over general R-modules M, and must use a suitable challenge space $\mathcal{C} \subseteq R$. Prior work [CCKP19] discusses soundness of the sumcheck protocol over rings R, based on an extension of the Schwartz–Zippel lemma for challenge spaces which are "sampling sets". We state generalisations of the Schwartz–Zippel lemma and the notion of sampling sets to modules.

Definition 3.4. *A* **sampling set** *$\mathcal{C}$* **for an** *R*-**module** *M is a subset of R such that for all $c_1, c_2 \in \mathcal{C}$ with $c_1 \neq c_2$, the mapping $M \to M$ that sends $m \mapsto (c_1 - c_2)m$ is injective.*

We recover the sampling sets of [CCKP19] as a special case of Definition 3.4 by setting $M = R$.

Lemma 3.5. *Let R be a ring, let M be an R-module, and let $f \in M[X_1, \ldots, X_\ell]$ be a non-zero ℓ-variate polynomial of total degree D over R. Let $\mathcal{C}$ be a sampling set for M. Then $\Pr_{\vec{r} \leftarrow \mathcal{C}^\ell}[f(\vec{r}) = 0] \leq \frac{D}{|\mathcal{C}|}$.*

Sketch. The proof follows the same template as the usual inductive proof of the standard Schwartz–Zippel lemma. The properties of $\mathcal{C}$ are used to establish that a polynomial $f \in M[X]$ of degree D has at most D roots in $\mathcal{C}$ which is the base case and is also used in the inductive step. $\square$

The condition that Definition 3.4 is a sampling set for M is a sufficient condition to prove soundness of the sumcheck protocol over modules Protocol 6. We will prove soundness guarantees for our sumcheck arguments via a knowledge-extraction argument, and alongside Definition 3.4, require the following slightly stronger property from our challenge space.

Definition 3.6. *A* **strong sampling set** $\mathcal{C}$ **for an** R**-module** M *is a subset of* R *such that for all* $c_1, c_2 \in \mathcal{C}$ *with* $c_1 \neq c_2$*, there exists* $r \in R$ *(depending on* c_1 *and* c_2*) such that* $r(c_1 - c_2)m = m$ *for all* $m \in M$*.*

Again, setting $M = R$ recovers strong sampling sets for rings mentioned in [CCKP19].

It is easy to see that a strong sampling set $\mathcal{C}$ for M is also a sampling set for M. Conversely, for many rings, a sampling set is also a strong sampling set, as shown in the following lemma.

Lemma 3.7. *Let* R *be a finite commutative ring. If* $r \in R$ *is not a zero-divisor then* r *is invertible.*

Proof. The function $f\colon R \to R$ defined as $f(x) := r \cdot x$ is a ring homomorphism. The kernel of f contains only 0, as otherwise r would be a zero divisor. By the first Isomorphism Theorem for rings, we know that $R/\ker(f) \simeq \mathrm{Im}(f)$, and we can write $|R| = |\ker(f)| \cdot |\mathrm{Im}(f)|$. But $|\ker(f)| = 1$, so $|R| = |\mathrm{Im}(f)|$. Since R is finite and $\mathrm{Im}(f) \subseteq R$, we deduce that $\mathrm{Im}(f) = R$. Hence, there exists $s \in R$ such that $rs = 1$. $\square$

Note that Lemma 3.7 does not hold over infinite rings. For example, take $R = \mathbb{Z}$ and $r = 2$. (The proof breaks down because $\mathrm{Im}(f) = 2\mathbb{Z}$ has the same cardinality as $\mathbb{Z}$, but is a subset of $\mathbb{Z}$.)

3.2 Commitments

A (non-interactive) *commitment scheme* is a tuple of algorithms $\mathsf{CM} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Commit}, \mathsf{Open})$ with the following syntax.

- $\mathsf{CM.Setup}(1^\lambda, n) \to \mathsf{pp}$: samples public parameters given a security parameter and a message length. The public parameters specify a key space $\mathbb{K}_*$, message space $\mathbb{M}^n$, randomizer space $\mathbb{R}$, and commitment space $\mathbb{C}$.
- $\mathsf{CM.KeyGen}(\mathsf{pp}) \to \mathsf{ck}$: samples a commitment key (which itself contains a description of pp).
- $\mathsf{CM.Commit}(\mathsf{ck}; \mathsf{m}; \rho) \to \mathsf{cm}$: the sender commits to $\mathsf{m} \in \mathbb{M}^n$ using a commitment key $\mathsf{ck} \in \mathbb{K}_*$ by sampling ρ from $\mathbb{R}$ according to some distribution and computing a commitment $\mathsf{cm} \in \mathbb{C}$.
- $\mathsf{CM.Open}(\mathsf{ck}, \mathsf{m}, \rho, \mathsf{cm}, c) \to b \in \{0,1\}$: checks that $\mathsf{cm} \in \mathbb{C}$ is a commitment to the message $\mathsf{m} \in \mathbb{M}^n$ with randomness $\rho \in \mathbb{R}$ and slackness value $c \in \mathbb{Z}$ under commitment key $\mathsf{ck} \in \mathbb{K}_*$.

We require CM to satisfy completeness and binding, and sometimes also hiding, as specified below.

Definition 3.8. CM *is* **complete** *if for every* $n \in \mathbb{N}$ *and adversary* $\mathcal{A}$*,*

$$\Pr\left[\mathsf{CM.Open}(\mathsf{ck}, \mathsf{m}, \rho, \mathsf{cm}, 1) = 1 \;\middle|\; \begin{array}{r} \mathsf{pp} \leftarrow \mathsf{CM.Setup}(1^\lambda, n) \\ \mathsf{ck} \leftarrow \mathsf{CM.KeyGen}(\mathsf{pp}) \\ (\mathsf{m}, \rho) \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{ck}) \\ (\mathsf{cm}, \rho) \leftarrow \mathsf{CM.Commit}(\mathsf{ck}; \mathsf{m}; \rho) \end{array}\right] = 1 \ .$$

Definition 3.9. CM *is (computationally)* **binding** *if for every $n \in \mathbb{N}$ and polynomial-size adversary $\mathcal{A}$,*

$$\Pr\left[\begin{array}{c|r} \mathsf{m}_0 \neq \mathsf{m}_1 & \mathsf{pp} \leftarrow \mathsf{CM.Setup}(1^\lambda, n) \\ \mathsf{CM.Open}\left(\mathsf{ck}, \mathsf{m}_0, \rho_0, \mathsf{cm}, c\right) = 1 & \mathsf{ck} \leftarrow \mathsf{CM.KeyGen}(\mathsf{pp}) \\ \mathsf{CM.Open}\left(\mathsf{ck}, \mathsf{m}_1, \rho_1, \mathsf{cm}, c\right) = 1 & (\mathsf{cm}, \mathsf{m}_0, \mathsf{m}_1, \rho_0, \rho_1, c) \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{ck}) \end{array}\right] = \mathrm{negl}(\lambda) \enspace .$$

Definition 3.10. CM *is (statistically)* **hiding** *if for every $n \in \mathbb{N}$ and adversary $\mathcal{A}$,*

$$\Pr\left[\mathcal{A}(\mathsf{cm}) = b \;\middle|\; \begin{array}{r} \mathsf{pp} \leftarrow \mathsf{CM.Setup}(1^\lambda, n) \\ \mathsf{ck} \leftarrow \mathsf{CM.KeyGen}(\mathsf{pp}) \\ (\mathsf{m}_0, \mathsf{m}_1) \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{ck}) \\ b \leftarrow \{0,1\} \\ (\mathsf{cm}, \rho) \leftarrow \mathsf{CM.Commit}\left(\mathsf{ck}; \mathsf{m}_b\right) \end{array}\right] = \frac{1}{2} + \mathrm{negl}(\lambda) \enspace .$$

If we the above probability equals $1/2$ *then* CM *is* **perfectly hiding**.

3.3 Interactive arguments

We say that $\mathsf{ARG} = (\mathbf{G}, \mathbf{P}, \mathbf{V})$ is an *interactive argument* of knowledge for a relation $\mathcal{R}$ if it satisfies the following completeness and knowledge properties.

- *Completeness.* For every adversary $\mathcal{A}$,

$$\Pr\left[\begin{array}{c|r} (\mathbb{x}, \mathbb{w}) \notin \mathcal{R} \text{ or} & \mathsf{pp} \leftarrow \mathbf{G}(1^\lambda) \\ \langle \mathbf{P}(\mathsf{pp}, \mathbb{x}, \mathbb{w}), \mathbf{V}(\mathsf{pp}, \mathbb{x}) \rangle = 1 & (\mathbb{x}, \mathbb{w}) \leftarrow \mathcal{A}(\mathsf{pp}) \end{array}\right] = 1 \enspace .$$

- *Witness-extended emulation.* There exists an expected polynomial-time emulator $\mathcal{E}$ such that for every polynomial-size adversary $\mathcal{A}$ the following probabilities are $\mathrm{negl}(\lambda)$-close:

$$\Pr\left[\mathcal{A}(\mathrm{tr}) = 1 \;\middle|\; \begin{array}{r} \mathsf{pp} \leftarrow \mathbf{G}(1^\lambda) \\ (\mathbb{x}, s) \leftarrow \mathcal{A}(\mathsf{pp}) \\ \mathrm{tr} \leftarrow \langle \mathcal{A}(\mathsf{pp}, \mathbb{x}, s), \mathbf{V}(\mathsf{pp}, \mathbb{x}) \rangle \end{array}\right] \text{ and}$$

$$\Pr\left[\begin{array}{l|r} \mathcal{A}(\mathrm{tr}) = 1 & \mathsf{pp} \leftarrow \mathbf{G}(1^\lambda) \\ \text{if } \mathrm{tr} \text{ is accepting, then } (\mathbb{x}, \mathbb{w}) \in \mathcal{R} & (\mathbb{x}, s) \leftarrow \mathcal{A}(\mathsf{pp}) \\ & (\mathrm{tr}, \mathbb{w}) \leftarrow \mathcal{E}^{\mathcal{A}(\mathsf{pp}, \mathbb{x}, s)}(\mathsf{pp}, \mathbb{x}) \end{array}\right] \enspace .$$

  Above $\mathcal{E}$ has oracle access to (the next-message functions of) $\mathcal{A}(\mathsf{pp}, \mathbb{x}, s)$.

We also consider argument systems with a zero knowledge property.

- *Semi-honest-verifier (statistical) zero knowledge:* There exists a probabilistic polynomial-time simulator $\mathcal{S}$ such that for every interactive stateful adversary $\mathcal{A}$ the following probabilities are $\mathrm{negl}(\lambda)$-close:

$$\Pr\left[\begin{array}{c|r} (\mathbb{x}, \mathbb{w}) \in \mathcal{R} & \mathsf{pp} \leftarrow \mathbf{G}(1^\lambda) \\ \mathcal{A}(\mathrm{tr}) = 1 & (\mathbb{x}, \mathbb{w}, \rho) \leftarrow \mathcal{A}(\mathsf{pp}) \\ & (\mathrm{tr}, b) \leftarrow \langle \mathbf{P}(\mathsf{pp}, \mathbb{x}, \mathbb{w}), \mathbf{V}(\mathsf{pp}, \mathbb{x}; \rho) \rangle \end{array}\right] \text{ and } \Pr\left[\begin{array}{c|r} (\mathbb{x}, \mathbb{w}) \in \mathcal{R} & \mathsf{pp} \leftarrow \mathbf{G}(1^\lambda) \\ \mathcal{A}(\mathrm{tr}) = 1 & (\mathbb{x}, \mathbb{w}, \rho) \leftarrow \mathcal{A}(\mathsf{pp}) \\ & (\mathrm{tr}, b) \leftarrow \mathcal{S}(\mathsf{pp}, \mathbb{x}, \rho) \end{array}\right] .$$

  Above, ρ is the randomness used by the verifier.

3.3.1 Extraction from trees

We say that ARG is *public coin* if each verifier message is a uniform random string (of a prescribed length). The public-coin interactive arguments in this paper have the property that a witness can be extracted from an appropriate tree of accepting transcripts. The definition below is a natural generalization of special-soundness for Sigma-protocols (where $m = 1$ and $n_1 = 2$).

Definition 3.11. *Let* ARG *be a public-coin interactive argument for a relation* $\mathcal{R}$ *where the verifier sends* m *messages. For* $n_1, \ldots, n_m \in \mathbb{Z}$*, we say that* T *is a* $(n_1, \ldots, n_m)$**-tree of accepting transcripts for** $\mathbb{x}$ *if (1)* T *is a tree of depth* m *where, for each* $i \in [m]$*, each vertex at layer* i *has* n_i *children (so the tree has* $\prod_{i\in[m]} n_i$ *leaves); (2) the* n_i *outgoing edges of every vertex in layer* i *are labeled with* n_i *different choices of randomness for the verifier's* i*-th message; (3) each vertex in layer* i *is labeled with a prover message; (4) every path from the root to a leaf in the tree is an accepting transcript for the interactive argument.*

Definition 3.12. ARG *has* $(n_1, \ldots, n_m)$**-tree extraction** *if*

$$\Pr\left[\begin{array}{c} T \textit{ is a } (n_1, \ldots, n_m)\textit{-tree of accepting transcripts for } \mathbb{x} \\ (\mathbb{x}, \mathbb{w}) \notin \mathcal{R} \end{array} \middle| \begin{array}{c} \mathsf{pp} \leftarrow \mathbf{G}(1^\lambda) \\ (\mathbb{x}, T) \leftarrow \mathcal{A}(\mathsf{pp}) \\ \mathbb{w} \leftarrow \chi(\mathsf{pp}, \mathbb{x}, T) \end{array}\right] = \operatorname{negl}(\lambda) \ .$$

The following lemma from [BCCGP16] states that tree extraction implies witness-extended emulation. Throughout this paper we rely on this generic implication in that it will suffice for our technical statements to establish tree extraction for the protocols that we study.

Lemma 3.13. *Let* ARG *be a public-coin interactive argument where the verifier sends* m *messages. If* ARG *has* $(n_1, \ldots, n_m)$*-tree extraction and* $\prod_{i\in[m]} n_i = \mathsf{poly}(\lambda)$ *then* ARG *has witness-extended emulation.*

4 Sumcheck argument for opening a bilinear commitment

Throughout this section, we use multilinear polynomials whose coefficients are defined by a vector $\vec{v}$ as follows.

Definition 4.1. *Let R be a ring and M an R-module. For $n \in \mathbb{N}$ a power of 2, set $\ell := \log n$ and let $\vec{v} \in M^n$ be vector whose entries we index via binary strings $(i_1, \ldots, i_\ell) \in \{0,1\}^\ell$. The ℓ-variate polynomial $p_{\vec{v}} \in M[X_1, \ldots, X_\ell]$ is defined as follows:*

$$p_{\vec{v}}(X_1, \ldots, X_\ell) := \sum_{i_1,\ldots,i_\ell \in \{0,1\}} v_{i_1,\ldots,i_\ell} X_1^{i_1} \cdots X_\ell^{i_\ell} \ .$$

4.1 Summing polynomials over subgroups

We state a straightforward generalization of a lemma from [BCG20] concerning sums of polynomials; we rely on this in our sumcheck argument.

Lemma 4.2. *Let H be a cyclic subgroup of the multiplicative group of a ring R. Let M be an R-module and let $p(X_1, \ldots, X_\ell) \in M[X_1, \ldots, X_\ell]$ be a polynomial. If we denote by $p_{i_1,\ldots,i_\ell} \in M$ the coefficient of $X_1^{i_1} \cdots X_\ell^{i_\ell}$ in the polynomial $p(X_1, \ldots, X_\ell)$, then*

$$\sum_{\vec{\omega} \in H^\ell} p(\vec{\omega}) = \left(\sum_{\vec{i} \equiv \vec{0} \bmod |H|} p_{\vec{i}} \right) \cdot |H|^\ell \ . \tag{4}$$

In Section 4.3 we will obtain a scalar-product protocol based on the sumcheck protocol, by applying Lemma 4.2 with $H = \{-1, 1\}$ and p equal to a polynomial derived from multilinear polynomials encoding a message m and commitment key ck. In this case, p will be quadratic in each variable, and by the bilinear properties of the commitment scheme and the only term contributing to the right hand side of Equation (4) will be a multiple of the commitment $\mathsf{CM.Commit}\,(\mathsf{ck}; \mathsf{m})$.

4.2 Bilinear commitments

Suppose that $\mathbb{K}, \mathbb{M}, \mathbb{R}, \mathbb{C}$ are modules over a ring R, and additionally $\mathbb{M}$ has an associated norm. To define bilinear and extendable commitments, we require the commitment scheme to have a special form, where the commitment key can be "unrolled" to a large key. In particular, let $\mathsf{CM.ExtendCK}$ be an algorithm that takes as input a commitment key $\mathsf{ck} \in \mathbb{K}_*$ and outputs an element of $\mathbb{K}^n \times \mathbb{K}_0$ and $\mathsf{CM.BilinearCommit}$ be an algorithm such that for all commitment keys $\mathsf{ck} \in \mathbb{K}_*$, it holds that

$$\mathsf{CM.BilinearCommit}(\mathsf{CM.ExtendCK}(\mathsf{ck}); \mathsf{m}; \rho) = \mathsf{CM.Commit}\,(\mathsf{ck}; \mathsf{m}; \rho) \, .$$

We now define a bilinear property of commitments. Note that this does *not* imply standard homomorphic properties such as Definition 5.8.

Definition 4.3. CM *is* **bilinear** *if for all $\lambda \in \mathbb{N}$ and $n \in \mathbb{N}$, all* $\mathsf{ckL}, \mathsf{ckL}', \mathsf{ckR}, \mathsf{ckR}' \in \mathbb{K}^{n/2}$ *such that* $(\mathsf{ck}_0, \mathsf{ckL}, \mathsf{ckR}) = \mathsf{CM.ExtendCK}(\mathsf{ck})$, $(\mathsf{ck}_0, \mathsf{ckL}', \mathsf{ckR}') = \mathsf{CM.ExtendCK}(\mathrm{ck}')$ *for* $\mathrm{ck}, \mathrm{ck}' \leftarrow \mathsf{CM.KeyGen}(\mathsf{pp})$, *all* $(\mathsf{mL}, \mathsf{mL}', \mathsf{mR}, \mathsf{mR}') \in \mathbb{M}^{n/2}$, *all $\rho, \rho' \in \mathbb{R}$ and all $r \in R$, we have*

$$\mathsf{CM.BilinearCommit}(\mathsf{ck}_0, \mathsf{ckL} + \mathsf{ckL}', \mathsf{ckR}; \mathsf{mL} + \mathsf{mL}', \mathsf{mR}; \rho + \rho')$$

$$\begin{aligned}&= \mathsf{CM.BilinearCommit}(\mathsf{ck}_0, \mathsf{ckL}, \mathsf{ckR}; \mathsf{mL}, \mathsf{mR}; \rho)\\&+ \mathsf{CM.BilinearCommit}(\mathsf{ck}_0, \mathsf{ckL}', \mathsf{ckR}; \mathsf{mL}', \mathsf{mR}; \rho')\end{aligned}$$

$$\begin{aligned}&\mathsf{CM.BilinearCommit}(\mathsf{ck}_0, \mathsf{ckL}, \mathsf{ckR} + \mathsf{ckR}'; \mathsf{mL}, \mathsf{mR} + \mathsf{mR}'; \rho + \rho')\\&= \mathsf{CM.BilinearCommit}(\mathsf{ck}_0, \mathsf{ckL}, \mathsf{ckR}; \mathsf{mL}, \mathsf{mR}; \rho)\\&+ \mathsf{CM.BilinearCommit}(\mathsf{ck}_0, \mathsf{ckL}, \mathsf{ckR}'; \mathsf{mL}, \mathsf{mR}'; \mathsf{ck}_0, \rho')\end{aligned}$$

$$\begin{aligned}&r \cdot \mathsf{CM.BilinearCommit}(\mathsf{ck}_0, \mathsf{ckL}, \mathsf{ckR}; \mathsf{mL}, \mathsf{mR}; \rho)\\&= \mathsf{CM.BilinearCommit}(\mathsf{ck}_0, r \cdot \mathsf{ckL}, \mathsf{ckR}; r \cdot \mathsf{mL}, \mathsf{mR}; r \cdot \rho)\\&= \mathsf{CM.BilinearCommit}(\mathsf{ck}_0, \mathsf{ckL}, r \cdot \mathsf{ckR}; \mathsf{mL}, r \cdot \mathsf{mR}; r \cdot \rho)\end{aligned}$$

We consider polynomials of commitments, writing expressions such as "$\mathsf{CM.Commit}\left(p_{\mathsf{m}}(\vec{X}), p_{\mathsf{ck}}(\vec{X})\right)$". We explain what this notation means in more detail. In this case, $p(\vec{X})$ is not a "polynomial commitment" to $p_{\mathsf{m}}(\vec{X})$ and $p_{\mathsf{ck}}(\vec{X})$ in the sense of e.g. Section 8, but is obtained by treating $\vec{X}$ as a collection of formal variables in the commitment algorithm. This notation is well-defined when $\mathsf{CM.Commit}(\cdot, \cdot)$ is bilinear.

Lemma 4.4. *Let* $\mathsf{CM.BilinearCommit}\colon \mathbb{K}_0 \times \mathbb{K}^2 \times \mathbb{M}^2 \times \mathbb{R} \to \mathbb{C}$ *be a bilinear function (see Definition 4.3). Let* $\vec{X} = (X_1, \ldots, X_\ell)$ *be formal variables. There is a bilinear function* $\widetilde{\mathsf{Commit}}\colon \mathbb{K}_0 \times (\mathbb{K}[\vec{X}])^2 \times (\mathbb{M}[\vec{X}])^2 \times \mathbb{R} \to \mathbb{C}[\vec{X}]$ *such that for every* $\mathsf{ck}_0 \in \mathbb{K}_0$*,* $(p_1(\vec{X}), p_2(\vec{X}); p_3(\vec{X}), p_4(\vec{X})) \in (\mathbb{K}[\vec{X}])^2 \times (\mathbb{M}[\vec{X}])^2$*, and* $\vec{r} \in R^\ell$*, it holds that*

$$\mathsf{CM.BilinearCommit}(\mathsf{ck}_0, p_1(\vec{r}), p_2(\vec{r}); p_3(\vec{r}), p_4(\vec{r}); 0) = \widetilde{\mathsf{Commit}}\left(\mathsf{ck}_0, p_1, p_2; p_3, p_4; 0\right)(\vec{r}) \ . \qquad (5)$$

Proof. Without loss of generality, we may assume that each polynomial has the same support (can be expressed as a linear combination of the same set of monomials) by padding the polynomials with extra monomials with zero coefficients. Write $p_j(\vec{X}) = \sum_{\vec{i} \in I} p_{j,\vec{i}} \vec{X}^{\vec{i}}$ for $j = 1, 2, 3, 4$. Then we have

$$\begin{aligned}&\mathsf{CM.BilinearCommit}(\mathsf{ck}_0, p_1(\vec{r}), p_2(\vec{r}); p_3(\vec{r}), p_4(\vec{r}); 0)\\&= \sum_{\vec{i} \in I} \mathsf{CM.BilinearCommit}(\mathsf{ck}_0, p_{1,\vec{i}}, p_2(\vec{r}); p_{3,\vec{i}}(\vec{r}), p_4(\vec{r}); 0) \cdot \vec{r}^{\,\vec{i}}\\&= \sum_{\vec{i}, \vec{j} \in I} \mathsf{CM.BilinearCommit}(\mathsf{ck}_0, p_{1,\vec{i}}, p_{2,\vec{j}}; p_{3,\vec{i}}, p_{4,\vec{j}}; 0) \cdot \vec{r}^{\,\vec{i}+\vec{j}}\end{aligned}$$

So we define $\widetilde{\mathsf{Commit}}\left(\mathsf{ck}_0, p_1, p_2; p_3, p_4; 0\right)(\vec{X}) := \sum_{\vec{i}, \vec{j} \in I} \mathsf{CM.Commit}\left(\mathsf{ck}_0, p_{1,\vec{i}}, p_{2,\vec{j}}; p_{3,\vec{i}}, p_{4,\vec{j}}; 0\right) \cdot \vec{X}^{\vec{i}+\vec{j}}$. □

From now on, we simply use notation such as "$\mathsf{CM.Commit}\left(\mathsf{ck}_0, p_{\mathsf{ckL}}(\vec{X}), p_{\mathsf{ckR}}(\vec{X}); p_{\mathsf{mL}}(\vec{X}), p_{\mathsf{mR}}(\vec{X}); 0\right)$", with the understanding that this actually refers to an application of "$\widetilde{\mathsf{Commit}}$" via the correspondence established in Lemma 4.4.

4.3 Construction

Let CM be an extendable bilinear commitment scheme over a normed ring R. We present a variant of the sumcheck protocol, which, given a commitment $\mathsf{cm} \in \mathbb{C}$, proves knowledge of an opening $\mathsf{m} \in \mathbb{M}$ of cm with respect to a commitment key $\mathsf{ck} \in \mathbb{K} \times \mathbb{K}_0$ such that $\mathsf{cm} = \mathsf{CM.Commit}(\mathsf{ck}; \mathsf{m}; 0)$.

Using bilinear commitments. Our sumcheck protocol works by applying a sumcheck to the polynomial $p(\vec{X}) := \mathsf{CM.Commit}\left(\mathsf{ck}_0, p_{\mathsf{ckL}}(\vec{X}), p_{\mathsf{ckR}}(\vec{X}); p_{\mathsf{mL}}(\vec{X}), p_{\mathsf{mR}}(\vec{X}); 0\right)$, for $\mathsf{ckL}, \mathsf{ckR} \in \mathbb{K}^{n/2}$, $\mathsf{mL}, \mathsf{mR} \in \mathbb{M}^{n/2}$ and $\mathsf{ck}_0 \in \mathbb{K}_0$ which is defined over the commitment space.

By the extendability and bilinear properties of the commitment scheme, and Lemma 4.2, we have that

$$\sum_{\vec{\omega} \in \{-1,1\}^\ell} p(\vec{\omega}) = 2^\ell \cdot \mathsf{CM.Commit}\left(\mathsf{ck}_0, \mathsf{ckL}, \mathsf{ckR}; \mathsf{mL}, \mathsf{mR}; 0\right) \quad .$$

Our protocol works by applying a standard sumcheck protocol to the polynomial p, with one important difference. In the final step of the standard sumcheck protocol, the verifier would usually compute $p(\vec{r})$ for a random point $\vec{r}$. Note that in our setting, $p(\vec{r})$ *is a commitment*. Our protocol ends differently, with the *prover* sending an *opening* of $p(\vec{r})$ to the verifier. This is crucial in showing that the modified sumcheck protocol is a proof of knowledge.

Formally, we give a sumcheck protocol for the following relation.

Definition 4.5. *The relation $\mathcal{R}_{\text{SC}}(c, B_{\mathsf{C}})$ is the set of tuples*

$$(\mathbb{x}, \mathbb{w}) = \Big((\mathsf{CM}, \mathsf{pp}, \mathsf{ck}_0, \mathsf{ckL}, \mathsf{ckR}, \mathsf{cm}), (\mathsf{mL}, \mathsf{mR})\Big) \quad ,$$

where $\mathsf{CM} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Commit}, \mathsf{Open})$ is a bilinear commitment scheme over a ring R (see Definition 4.3), $\mathsf{pp} = (\mathbb{K}_0 \times \mathbb{K}^n, \mathbb{M}^n, \mathbb{C})$, $(\mathsf{ck}_0, \mathsf{ckL}, \mathsf{ckR}) \in \mathbb{K}_0 \times \mathbb{K}^{n/2} \times \mathbb{K}^{n/2}$, $\mathsf{cm} \in \mathbb{C}$, $(\mathsf{mL}, \mathsf{mR}) \in \mathbb{M}(B_{\mathsf{C}})^{n/2} \times \mathbb{M}(B_{\mathsf{C}})^{n/2}$, and it holds that $\mathsf{CM.Open}\left((\mathsf{ck}_0, \mathsf{ckL}, \mathsf{ckR}), (\mathsf{mL}, \mathsf{mR}), 0, \mathsf{cm}, c\right) = 1$.

Our protocol proceeds in rounds, where in each round the messages and keys are halved in size. To facilitate the protocol description and the subsequent proofs, we use the following definition for round keys and messages.

Definition 4.6 (round commitment keys and messages)**.** *Let $n = 2^\ell$. Let $\mathsf{ckL}, \mathsf{ckR} \in \mathbb{K}^n$ and let $\mathsf{mL}, \mathsf{mR} \in \mathbb{M}^n$. Let $r_1, \ldots, r_\ell \in R$. Define $\mathsf{ckL}^{(j)} \in \mathbb{K}^{2^{\ell-j}}$ to be the vector of coefficients of the polynomial $p_{\mathsf{ckL}}(r_1, \ldots, r_j, X_{j+1}, \ldots, X_\ell)$:*

$$p_{\mathsf{ckL}}(r_1, \ldots, r_j, X_{j+1}, \ldots, X_\ell) = \sum_{\vec{i}(>j) \in \{0,1\}^{\ell-j}} \mathsf{ckL}^{(j)}_{\vec{i}(>j)} \cdot X_{j+1}^{i_{j+1}} \cdots X_\ell^{i_\ell} \quad .$$

Note that the entries of $\mathsf{ckL}^{(j)}$ satisfy the recurrence relations

$$\mathsf{ckL}^{(j)}_{\vec{i}(>j)} = \mathsf{ckL}^{(j-1)}_{0,\vec{i}(>j)} + r_j \cdot \mathsf{ckL}^{(j-1)}_{1,\vec{i}(>j)} \quad ,$$

and that $\mathsf{ckL}^{(0)} = \mathsf{ckL}$, and $\mathsf{ckL}^{(\ell)} = p_{\mathsf{ckL}}(\vec{r})$. The vectors $\mathsf{ckR}^{(j)} \in \mathbb{K}^{2^{\ell-j}}$ and $\mathsf{mL}^{(j)}$, $\mathsf{mR}^{(j)} \in \mathbb{M}^{2^{\ell-j}}$ are defined similarly in terms of the polynomials $p_{\mathsf{ckR}}(r_1, \ldots, r_j, X_{j+1}, \ldots, X_\ell)$, $p_{\mathsf{mL}}(r_1, \ldots, r_j, X_{j+1}, \ldots, X_\ell)$, and $p_{\mathsf{mR}}(r_1, \ldots, r_j, X_{j+1}, \ldots, X_\ell)$, and satisfy similar relations.

Construction 4.7 (sumcheck argument)**.** We describe a public-coin interactive argument $\mathsf{SCA} = (\mathbf{P}, \mathbf{V})$ for the relation $\mathcal{R}_{\text{SC}}(1, B_{\mathsf{C}})$. The prover $\mathbf{P}$ takes as input an instance $\mathbb{x} = (\mathsf{CM}, \mathsf{pp}, \mathsf{ck}_0, \mathsf{ckL}, \mathsf{ckR}, \mathsf{cm})$ and a witness $\mathbb{w} = (\mathsf{mL}, \mathsf{mR})$. The verifier $\mathbf{V}$ takes as input the instance $\mathbb{x}$. The instance and witness jointly define the polynomial

$$p(\vec{X}) := \mathsf{CM.BilinearCommit}(\mathsf{ck}_0, p_{\mathsf{ckL}}(\vec{X}), p_{\mathsf{ckR}}(\vec{X}); p_{\mathsf{mL}}(\vec{X}), p_{\mathsf{mR}}(\vec{X}); 0) \quad .$$

The prover and verifier work as follows.

1. *Interaction.* For $i = 1, \ldots, \ell$:

   - The prover computes and sends $q_i(X_i) := 2^{-(\ell-i)} \sum_{\omega_{i+1},\ldots,\omega_\ell \in \{-1,1\}} p(r_1, \ldots, r_{i-1}, X_i, \omega_{i+1}, \ldots, \omega_\ell)$, which consists of three elements of $\mathbb{C}$.[9]
   - Verifier sends a random challenge $r_i \leftarrow \mathcal{C}$.

2. *Opening.* The prover computes the opening values $\mathsf{mL}^{(\ell)} = p_{\mathsf{mL}}(\vec{r})$ and $\mathsf{mR}^{(\ell)} = p_{\mathsf{mR}}(\vec{r})$ to the commitment $p(\vec{r})$, and sends $\mathsf{mL}^{(\ell)}, \mathsf{mR}^{(\ell)} \in \mathbb{M}$ to the verifier.

3. *Verification:* The verifier computes the commitment keys $\mathsf{ckL}^{(\ell)} = p_{\mathsf{ckL}}(\vec{r})$ and $\mathsf{ckR}^{(\ell)} = p_{\mathsf{ckR}}(\vec{r})$. Then the verifier checks that the following conditions hold:

   - $\left\|\mathsf{mL}^{(\ell)}\right\|_{\mathbb{M}}, \left\|\mathsf{mR}^{(\ell)}\right\|_{\mathbb{M}} \leq n \cdot (\gamma_R m(\mathcal{C}))^\ell \cdot B_{\mathsf{C}}$;
   - $q_1(1) + q_1(-1) = 2 \cdot \mathsf{cm}$;
   - for every $i \in \{2, \ldots, \ell\}$, $q_i(1) + q_i(-1) = 2 \cdot q_{i-1}(r_{i-1})$;
   - $\mathsf{CM.Open}\left((\mathsf{ck}_0, \mathsf{ckL}^{(\ell)}, \mathsf{ckR}^{(\ell)}), (\mathsf{mL}^{(\ell)}, \mathsf{mR}^{(\ell)}), 0, q_\ell(r_\ell), 1\right) = 1$.

   Above $\mathcal{C}$ is the challenge space, γ_R is the expansion factor of the norm over $\mathbb{M}$ when multiplying by scalars in R (see Definition 3.2) and $m(\mathcal{C})$ is the maximum norm of elements in $\mathcal{C}$ (see Definition 3.3).

Theorem 4.8. *The sumcheck argument* SCA *in Construction 4.7 satisfies the following properties:*
- *The prover performs the following operations: $O(n)$ scalar multiplications in $\mathbb{K}$; $O(n)$ scalar multiplications in $\mathbb{M}$; $O(n)$ commitments of length 1; and $O(n)$ additions in $\mathbb{C}$.*
- *The verifier performs the following operations: $O(n)$ additions and scalar multiplications in $\mathbb{K}$; 1 commitment of length 1; and $O(\log n)$ additions and scalar multiplications in $\mathbb{C}$.*
- *If* CM *is extendable (see Definition 4.11), then Construction 4.7 has perfect completeness.*
- *If* CM *is invertible (see Definition 4.14), then Construction 4.7 is a proof of knowledge.*

Proof. We prove the theorem via several lemmas. In Lemma 4.9 and Lemma 4.10 we discuss the arithmetic complexity of the prover and of the verifier. In Lemma 4.12 we prove perfect completeness. In Definition 4.14 we define the invertibility property, and in Lemma 4.16 we prove knowledge soundness. □

4.4 Efficiency

Lemma 4.9. *The prover in Construction 4.7 performs the following operations: $O(n)$ scalar multiplications in $\mathbb{K}$; $O(n)$ scalar multiplications in $\mathbb{M}$; $O(n)$ commitments of length 1; and $O(n)$ additions in $\mathbb{C}$.*

Proof. We describe how the honest prover, given vectors $\mathsf{ckL}, \mathsf{ckR} \in \mathbb{K}^n$, $\mathsf{mL}, \mathsf{mR} \in \mathbb{M}^n$, $\mathsf{ck}_0 \in \mathbb{K}_0$ and challenges $r_1, \ldots, r_\ell \in \mathcal{C}$, can compute the coefficients of the quadratic polynomials $q_1(X_1), \ldots, q_\ell(X_\ell)$ in $O(n)$ arithmetic and commitment operations over $\mathbb{K}$, $\mathbb{M}$ and $\mathbb{C}$. (The honest prover can then evaluate each of these polynomials at the locations $-1, 0, 1$, and send the evaluations.)

We proceed as follows. First, we recall the definitions of the polynomials $p(\vec{X})$ and $q_j(X_j)$ for each $j \in [\ell]$. Next, we express each $q_j(X_j)$ in terms of these coefficients. Finally, we give an algorithm that computes the coefficients of $q_1(X_1), \ldots, q_\ell(X_\ell)$ in $O(n)$ operations.

[9]Note that computing polynomials $q_i(X_i)$ does not require division by powers of 2 in the ring R. This is justified in Lemma 4.9 where an efficient prover algorithm is presented and analysed.

Recall that $p(\vec{X}) = \mathsf{CM.Commit}\left(\mathsf{ck}_0, p_{\mathsf{ckL}}(\vec{X}), p_{\mathsf{ckR}}(\vec{X}); p_{\mathsf{mL}}(\vec{X}), p_{\mathsf{mR}}(\vec{X}); 0\right)$ and

$$q_j(X_j) = 2^{-(\ell-j)} \cdot \sum_{\omega_{j+1},\ldots,\omega_\ell \in \{-1,1\}} p(r_1, \ldots, r_{j-1}, X_j, \omega_{j+1}, \ldots, \omega_\ell) \ .$$

We discuss partial evaluations of $p_{\mathsf{ckL}}(\vec{X})$, $p_{\mathsf{ckR}}(\vec{X})$, $p_{\mathsf{mL}}(\vec{X})$ and $p_{\mathsf{mR}}(\vec{X})$, and give explicit formulae for computing the polynomials $q_j(X_j)$ in terms of $\mathsf{ckL}^{(j)}$, $\mathsf{ckR}^{(j)}$, $\mathsf{mL}^{(j)}$ and $\mathsf{mR}^{(j)}$.

Expanding $p(r_1, \ldots, r_{\ell-1}, X_\ell)$ using the bilinear properties of $\mathsf{CM.Commit}$, we see that

$$\begin{aligned} p(r_1, \ldots, r_{\ell-1}, X_\ell) = q_\ell(X_\ell) = &\mathsf{CM.Commit}\left(\mathsf{ck}_0, \mathsf{ckL}_0^{(\ell-1)}, \mathsf{ckR}_0^{(\ell-1)}; \mathsf{mL}_0^{(\ell-1)}, \mathsf{mR}_0^{(\ell-1)}; 0\right) \\ &+ X_\ell \cdot \mathsf{CM.Commit}\left(\mathsf{ck}_0, \mathsf{ckL}_0^{(\ell-1)}, \mathsf{ckR}_1^{(\ell-1)}; \mathsf{mL}_0^{(\ell-1)}, \mathsf{mR}_1^{(\ell-1)}; 0\right) \\ &+ X_\ell \cdot \mathsf{CM.Commit}\left(\mathsf{ck}_0, \mathsf{ckL}_1^{(\ell-1)}, \mathsf{ckR}_0^{(\ell-1)}; \mathsf{mL}_1^{(\ell-1)}, \mathsf{mR}_0^{(\ell-1)}; 0\right) \\ &+ X_\ell^2 \cdot \mathsf{CM.Commit}\left(\mathsf{ck}_0, \mathsf{ckL}_1^{(\ell-1)}, \mathsf{ckR}_1^{(\ell-1)}; \mathsf{mL}_1^{(\ell-1)}, \mathsf{mR}_1^{(\ell-1)}; 0\right) \ . \end{aligned} \tag{6}$$

Since $q_{\ell-1}(r_{\ell-1}) = \frac{1}{2}\left(q_\ell(1) + q_\ell(-1)\right)$, which is the sum of the constant and X_ℓ^2 coefficients in Equation (6), we have

$$\begin{aligned} q_{\ell-1}(r_{\ell-1}) = &\mathsf{CM.Commit}\left(\mathsf{ck}_0, \mathsf{ckL}_0^{(\ell-1)}, \mathsf{ckR}_0^{(\ell-1)}; \mathsf{mL}_0^{(\ell-1)}, \mathsf{mR}_0^{(\ell-1)}; 0\right) \\ &+ \mathsf{CM.Commit}\left(\mathsf{ck}_0, \mathsf{ckL}_1^{(\ell-1)}, \mathsf{ckR}_1^{(\ell-1)}; \mathsf{mL}_1^{(\ell-1)}, \mathsf{mR}_1^{(\ell-1)}; 0\right) \ . \end{aligned}$$

Expanding using the recurrence relations, one finds that

$$\begin{aligned} q_{\ell-1}(r_{\ell-1}) &= \sum_{i_\ell \in \{0,1\}} \mathsf{CM.Commit}\left(\mathsf{ck}_0, \mathsf{ckL}_{i_\ell}^{(\ell-1)}, \mathsf{ckR}_{i_\ell}^{(\ell-1)}; \mathsf{mL}_{i_\ell}^{(\ell-1)}, \mathsf{mR}_{i_\ell}^{(\ell-1)}; 0\right) \\ &= \sum_{i_\ell \in \{0,1\}} \mathsf{CM.Commit}\left(\mathsf{ck}_0, \mathsf{ckL}_{0,i_\ell}^{(\ell-2)}, \mathsf{ckR}_{0,i_\ell}^{(\ell-2)}; \mathsf{mL}_{0,i_\ell}^{(\ell-2)}, \mathsf{mR}_{0,i_\ell}^{(\ell-2)}; 0\right) \\ &\quad + r_{\ell-1} \cdot \sum_{i_\ell \in \{0,1\}} \mathsf{CM.Commit}\left(\mathsf{ck}_0, \mathsf{ckL}_{0,i_\ell}^{(\ell-2)}, \mathsf{ckR}_{1,i_\ell}^{(\ell-2)}; \mathsf{mL}_{0,i_\ell}^{(\ell-2)}, \mathsf{mR}_{1,i_\ell}^{(\ell-2)}; 0\right) \\ &\quad + r_{\ell-1} \cdot \sum_{i_\ell \in \{0,1\}} \mathsf{CM.Commit}\left(\mathsf{ck}_0, \mathsf{ckL}_{1,i_\ell}^{(\ell-2)}, \mathsf{ckR}_{0,i_\ell}^{(\ell-2)}; \mathsf{mL}_{1,i_\ell}^{(\ell-2)}, \mathsf{mR}_{0,i_\ell}^{(\ell-2)}; 0\right) \\ &\quad + r_{\ell-1}^2 \cdot \sum_{i_\ell \in \{0,1\}} \mathsf{CM.Commit}\left(\mathsf{ck}_0, \mathsf{ckL}_{1,i_\ell}^{(\ell-2)}, \mathsf{ckR}_{1,i_\ell}^{(\ell-2)}; \mathsf{mL}_{1,i_\ell}^{(\ell-2)}, \mathsf{mR}_{1,i_\ell}^{(\ell-2)}; 0\right) \ . \end{aligned}$$

Continuing, we see that for each $j \in [\ell]$, we can express $q_j(X_j)$ in terms of the coefficients $\mathsf{ckL}_{\vec{i}(>j)}^{(j-1)}$, $\mathsf{ckR}_{\vec{i}(>j)}^{(j-1)}$, $\mathsf{mL}_{\vec{i}(>j)}^{(j-1)}$ and $\mathsf{mR}_{\vec{i}(>j)}^{(j-1)}$:

$$\begin{aligned} q_j(r_j) = &\sum_{\vec{i}(>j)} \mathsf{CM.Commit}\left(\mathsf{ck}_0, \mathsf{ckL}_{0,\vec{i}(>j)}^{(j-1)}, \mathsf{ckR}_{0,\vec{i}(>j)}^{(j-1)}; \mathsf{mL}_{0,\vec{i}(>j)}^{(j-1)}, \mathsf{mR}_{0,\vec{i}(>j)}^{(j-1)}; 0\right) \\ &+ X_{j-1} \cdot \sum_{\vec{i}(>j)} \mathsf{CM.Commit}\left(\mathsf{ck}_0, \mathsf{ckL}_{0,\vec{i}(>j)}^{(j-1)}, \mathsf{ckR}_{1,\vec{i}(>j)}^{(j-1)}; \mathsf{mL}_{0,\vec{i}(>j)}^{(j-1)}, \mathsf{mR}_{1,\vec{i}(>j)}^{(j-1)}; 0\right) \end{aligned}$$

$$+ X_{j-1} \cdot \sum_{\vec{i}(>j)} \mathsf{CM.Commit}\left(\mathsf{ck}_0, \mathsf{ckL}^{(j-1)}_{1,\vec{i}(>j)}, \mathsf{ckR}^{(j-1)}_{0,\vec{i}(>j)}; \mathsf{mL}^{(j-1)}_{1,\vec{i}(>j)}, \mathsf{mR}^{(j-1)}_{0,\vec{i}(>j)}; 0\right)$$
$$+ X^2_{j-1} \cdot \sum_{\vec{i}(>j)} \mathsf{CM.Commit}\left(\mathsf{ck}_0, \mathsf{ckL}^{(j-1)}_{1,\vec{i}(>j)}, \mathsf{ckR}^{(j-1)}_{1,\vec{i}(>j)}; \mathsf{mL}^{(j-1)}_{1,\vec{i}(>j)}, \mathsf{mR}^{(j-1)}_{1,\vec{i}(>j)}; 0\right) \quad . \tag{7}$$

We now give an algorithm that computes the coefficients of $q_1(X_1), \dots, q_\ell(X_\ell)$ in $O(n)$ arithmetic operations. For $j = 0$, the prover already knows the coefficients $\mathsf{ckL}^{(0)}_{\vec{i}(>0)}$, $\mathsf{ckR}^{(0)}_{\vec{i}(>0)}$, $\mathsf{mL}^{(0)}_{\vec{i}(>0)}$ and $\mathsf{mR}^{(0)}_{\vec{i}(>0)}$. Then, for each $j \in [\ell]$:

- The prover has the coefficients $\mathsf{ckL}^{(j)}_{\vec{i}(>j)}$, $\mathsf{ckR}^{(j)}_{\vec{i}(>j)}$, $\mathsf{mL}^{(j)}_{\vec{i}(>j)}$ and $\mathsf{mR}^{(j)}_{\vec{i}(>j)}$ for every $\vec{i}(> j) \in \{0,1\}^{\ell-j}$.
- The sums in Equation (7) giving the coefficients of $q_j(X_j)$ contain $4 \cdot 2^{\ell-j}$ terms in total. Given the values of $\mathsf{ckL}^{(j)}_{\vec{i}(>j)}$, $\mathsf{ckR}^{(j)}_{\vec{i}(>j)}$, $\mathsf{mL}^{(j)}_{\vec{i}(>j)}$ and $\mathsf{mR}^{(j)}_{\vec{i}(>j)}$ for every $\vec{i}(> j) \in \{0,1\}^{\ell-j}$ we can compute all of the terms in $4 \cdot 2^{\ell-j}$ commit operations of length 1 and add them together in $4 \cdot 2^{\ell-j}$ additions over $\mathbb{C}$ to find the coefficients of $q_j(X_j)$.
- On receiving r_j from $\mathbf{V}$, the prover computes the coefficients $\mathsf{ckL}^{(j)}_{\vec{i}(>j)}$, $\mathsf{ckR}^{(j)}_{\vec{i}(>j)}$, $\mathsf{mL}^{(j)}_{\vec{i}(>j)}$ and $\mathsf{mR}^{(j)}_{\vec{i}(>j)}$ for every $\vec{i}(> j) \in \{0,1\}^{\ell-j}$ via the recurrence relations. This requires $2^{\ell-j}$ scalar multiplications and $2^{\ell-j}$ additions in $\mathbb{K}$ and $\mathbb{M}$. The prover need not compute $\mathsf{ckL}^{(\ell)}$ or $\mathsf{ckR}^{(\ell)}$.

This means that the total cost of computing the quadratic polynomials $q_1(X_1), \dots, q_\ell(X_\ell)$ is the sum of a geometric series and is $O(2^\ell) = O(n)$ commit operations of length 1 and additions over $\mathbb{K}$. Each of the polynomials $q_1(X_1), \dots, q_\ell(X_\ell)$ can be evaluated to find the necessary evaluation points to send to the verifier when required. □

Lemma 4.10. *The verifier in Construction 4.7 performs the following operations: $O(n)$ additions and scalar multiplications in $\mathbb{K}$; 1 commitment of length 1; and $O(\log n)$ additions and scalar multiplications in $\mathbb{C}$.*

Proof. In the verification phase (Item 3), the verifier: (a) computes the commitment keys $\mathsf{ckL}^{(\ell)} = p_{\mathsf{ckL}}(\vec{r})$ and $\mathsf{ckR}^{(\ell)} := p_{\mathsf{ckR}}(\vec{r})$ using $O(n)$ additions and scalar multiplications in $\mathbb{K}$; (b) uses $O(\ell) = O(\log n)$ arithmetic operations over $\mathbb{C}$ to check that $q_1(1) + q_1(-1) = 2 \cdot \mathsf{cm}$ and, for each $i \in \{2, \dots, \ell\}$, that $q_i(1) + q_i(-1) = 2 \cdot q_{i-1}(r_{i-1})$; (c) uses 1 commit operation of length 1 to check that $q_\ell(r_\ell) = \mathsf{CM.Commit}\left(\mathsf{ck}_0, \mathsf{ckL}^{(\ell)}, \mathsf{ckR}^{(\ell)}; \mathsf{mL}^{(\ell)}; \mathsf{mR}^{(\ell)}; 0\right)$. Note that computing the commitment keys is the computation that dominates the verifier's running time. □

4.5 Completeness

Definition 4.11. CM *is* **extendable** *if for all $\lambda, n, n' \in \mathbb{N}$, $\mathsf{pp} \leftarrow \mathsf{CM.Setup}(1^\lambda, n + n')$, for all $(\mathsf{ck}_0, \mathsf{ck}) \in \mathbb{K}_0 \times \mathbb{K}^{n+n'}$, such that $(\mathsf{ck}_0, \mathsf{ck}) = \mathsf{CM.ExtendCK}(\mathrm{ck})$ for $\mathrm{ck} \leftarrow \mathsf{CM.KeyGen}(\mathsf{pp})$, $\mathsf{m} \in \mathbb{M}^{n+n'}$ and all $\rho \in \mathbb{R}$, $\rho' \in \mathbb{R}'$, we have*

$$\begin{aligned}\mathsf{CM.BilinearCommit}(\mathsf{ck}_0, \mathsf{ck}; \mathsf{m}; \rho + \rho') =& \mathsf{CM.BilinearCommit}(\mathsf{ck}_0, \mathsf{ck}[: n]; \mathsf{m}[: n]; \rho) \\ &+ \mathsf{CM.BilinearCommit}(\mathsf{ck}_0, \mathsf{ck}[n :]; \mathsf{m}[n :]; \rho')\end{aligned}$$

Lemma 4.12. *If CM is extendable, then Construction 4.7 has perfect completeness.*

Proof. Fix any challenges $r_1, \dots, r_\ell \in \mathcal{C}$ from the verifier. We need to show that the (honest) prover makes the verifier accept (all the conditions in Item 3 of the protocol hold).

First, the definitions of $p(\vec{X})$ and $\{q_i(X_i)\}_{i=1,\dots,\ell}$, along with polynomial arithmetic, directly imply that $q_\ell(r_\ell) = \mathsf{CM.Commit}\left(p_{\mathsf{ckL},\vec{r}}, p_{\mathsf{mL},\vec{r}}; p_{\mathsf{ckR},\vec{r}}, p_{\mathsf{mR},\vec{r}}; \mathsf{ck}_0, 0\right)$, which by the completeness property of the commitment scheme implies that $\mathsf{CM.Open}\left((\mathsf{ck}_0, \mathsf{ckL}^{(\ell)}, \mathsf{ckR}^{(\ell)}), (\mathsf{mL}^{(\ell)}, \mathsf{mR}^{(\ell)}), 0, q_\ell(r_\ell), 1\right) = 1$. The fact that $q_i(1) + q_i(-1) = 2 \cdot q_{i-1}(r_{i-1})$ holds for each $i \in \{2, \dots, \ell\}$ also follows from the definitions of $p(\vec{X})$ and $\{q_i(X_i)\}_{i=1,\dots,\ell}$, along with polynomial arithmetic.

Next, the norm bounds on $p_{\mathsf{ckL},\vec{r}} = p_{\mathsf{ckL}}(\vec{r})$ and $p_{\mathsf{ckR},\vec{r}} = p_{\mathsf{ckR}}(\vec{r})$ follow from applications of the triangle inequality and the multiplicative property of the norm.

Finally, we are left to show that $q_1(1)+q_1(-1) = 2\cdot\mathsf{cm}$, for which it suffices to show that $\sum_{\vec{\omega}\in\{-1,1\}^\ell} p(\vec{\omega}) = 2^\ell \cdot \mathsf{cm}$ because $q_1(1) + q_1(-1) = 2^{-(\ell-1)} \cdot \sum_{\vec{\omega}\in\{-1,1\}^\ell} p(\vec{\omega})$. We have

$$\begin{aligned} p(\vec{X}) &= \mathsf{CM.Commit}\left(\mathsf{ck}_0, p_{\mathsf{ckL}}(\vec{X}), p_{\mathsf{ckR}}(\vec{X}); p_{\mathsf{mL}}(\vec{X}), p_{\mathsf{mR}}(\vec{X}); 0\right) \\ &= \sum_{\vec{i}\in\{0,1\}^\ell} \mathsf{CM.Commit}\left(\mathsf{ck}_0, \mathsf{ckL}_{\vec{i}}, p_{\mathsf{ckR}}(\vec{X}); \mathsf{mL}_{\vec{i}}, p_{\mathsf{mR}}(\vec{X}); 0\right) \cdot \vec{X}^{\vec{i}} \\ &= \sum_{\vec{i},\vec{j}\in\{0,1\}^\ell} \mathsf{CM.Commit}\left(\mathsf{ck}_0, \mathsf{ckL}_{\vec{i}}, \mathsf{ckR}_{\vec{j}}; \mathsf{mL}_{\vec{i}}, \mathsf{mR}_{\vec{j}}; 0\right) \cdot \vec{X}^{\vec{i}+\vec{j}} \quad . \end{aligned}$$

By Lemma 4.2 and the extendability property of CM, we deduce that

$$\begin{aligned} \sum_{\vec{\omega}\in\{-1,1\}^\ell} p(\vec{\omega}) &= 2^\ell \cdot \sum_{\vec{i}\equiv\vec{0} \bmod 2} p_{\vec{i}} \\ &= 2^\ell \cdot \sum_{\vec{i}\in\{0,1\}^\ell} \mathsf{CM.Commit}\left(\mathsf{ck}_0, \mathsf{ckL}_{\vec{i}}, \mathsf{ckR}_{\vec{i}}; \mathsf{mL}_{\vec{i}}, \mathsf{ckR}_{\vec{i}}; 0\right) \\ &= 2^\ell \cdot \mathsf{CM.Commit}\left(\mathsf{ck}_0, \mathsf{ckL}, \mathsf{ckR}; \mathsf{mL}, \mathsf{mR}; 0\right) = 2^\ell \cdot \mathsf{cm} \quad . \end{aligned}$$

□

4.6 Knowledge soundness

Before defining invertibility and proving the knowledge soundness of the protocol, we define commitment keys with respect to a part of an extraction tree. This definition is similar to Definition 4.6 with the exception that it considers multiple different random challenges for the same round.

Definition 4.13. *Let $\vec{r}$ be randomness labels in a path of length $j-1$ of the tree T and $\{r_j^{(i)}\}_{i\in[K]}$ be the randomness labels of its K outgoing edges. Let $\mathsf{ckL}^{(j,i)}$, $\mathsf{ckR}^{(j,i)} \in \mathbb{K}^{2^{\ell-j}}$ be the vectors of coefficients of the polynomials $p_{\mathsf{ckL}}(\vec{r}, r_j^{(i)}, X_{j+1}, \dots, X_\ell)$, and $p_{\mathsf{ckR}}(\vec{r}, r_j^{(i)}, X_{j+1}, \dots, X_\ell)$, where $\ell = \log n$. As in Lemma 4.9, the following recurrence relations hold between coefficients:*

$$\mathsf{ckL}^{(j,i)}_{\vec{i}(>j)} = \mathsf{ckL}^{(j-1)}_{0,\vec{i}(>j)} + r_j^{(i)} \cdot \mathsf{ckL}^{(j-1)}_{1,\vec{i}(>j)} \quad .$$

Similarly for ckR.

Now, we are ready to define invertibility, which is our main tool for proving knowledge soundness.

Definition 4.14. *Let* $\mathsf{CM} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Commit}, \mathsf{Open})$ *be a bilinear commitment scheme. For* $K, B_{\mathsf{INV}}, C, D \colon \mathbb{N} \to \mathbb{R}$ *(possibly functions of the security parameter* λ*), we say that* CM *is* $(K, B_{\mathsf{INV}}, C, D)$**-invertible** *for the challenge space* $\mathcal{C}$ *if there is a polynomial-time inverter algorithm* $\mathcal{D}$ *such that for every* $n \in \mathbb{N}$*,* $L = 2^I \leq n/2$*, vector* $\vec{r} \in \mathcal{C}^I$*, distinct challenges* $r_j^{(i)} \in \mathcal{C}$ *for* $i \in [K]$*, and any polynomial-time algorithm* $\mathcal{A}$*, the following experiment outputs* 1 *with all but negligible probability.*

1. $\mathsf{pp} \leftarrow \mathsf{CM.Setup}(1^\lambda, n)$.
2. $(\mathsf{ck}_0, \mathsf{ckL}^*, \mathsf{ckR}^*) = \mathsf{CM.ExtendCK}(\mathrm{ck})$*, where* $\mathrm{ck} \leftarrow \mathsf{CM.KeyGen}(\mathsf{pp})$.
3. $\mathcal{A}(\mathsf{pp}, \mathrm{ck})$ *outputs messages* $(\mathsf{mL}^{(1)}, \mathsf{mR}^{(1)}), \ldots, (\mathsf{mL}^{(K)}, \mathsf{mR}^{(K)}) \in \mathbb{M}(B)^L \times \mathbb{M}(B)^L$*, where* $D \cdot B \leq B_{\mathsf{INV}}$*, a polynomial* $q(X) = q_0 + q_1 X + q_2 X^2$ *in* $\mathbb{C}[X]$*, and a slackness* $c \in \mathbb{Z}$.
4. *Let* $\mathsf{ckL}^{(\ell-I+1,i)}$*,* $\mathsf{ckR}^{(\ell-I+1,i)}$*,* $\mathsf{ckL} = \mathsf{ckL}^{(\ell-I)}_{0,\vec{i}(>j)}$*,* $\mathsf{ckL}' = \mathsf{ckL}^{(\ell-I)}_{1,\vec{i}(>j)}$*,* $\mathsf{ckR} = \mathsf{ckR}^{(\ell-I)}_{0,\vec{i}(>j)}$*, and* $\mathsf{ckR}' = \mathsf{ckR}^{(\ell-I)}_{1,\vec{i}(>j)}$ *be the keys corresponding to* $\vec{r}$ *and* $r_j^{(i)}$ *as in Definition 4.13. The experiment outputs* 1 *if either*

$$\mathsf{CM.Open}\left((\mathsf{ck}_0, \mathsf{ckL}^{(\ell-I+1,i)}, \mathsf{ckR}^{(\ell-I+1,i)}, (\mathsf{mL}^{(i)}, \mathsf{mR}^{(i)}), 0, q(r_j^{(i)}), c\right) \neq 1 \ ,$$

for some $i \in [K]$*, or the inverter* $\mathcal{D}$*, that takes the input* pp, ck *and the output of* $\mathcal{A}$*, outputs* $M_L, M_R \in \mathbb{M}(D \cdot B)^{2L}$*, such that* M_L, M_R *are distinct and satisfy*

$$\mathsf{CM.Open}\left((\mathsf{ck}_0, \mathsf{ckL}||\mathsf{ckL}', \mathsf{ckR}||\mathsf{ckR}'), (M_L, M_R), 0, 1/2 \cdot (q(1) + q(-1)), C\, c\right) = 1 \ ,$$

Otherwise, the output is 0.

Remark 4.15. For many of our instantiations, $\mathsf{ck}_0, \mathsf{ckL}^*, \mathsf{ckR}^*$ are all uniformly random vectors over a module. This implies that the derived commitment key $(\mathsf{ck}_0, \mathsf{ckL}, \mathsf{ckL}', \mathsf{ckR}, \mathsf{ckR}') \in \mathbb{K}_0 \times \mathbb{K}^L \times \mathbb{K}^L \times \mathbb{K}^L \times \mathbb{K}^L$ in Definition 4.14 is simply a uniformly random vector of a different length over the same module. In these cases, Definition 4.14 can be simplified accordingly.

Lemma 4.16. *Suppose that* CM *is* $(K, B_{\mathsf{INV}}, C, D)$*-invertible for challenge space* $\mathcal{C}$*, and* $B' := n \cdot (\gamma_R m(\mathcal{C}) D)^\ell \cdot B_{\mathsf{C}}$ *satisfies* $B' \leq B_{\mathsf{INV}}$ *. Then there exists an efficient algorithm* χ *such that, given a commitment key* $(\mathsf{ck}_0, \mathsf{ckL}, \mathsf{ckR}) \in \mathbb{K}_0 \times \mathbb{K}^{n/2} \times \mathbb{K}^{n/2}$*, commitment* $\mathsf{cm} \in \mathbb{C}$*, and a* K^ℓ*-tree of accepting transcripts for Construction 4.7 applied to* $\mathcal{R}_{\mathrm{SC}}(1, B_{\mathsf{C}})$*, extracts a witness for* $\mathcal{R}_{\mathrm{SC}}(c, B')$ *with* $c := C^\ell$*.*

Proof. First we describe the extractor algorithm, then show that it runs in polynomial time, and finally prove that it produces the required output.

The extractor. We describe the extractor χ for Construction 4.7. The inputs and output are as follows:

- *Input:* The instance $(\mathsf{CM}, \mathsf{pp}, \mathsf{ck}_0, \mathsf{ckL}, \mathsf{ckR}, \mathsf{cm})$ and a K^ℓ-tree of accepting transcripts for Construction 4.7, for which the opening messages satisfy $\|\mathsf{mL}^{(\ell)}\|_{\mathbb{M}}, \|\mathsf{mR}^{(\ell)}\|_{\mathbb{M}} \leq n \cdot (\gamma_R m(\mathcal{C}))^\ell \cdot B_{\mathsf{C}}$.
- *Output:* Messages $(\mathsf{mL}, \mathsf{mR}) \in \mathbb{M}(B')^{n/2} \times \mathbb{M}(B')^{n/2}$, satisfying $\mathsf{CM.Open}((\mathsf{ck}_0, \mathsf{ckL}, \mathsf{ckR}), (\mathsf{mL}, \mathsf{mR}), 0, \mathsf{cm}, c) = 1$.

Let $\vec{r}_j^{(i)} = (r_1, \ldots, r_j^{(i)})$ for $i \in [K]$ be the prefix of $K^{\ell-j}$ paths of the extraction tree and let $q_j[\vec{r}_{j-1}](X_j)$ be the polynomial of the j-th round in the transcript corresponding to $\vec{r}_{j-1} = (r_1, \ldots, r_{j-1})$. Define $\mathsf{ckL}^{(j,i)}, \mathsf{ckR}^{(j,i)} \in \mathbb{K}^{2^{\ell-j}}$ to be the vector of coefficients of $p_{\mathsf{ckL}}(\vec{r}_j^{(i)}, X_{j+1}, \ldots, X_\ell)$, and $p_{\mathsf{ckR}}(\vec{r}_j^{(i)}, X_{j+1}, \ldots, X_\ell)$ as in Definition 4.13. The algorithm works as follows.

For $j = \ell, \ldots, 1$:

- Let $L = 2^{\ell-j}$ and $p_{\mathsf{mL},\vec{r}_j^{(i)}}, p_{\mathsf{mR},\vec{r}_j^{(i)}} \in \mathbb{M}(n\ D^{\ell-j}\ \gamma_R^\ell\ m(\mathcal{C})^\ell \cdot B_\mathsf{C})^L$ be the opening values associated with challenge string $\vec{r}_j^{(i)} = (r_1, \ldots, r_j^{(i)})$.
- For each subtree $\{\vec{r}_j^{(i)}\}_{i\in[K]} = \{(r_1, \ldots, r_{j-1}, r_j^{(i)})\}_{i\in[K]}$ in the extraction tree:
  - For each $i \in [K]$, we have that

$$\mathsf{CM.Open}\left((\mathsf{ck}_0, \mathsf{ckL}^{(j,i)}, \mathsf{ckR}^{(j,i)}), (p_{\mathsf{mL},\vec{r}_j^{(i)}}, p_{\mathsf{ckR},\vec{r}_j^{(i)}}), 0, q_j[\vec{r}_{j-1}](r_j^{(i)}), C^{\ell-j}\right) = 1$$

  - Run the inverter $\mathcal{D}$ on input pp, the keys $(\mathsf{ck}_0, \mathsf{ckL}, \mathsf{ckR})$, the messages $(p_{\mathsf{mL},\vec{r}_j^{(i)}}, p_{\mathsf{ckR},\vec{r}_j^{(i)}})$, the polynomial $q_j[\vec{r}_{j-1}]$ and the slackness $C^{\ell-j}$ to produce new opening values $p_{\mathsf{mL},\vec{r}_{j-1}}, p_{\mathsf{mR},\vec{r}_{j-1}} \in \mathbb{M}(n\ D^{\ell-j}\ \gamma_R^\ell\ m(\mathcal{C})^\ell \cdot B_\mathsf{C})^{2L}$ such that

$$\mathsf{CM.Open}\left((\mathsf{ck}_0, \mathsf{ckL}^{(j-1)}_{0,\vec{i}(>j)} \| \mathsf{ckL}^{(j-1)}_{1,\vec{i}(>j)}, \mathsf{ckR}^{(j-1)}_{0,\vec{i}(>j)} \| \mathsf{ckR}^{(j-1)}_{1,\vec{i}(>j)}), (p_{\mathsf{mL},\vec{r}_{j-1}^{(i)}}, p_{\mathsf{ckR},\vec{r}_{j-1}^{(i)}}), 0, \mathsf{cm}_{j-1}, C^{\ell-j+1}\right) = 1$$

    for $\mathsf{cm}_{j-1} = 1/2 \cdot (q_j[\vec{r}_{j-1}](1) + q_j[\vec{r}_{j-1}](-1)) = q_{j-1}[\vec{r}_{j-2}](r_{j-1})$. If the inverter fails to produce a valid output, then output $\perp$.

The values of $c = C^\ell$ and $B' := n \cdot (\gamma_R m(\mathcal{C}) D)^\ell \cdot B_\mathsf{C}$ follow in a straightforward manner by induction.

Running time. Let $T(L)$ denote the running time of the inverter $\mathcal{D}$ on inputs with parameter L. At step j, the extractor runs the inverter $\mathcal{D}$ a total of K^{j-1} times with parameter $L = 2^{\ell-j}$. Therefore, the running time of the extractor is $\sum_{j=1}^{\ell} K^{j-1} T(2^{\ell-j})$.

Success probability. Let $\epsilon(L)$ denote the failure probability of the inverter $\mathcal{D}$ on properly-distributed inputs with length parameter L. If any execution of the inverter for any j fails then the extractor will terminate in failure rather than producing an opening of cm. Therefore, by a union bound, the failure probability of the extractor is at most $\sum_{j=1}^{\ell} K^{j-1} \epsilon(2^{\ell-j})$. □

5 Instantiations of bilinear commitments

We describe several instantiations of bilinear commitments that are suitable for a sumcheck argument (Section 4); in particular, we will require the commitment to be extendable and invertible.

We proceed as follows. In Section 5.1 we describe the *bilinear relation assumption*, which is an abstract cryptographic assumption that captures several well-known assumptions, such as the discrete logarithm assumption, assumptions based on bilinear pairings, the SIS assumption, and assumptions related to groups of unknown order; to support all these settings, we use the notion of an *argument-friendly bilinear module*.

Next, we provide examples of bilinear commitments. In Section 5.2 we describe a generalization of the Pedersen commitment. Then, in Section 5.3 we define the scalar-product commitment, a commitment scheme that depends on the scalar-product of parts of the message. Finally, in Section 5.4 we observe that in certain cases we can also define a compressed version of the scalar-product commitment.

5.1 Bilinear modules

We define bilinear modules and then describe a general cryptographic assumption, the *bilinear relation assumption*, that will imply the binding property of bilinear commitments that we consider.

Definition 5.1. *A* **bilinear module** *is a tuple $\mathcal{M} = (R, M_L, M_R, M_T, e)$ where M_L, M_R, M_T are R-modules with a non-degenerate bilinear map $e\colon M_L \times M_R \to M_T$. An* **argument-friendly** *bilinear module is a tuple $\mathcal{M} = (R, M_L, M_R, M_T, e, \mathcal{C}, B_{\mathsf{C}}, B_{\max})$ where (R, M_L, M_R, M_T, e) is a bilinear module, $\mathcal{C} \subseteq R$ is a sampling set, $B_{\mathsf{C}}, B_{\max} \in \mathbb{Z}$, and R and M_L are equipped with norms $\|\cdot\|_R$ and $\|\cdot\|_{M_L}$ respectively.*

To simplify notation in later analysis, although multiplication of elements of M_L and R may cause norm expansion by different factors γ_R and γ_{M_L}, we will only use the notation γ_R, which will represent the maximum of these quantities.

We use arithmetic notation as a shorthand for the application of e. For example, given $a \in M_L$ and $\mathsf{G} \in M_R$, we use "$a \cdot \mathsf{G}$" to denote $e(a, \mathsf{G}) \in M_T$. Similarly, given $\vec{a} \in M_L^n$ and $\vec{\mathsf{G}} \in M_R^n$, we use "$\langle \vec{a}, \vec{\mathsf{G}} \rangle$" to denote $\sum_{i\in[n]} e(a_i, \mathsf{G}_i) \in M_T$.

Definition 5.2. *Let $\mathcal{G}$ be an algorithm that on input 1^λ, $n \in \mathbb{N}$, and a relation $\mathcal{S}$ outputs an argument-friendly bilinear module $\mathcal{M}$, let $\mathcal{K}$ be an algorithm that outputs a vector in M_R^n. We say that $(\mathcal{G}, \mathcal{K}, \mathcal{S})$ satisfies the* **bilinear relation assumption** *if for every $n \in \mathbb{N}$ and polynomial-size adversary $\mathcal{A}$*

$$\Pr\left[\begin{array}{c|r} \mathcal{S}(\mathcal{M}) = 1 & \\ \vec{\mathsf{G}} \in M_R^n & \mathcal{M} = (R, M_L, M_R, M_T, e, \mathcal{C}, B_{\mathsf{C}}, B_{\max}) \leftarrow \mathcal{G}(1^\lambda, n) \\ \vec{a} \in M_L(B_{\max})^n & \vec{\mathsf{G}} \leftarrow \mathcal{K}(\mathcal{M}) \\ \vec{a} \neq 0^n & \vec{a} \leftarrow \mathcal{A}(\mathcal{M}, \vec{\mathsf{G}}) \\ \langle \vec{a}, \vec{\mathsf{G}} \rangle = 0 & \end{array}\right] = \mathrm{negl}(\lambda) \ .$$

The relation $\mathcal{S}$ will be used to control the "gap" between the parameter B_{C}, which represents the norm of honestly committed messages in some protocol we wish to execute, and $B_{\max}$, which represents the maximum norm of messages for which we wish the bilinear relation assumption to be hard, in order to prove that the sumcheck argument (Construction 4.7) is secure. In particular, Lemma 4.16 shows that when Construction 4.7 is executed on input messages with norm at most B_{C}, then one can extract openings of norm at most $B' := n \cdot (\gamma_R m(\mathcal{C}) D)^\ell \cdot B_{\mathsf{C}}$. We choose $\mathcal{S}$ to guarantee that $B_{\mathsf{INV}} \leq B_{\max}$, where B_{INV} depends on B' and the commitment scheme. We discuss the choice of relations $\mathcal{S}$ used in the bilinear relation assumption, and explain what suffices when applying the sumcheck argument (Construction 4.7) to each of the commitment schemes in the corresponding sections.

Algebraic instantiations. We describe instantiations of the Bilinear Relation Assumption in several cryptographic settings. In each setting, we specify the output of the algorithms $\mathcal{G}(1^\lambda, n)$ and $\mathcal{K}(\mathcal{M})$. The bilinear relation assumption in each of these settings gives us a corresponding well-known cryptographic assumption (such as the discrete logarithm or SIS assumptions).

- *Discrete logarithms.* The algorithm $\mathcal{G}_{\mathrm{DL}}(1^\lambda, n)$ samples a discrete logarithm group $\mathbb{G}$ of prime order $q = \exp(\lambda)$ with generator G (ignoring n). Here $R := \mathbb{F}_q$, $M_L := \mathbb{F}_q$ equipped with the trivial norm (equals 1 for any non-zero element of R and equals 0 otherwise), $M_R := \mathbb{G}$, $M_T := \mathbb{G}$, e is group exponentiation, $\mathcal{C} := \mathbb{F}_q$, $B_{\mathsf{C}} := 1$ and $B_{\max} := 1$. The bilinear relation assumption becomes the discrete logarithm assumption, linked to the original Pedersen commitment scheme.

- *Pairings.* The algorithm $\mathcal{G}_{\mathrm{BP}}(1^\lambda, n)$ samples groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ of prime order $q = \exp(\lambda)$, G_1 that generates $\mathbb{G}_1$, G_2 that generates $\mathbb{G}_2$, G_T that generates $\mathbb{G}_T$ and a map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Here $R := \mathbb{F}_q$, $M_L := \mathbb{G}_1$ equipped with the trivial norm, $M_R := \mathbb{G}_2$, $M_T := \mathbb{G}_T$, e is the map as above, $\mathcal{C} := \mathbb{F}_q$, $B_{\mathsf{C}} := 1$ and $B_{\max} := 1$.

  The bilinear relation assumption becomes the double-pairing assumption, which is implied by the decisional Diffie-Hellman in $\mathbb{G}_2$, and connected to the commitment scheme of [AFGHO16]. One could also construct a commitment scheme with $M_L := \mathbb{G}_2$ and $M_R := \mathbb{G}_1$.

- *Lattices.* The algorithm $\mathcal{G}_{\mathsf{SIS}}(1^\lambda, n)$ outputs a ring $\mathbb{Z}[X]/\langle X^d + 1\rangle$, where d is a power of 2, a prime number q, and numbers $B_{\mathsf{SIS}}, B \in \mathbb{Z}$. Here $R := \mathbb{Z}[X]/\langle X^d + 1\rangle$, $M_L := \mathbb{Z}[X]/\langle X^d + 1\rangle$ equipped with the ℓ_∞-norm, $M_R := \left(\mathbb{Z}_q[X]/\langle X^d + 1\rangle\right)^r$, $M_T = \left(\mathbb{Z}_q[X]/\langle X^d + 1\rangle\right)^r$, e is polynomial multiplication modulo q and $X^d + 1$, $\mathcal{C} := \{X^i \in \left(\mathbb{Z}[X]/\langle X^d + 1\rangle\right) : 0 \le i \le 2d - 1\}$, $B_{\mathsf{C}} := B$, and $B_{\max} := B_{\mathsf{SIS}}$. The algorithm $\mathcal{K}_{\mathsf{SIS}}(\mathcal{M})$ outputs a uniformly random vector in $M_R^n = (\mathbb{Z}_q[X]/\langle X^d + 1\rangle)^{r \times n}$.

  The parameter B_{SIS} should be less than $\min\{q, 2^{2\sqrt{r \log q \log \delta}}\}$[10] so that it is difficult to find solutions to the SIS problem of norm less than B_{SIS} [GN08], which implies the bilinear relation assumption used in our commitment schemes, in this case connected to Ajtai's one way function.

- *Groups of unknown order.* There are two instantiations of groups of unknown order (GUO), the RSA groups and the class groups of an imaginary quadratic order. Both relate to the commitment schemes of [BFS20].

  The algorithm $\mathcal{G}_{\mathsf{RSA}}(1^\lambda, n)$ outputs an RSA group $\mathbb{G}$ and primes q and p (which are unrelated to the primes that determine the order of $\mathbb{G}$). Here $R := \mathbb{Z}$, $M_L := \mathbb{Z}$ equipped with the ℓ_∞-norm, $M_R = \mathbb{G}$, $M_T = \mathbb{G}$, e is group exponentiation, $\mathcal{C} = \mathbb{Z}(\frac{p-1}{2})$, $B_{\mathsf{C}} := \frac{p-1}{2}$ and $B_{\max} = \frac{q-1}{2}$. The algorithm $\mathcal{K}_{\mathsf{RSA}}(\mathcal{M})$ samples a uniformly random element $\mathsf{G} \in M_R = \mathbb{G}$ and outputs the vector $(\mathsf{G}, \mathsf{G}^q, \ldots, \mathsf{G}^{q^{n-1}}) \in M_R^n$.

  The algorithm $\mathcal{G}_{\mathsf{CL}}(1^\lambda, n)$ outputs a class group $\mathbb{G}$ and primes q and p (which are unrelated to the primes that determine the order of $\mathbb{G}$). Here $R := \mathbb{Z}$, $M_L := \mathbb{Z}$ equipped with the ℓ_∞-norm, $M_R = \mathbb{G}$, $M_T = \mathbb{G}$, e is group exponentiation, $\mathcal{C} = \mathbb{Z}(\frac{p-1}{2})$, $B_{\mathsf{C}} := \frac{p-1}{2}$ and $B_{\max} = \lfloor \frac{q-1}{4} \rfloor$[11]. The algorithm $\mathcal{K}_{\mathsf{CL}}(\mathcal{M})$ samples a uniformly random element $\mathsf{G} \in M_R = \mathbb{G}$ and outputs the vector $(\mathsf{G}, \mathsf{G}^q, \ldots, \mathsf{G}^{q^{n-1}}) \in M_R^n$.

[10] The constant δ is related to the optimal block-size in the BKZ algorithm applied to the SIS problem [GN08] and is typically set to $\delta \approx 1.005$.

[11] The difference in $B_{\max}$ between RSA and class groups is related to the fact that computing square roots is easy in class groups.

On challenge spaces and sampling sets. We discuss the sampling properties of the challenge space for each instantiation.

- *Discrete logarithms.* In this setting, $\mathcal{C} = \mathbb{F}_q$, and any non-zero challenge difference is invertible in $\mathbb{F}_q$. Hence, $\mathcal{C}$ is a strong sampling set for $R = M_L = \mathbb{F}_q$ and also a strong sampling set for $M_R = M_T = \mathbb{G}$.
- *Pairings.* As in the discrete logarithm setting, $\mathcal{C} = \mathbb{F}_q$, and any non-zero challenge difference is invertible. Hence, $\mathcal{C}$ is a strong sampling set for $R = \mathbb{F}_q$, $M_L = \mathbb{G}_1$, $M_R = \mathbb{G}_2$ and $M_T = \mathbb{G}_T$.
- *Lattices.* In our security proofs in the lattice setting, we rely on the lemma below, which states that the difference of any two elements in the challenge space has a short "pseudoinverse".

  Lemma 5.3 ([BCKLN14, Lemma 3.1]). *Let d be a power of 2 and let $0 \leq i < j \leq 2d - 1$. Over the ring $\mathbb{Z}[X]/(X^d + 1)$, the element $2(X^i - X^j)^{-1}$ has coefficients in $\{-1, 0, 1\}$.*

  This implies that $\mathcal{C}$ is a strong sampling set for $M_R = M_T = \big(\mathbb{Z}_q[X]/\langle X^d + 1\rangle\big)^r$.
- *GUO.* In this setting, the challenge space $\mathcal{C} = \mathbb{Z}(\frac{p-1}{2})$ is a sampling set for $M_T = \mathbb{G}$ *computationally*, according to the following definition.

  Definition 5.4. *Let $\mathcal{G}$ be an algorithm that on input 1^λ and $n \in \mathbb{N}$ outputs an argument-friendly bilinear module $\mathcal{M}$. We say that $\mathcal{G}$ provides* **computational sampling sets** *for M_T if for every polynomial-size adversary $\mathcal{A}$*

$$\Pr\left[\begin{array}{c} c_1, c_2 \in \mathcal{C} \\ m \in M_T \\ c_1 \neq c_2 \\ m \neq 0 \\ (c_1 - c_2)m = 0 \end{array} \middle| \begin{array}{r} \mathcal{M} = (R, M_L, M_R, M_T, e, \mathcal{C}, B_{\mathsf{C}}, B_{\max}) \leftarrow \mathcal{G}(1^\lambda, n) \\ (c_1, c_2, m) \leftarrow \mathcal{A}(\mathcal{M}, \vec{\mathsf{G}}) \end{array}\right] = \mathrm{negl}(\lambda) \ .$$

  In other words, multiplication of elements in $\mathbb{G}$ by $(c_1 - c_2)$ may fail to be injective, but it is computationally difficult to find witnesses to this failure. Definition 5.4 is satisfied for groups of unknown order because if $(c_1 - c_2)m = 0$, then m is an element of known order, and we can break the adaptive root assumption for the group $\mathbb{G}$ (see [BFS20, Appendix A]). This computational property will suffice wherever the sampling set property is used in later security arguments.

Inversion constant. We define the K-th inversion constant D_K, which is related to the norm of the elements of the adjugate of a Vandermode matrix constructed from elements from a sampling set.

Definition 5.5. *Let R be a ring with norm $\|\cdot\|_R$ and $\mathcal{C} \subseteq R$ be a sampling set. Let $V_{c_1,\dots,c_K}$ be the Vandermonde matrix with respect to distinct $c_1, \dots, c_K \in \mathcal{C}$:*

$$\begin{bmatrix} 1 & 1 & \cdots & 1 \\ c_1 & c_2 & \cdots & c_K \\ \vdots & \vdots & \ddots & \vdots \\ c_1^{K-1} & c_2^{K-1} & \cdots & c_K^{K-1} \end{bmatrix} .$$

*The K-***th Vandermonde norm** *with respect to the sampling set $\mathcal{C}$ is*

$$u_K := \max_{c_1,\dots,c_K \in \mathcal{C}} \|\det V_{c_1,\dots,c_K}\|_R \ .$$

*Let $A_{c_1,\dots,c_K}$ be the adjugate of $V_{c_1,\dots,c_K}$ (i.e., $A_{c_1,\dots,c_K} \cdot V_{c_1,\dots,c_K} = \det(V_{c_1,\dots,c_K}) \cdot I_K$). The K-**th inversion constant** with respect to the sampling set $\mathcal{C}$ is*

$$D_K := K \cdot \max_{c_1,\dots,c_K \in \mathcal{C}} \max_{i,j \in [K]} \|A_{c_1,\dots,c_K}[i,j]\|_R \quad .$$

5.2 Generalised Pedersen commitments

The Pedersen commitment scheme is an example of an invertible bilinear commitment scheme.

Definition 5.6. *Let $(\mathcal{G}, \mathcal{K})$ algorithms as in the Bilinear Relation Assumption. The* **generalised Pedersen commitment scheme** *is defined via the following algorithms.*

- $\mathsf{Ped.Setup}(1^\lambda, n)$: *Sample an argument-friendly bilinear module $\mathcal{M} = (R, M_L, M_R, M_T, e, \mathcal{C}, B_{\mathsf{C}}, B_{\max}) \leftarrow \mathcal{G}(1^\lambda, n+1)$ and a number r_{Ped}; set $\mathbb{K} := M_R$, $\mathbb{K}_0 := M_R^{r_{\mathsf{Ped}}}$, $\mathbb{M} := M_L(B_{\max})$, $\mathbb{R} := M_L(B_{\max})^{r_{\mathsf{Ped}}}$, and $\mathbb{C} := M_T$; output $\mathsf{pp} := (\mathcal{M}, \mathbb{K}_0 \times \mathbb{K}^n, \mathbb{M}^n, \mathbb{R}, \mathbb{C})$.*
- $\mathsf{Ped.KeyGen}(\mathsf{pp})$: *Sample $\mathsf{ck} \leftarrow \mathcal{K}(\mathcal{M})$, where $\mathsf{ck} \in M_R^{n+r_{\mathsf{Ped}}} = \mathbb{K}_0 \times \mathbb{K}^n$.*
- $\mathsf{Ped.Commit}(\mathsf{ck}; \mathsf{m}; \rho)$: *Parse ck as $(\mathsf{ck}_0, \mathsf{ckL}, \mathsf{ckR}) \in M_R^{r_{\mathsf{Ped}}} \times M_R^{n/2} \times M_R^{n/2}$, m as $(\mathsf{mL}, \mathsf{mR}) \in M_L(B_{\mathsf{C}})^{n/2} \times M_L(B_{\mathsf{C}})^{n/2}$, $\rho \in M_L(B_{\mathsf{C}})^{r_{\mathsf{Ped}}}$ and output $\mathsf{cm} := \langle \mathsf{mR}, \mathsf{ckL} \rangle + \langle \rho, \mathsf{ck}_0 \rangle \in M_T = \mathbb{C}$.*
- $\mathsf{Ped.Open}(\mathsf{ck}, \mathsf{m}, \rho, \mathsf{cm}, c)$: *Outputs 1 if $c \cdot \mathsf{cm} = \langle \mathsf{mR}, \mathsf{ckL} \rangle + \langle \rho, \mathsf{ck}_0 \rangle$, where $\mathsf{ck} = (\mathsf{ck}_0, \mathsf{ckL}, \mathsf{ckR}) \in M_R^{r_{\mathsf{Ped}}} \times M_R^{n/2} \times M_R^{n/2}$, $\mathsf{m} = (0, \mathsf{mR}) \in M_L(B_{\max})^{n/2} \times M_L(B_{\max})^{n/2}$, $\rho \in M_L(B_{\max})^{r_{\mathsf{Ped}}}$.*

Remark 5.7. The length parameter r_{Ped} produced by $\mathsf{CM.Setup}$ controls the size of the randomness space, to ensure that commitment randomness is sampled with sufficient entropy to produce hiding commitments in each setting. In the discrete logarithm and pairing settings, $r_{\mathsf{Ped}} = 1$. In the lattice setting, where $M_R = \left(\mathbb{Z}_q[X]/\langle X^d + 1 \rangle\right)^r$, one can show that $r_{\mathsf{Ped}} = 2r \log q$ gives statistically hiding commitments according to the leftover hash lemma [HILL99] (as used in e.g. the lattice-based arguments of [BBCPGL18]). Similarly, in the GUO setting $r_{\mathsf{Ped}} = \log \frac{2^\lambda |\mathbb{G}|}{B_{\max}}$ suffices, where $|\mathbb{G}|$ is an upper bound on the size of the group [BDFG20].

We assume that $\mathsf{Ped.ExtendCK}(\mathsf{ck})$ is the identity function and that $\mathsf{Ped.BilinearCommit}$ is equal to $\mathsf{Ped.Commit}$. In certain settings it is possible to consider different functions. For instance, in the GUO setting we can set the keyspace $\mathbb{K}_* = \mathbb{G}$ and define $\mathsf{Ped.ExtendCK}(\mathsf{G})$ as $(\mathsf{G}, \mathsf{G}^{B_{\max}}, \dots, \mathsf{G}^{B_{\max}^n})$.

Note that as well as being bilinear, the generalised Pedersen commitment scheme is also *R-homomorphic*, which is *not* implied by bilinearity in general.

Definition 5.8. *For a ring R,* CM *is R-***homomorphic** *if $\mathbb{M}, \mathbb{R}, \mathbb{C}$ are R-modules and for all $\lambda \in \mathbb{N}$ and all $n \in \mathbb{N}$, the commitment function $\mathsf{CM.Commit}\colon \mathbb{K}_* \times \mathbb{M}^n \times \mathbb{R} \to \mathbb{C}$ is an R-module homomorphism with respect to $\mathbb{M} \times \mathbb{R}$, i.e., for all $\mathsf{ck} \in \mathbb{K}_*$, all $\mathsf{m}, \mathsf{m}' \in \mathbb{M}^n$, all $\rho, \rho' \in \mathbb{R}$, and all $r \in R$,*

$$\begin{aligned} \mathsf{CM.Commit}\left(\mathsf{ck}; \mathsf{m} + \mathsf{m}'; \rho + \rho'\right) &= \mathsf{CM.Commit}\left(\mathsf{ck}; \mathsf{m}; \rho\right) + \mathsf{CM.Commit}\left(\mathsf{ck}; \mathsf{m}'; \rho'\right) \\ r \cdot \mathsf{CM.Commit}\left(\mathsf{ck}; \mathsf{m}; \rho\right) &= \mathsf{CM.Commit}\left(\mathsf{ck}; r \cdot \mathsf{m}; r \cdot \rho\right) \end{aligned}$$

Lemma 5.9. *The generalised Pedersen commitment scheme is a bilinear commitment scheme, assuming the Bilinear Relation Assumption. Moreover, in all settings from Section 5.1 (discrete logarithm, pairing, GUO, lattice) such that the relation $\mathcal{S}$ checks that $n \cdot (\gamma_R m(\mathcal{C}) D)^\ell \cdot B_{\mathsf{C}} < B_{\max}$, it is $(3, B_{\max}, C, D)$-invertible for the challenge space $\mathcal{C}$, where C and D are constants that depend on the setting (see proof).*

Proof. It is straightforward to verify that the generalised Pedersen commitment scheme satisfies the correctness, binding and hiding properties of a commitment scheme, based on the Bilinear Relation Assumption. The commitment scheme is also bilinear. Considering the left input arguments ckL and mL, we have

$$\begin{aligned}
&\mathsf{Ped.BilinearCommit}(\mathsf{ck}_0, \mathsf{ckL} + \mathsf{ckL}', \mathsf{ckR}; \mathsf{mL} + \mathsf{mL}', \mathsf{mR}; \rho + \rho') \\
&= \langle \mathsf{mR}, \mathsf{ckL} + \mathsf{ckL}' \rangle + \langle \rho + \rho', \mathsf{ck}_0 \rangle \\
&= (\langle \mathsf{mR}, \mathsf{ckL} \rangle + \langle \rho, \mathsf{ck}_0 \rangle) + (\langle \mathsf{mR}, \mathsf{ckL}' \rangle + \langle \rho', \mathsf{ck}_0 \rangle) \\
&= \mathsf{Ped.BilinearCommit}(\mathsf{ck}_0, \mathsf{ckL}, \mathsf{ckR}; \mathsf{mL}, \mathsf{mR}; \rho) + \mathsf{Ped.BilinearCommit}(\mathsf{ck}_0, \mathsf{ckL}', \mathsf{ckR}; \mathsf{mL}', \mathsf{mR}; \rho') \ ,
\end{aligned}$$

$$\begin{aligned}
&r \cdot \mathsf{Ped.BilinearCommit}(\mathsf{ck}_0, \mathsf{ckL}, \mathsf{ckR}; \mathsf{mL}, \mathsf{mR}; \rho) \\
&= r(\langle \mathsf{mR}, \mathsf{ckL} \rangle + \langle \rho, \mathsf{ck}_0 \rangle) \\
&= \langle r \cdot \mathsf{mR}, \mathsf{ckL} \rangle + \langle r \cdot \rho, \mathsf{ck}_0 \rangle \\
&= \mathsf{Ped.BilinearCommit}(\mathsf{ck}_0, \mathsf{ckL}, r \cdot \mathsf{ckR}; \mathsf{mL}, r \cdot \mathsf{mR}; r \cdot \rho) \\
&= \langle \mathsf{mR}, r \cdot \mathsf{ckL} \rangle + \langle r \cdot \rho, \mathsf{ck}_0 \rangle \\
&= \mathsf{Ped.BilinearCommit}(\mathsf{ck}_0, r \cdot \mathsf{ckL}, \mathsf{ckR}; r \cdot \mathsf{mL}, \mathsf{mR}; r \cdot \rho) \ .
\end{aligned}$$

The commitment scheme does not use the right input key ckR and and is trivially linear in ckR and mR.

The rest of the proof is to establish the invertibility of the commitment scheme in the various settings.

Let $(\mathsf{ckL}, \mathsf{ckL}', \mathsf{ckR}, \mathsf{ckR}', \mathsf{ck}_0) \in \mathbb{K}^L \times \mathbb{K}^L \times \mathbb{K}^L \times \mathbb{K}^L \times \mathbb{K}_0$ be the commitment keys as in the invertibility definition. Also, fix an output of $\mathcal{A}$: messages $(\mathsf{mL}^{(1)}, \mathsf{mR}^{(1)}), \ldots, (\mathsf{mL}^{(3)}, \mathsf{mR}^{(3)})$, pairwise-distinct challenges $r_1, \ldots, r_3 \in \mathcal{C}$, polynomial $q(X) = q_0 + q_1 X + q_2 X^2$ in $\mathbb{C}[X]$, and slackness $c \in \mathbb{Z}$. The experiment outputs 1 if there exists an $i \in \{1, 2, 3\}$ such that

$$\mathsf{Ped.Open}((\mathsf{ck}_0, \mathsf{ckL} + r_i \cdot \mathsf{ckL}', \mathsf{ckR} + r_i \cdot \mathsf{ckR}'), (\mathsf{mL}^{(i)}, \mathsf{mR}^{(i)}), 0, q(r_i), c) \neq 1.$$

Otherwise, we have that

$$c \cdot q(r_i) = c \cdot (q_0 + q_1 r_i + q_2 r_i^2) = \langle \mathsf{mR}^{(i)}, \mathsf{ckL} + r_i \cdot \mathsf{ckL}' \rangle \ . \tag{8}$$

Letting $V := \begin{pmatrix} r_i^2 & r_i & 1 \end{pmatrix}_{i \in \{1,2,3\}}$, we can find representations of a multiple of q_0, q_1, q_2 in terms of ckL and ckL' by computing

$$\begin{pmatrix} \det(V) c \cdot q_0 \\ \det(V) c \cdot q_1 \\ \det(V) c \cdot q_2 \end{pmatrix} := \operatorname{adj}(V) \cdot \begin{pmatrix} \langle \mathsf{mR}^{(1)}, \mathsf{ckL} + r_1 \cdot \mathsf{ckL}' \rangle \\ \langle \mathsf{mR}^{(2)}, \mathsf{ckL} + r_2 \cdot \mathsf{ckL}' \rangle \\ \langle \mathsf{mR}^{(3)}, \mathsf{ckL} + r_3 \cdot \mathsf{ckL}' \rangle \end{pmatrix}$$

In each setting, we can find representations of $C \cdot c \cdot q_0$, $C \cdot c \cdot q_1$, and $C \cdot c \cdot q_2$ in terms of ckL and ckL', where C is a constant chosen according to the setting. The parameter D is an upper bound on the norm of these representations and is also chosen differently in each setting.

Discrete logarithm setting. Since $\det(V) \in \mathbb{F}_q$, we can multiply with $\det(V)^{-1}$. In these settings, $C = 1$ and $D = 1$.

Bilinear pairing setting. Same as above.

Lattice setting. We multiply with $8 \det(V)^{-1}$ to recover the coefficients with $C = 8$. We have $\det(V) = (r_1 - r_2)(r_2 - r_3)(r_3 - r_1)$, and so $8 \det(V)^{-1} = \frac{2}{r_1 - r_2} \frac{2}{r_2 - r_3} \frac{2}{r_3 - r_1}$. From Lemma 5.3, we know that each factor has infinity norm 1, so $8 \det(V)^{-1}$ has infinity norm at most $\gamma_R^2 = d^2$, and the coefficients of ckL and

ckL' have norm at most $d^2\ D_3 \cdot \max(\|\mathsf{mR}^{(i)}\|)$. In this setting, $D = 2d^2 D_3$, where D_3 is the 3-rd inversion constant as in Definition 5.5.

GUO setting. In this setting, the keys $\mathsf{ckL}, \mathsf{ckL}'$ are vectors of powers of a group element $\mathsf{G} \in \mathbb{G}$, hence $\det(V) \cdot q_i = c_i\ \mathsf{G}$ for $i = 0, 1, 2$. It must be the case that we can compute $\frac{c_i}{\det(V)}\ \mathsf{G}$, since otherwise we would find a fractional root of G. Additionally, $\frac{c_i}{\det(V)}\ \mathsf{G}$ must be equal to q_i, otherwise we would have found an element of known order. Finding a fractional root of G breaks the strong RSA assumption and finding an element of known order breaks the adaptive root assumption (see [BFS20, Appendix A]). Note that the coefficients of ckL and ckL' have norm at most $D_3 \cdot \max(\|\mathsf{mR}^{(i)}\|)$. In this setting, $C = 1$ and $D = 2D_3$. Note that in this setting $\mathcal{C} = \mathbb{Z}(\frac{p-1}{2})$ and D is a function of p.

Finally, in all settings we can write

$$C \cdot cq(X) = \langle \pi_0(X), \mathsf{ckL} \rangle + \langle \pi_1(X), \mathsf{ckL}' \rangle \ , \tag{9}$$

where $\pi_0(X) = a_0 + b_0 X + c_0 X^2$ and $\pi_1(X) = a_1 + b_1 X + c_1 X^2$. So, it holds that

$$C \cdot c(q(1) + q(-1)) = 2\ \langle M_R, \mathsf{ckL} + r_j \cdot \mathsf{ckL}' \rangle$$

with $M_R = (a_0 + c_0, a_1 + c_1)$. Notice that in all settings $\|M_R\| \leq D \cdot \max(\|\mathsf{mR}^{(i)}\|)$. Since generalised Pedersen commitments are homomorphic, we have

$$\mathsf{Ped.Open}((\mathsf{ck}_0, \mathsf{ckL}\|\mathsf{ckL}', \mathsf{ckR}\|\mathsf{ckR}'), (0, M_R), 0, 1/2 \cdot (q(1) + q(-1)), Cc) = 1\ .$$

Thus, since the message M_R is efficiently computable, the invertibility experiment outputs 1. $\square$

5.3 Scalar-product commitments

We define a commitment scheme which includes the scalar-product of two committed messages. For this commitment scheme, we assume that M_L of the argument-friendly bilinear module is a ring, so that the scalar-product of two messages in M_L^n is well-defined (which is not the case over a general module).

Definition 5.10. *Let $(\mathcal{G}, \mathcal{K})$ algorithms as in the Bilinear Relation Assumption. The* **scalar-product commitment scheme** *is defined via the following algorithms.*

- $\mathsf{SP.Setup}(1^\lambda, n)$: *Sample an argument-friendly bilinear module* $(R, M_L, M_R, M_T, e, \mathcal{C}, B_\mathsf{C}, B_{\max}) \leftarrow \mathcal{G}(1^\lambda, n+1)$*; set* $\mathbb{K} := M_R$, $\mathbb{K}_0 := M_R^{3r_\mathsf{Ped}+1}$, $\mathbb{M} := M_L(B_{\max})$, $\mathbb{R} := M_L(B_{\max})^{3r_\mathsf{Ped}}$, *and* $\mathbb{C} := M_T$*; output* $\mathsf{pp} := (\mathcal{M}, \mathbb{K}^n \times \mathbb{K}_0, \mathbb{M}^n, \mathbb{R}, \mathbb{C})$.
- $\mathsf{SP.KeyGen}(\mathsf{pp})$: *Sample* $\mathsf{ck} \leftarrow \mathcal{K}(\mathcal{M})$ *where* $\mathsf{ck} \in M_R^{n+1} = \mathbb{K}^n \times \mathbb{K}_0$.
- $\mathsf{SP.Commit}(\mathsf{ck}; \mathsf{m}; \rho)$: *Parse* ck *as* $(\mathsf{ck}_0, \mathsf{ck}_{\mathit{rand}}, \mathsf{ckL}_{\mathit{rand}}, \mathsf{ckR}_{\mathit{rand}}, \mathsf{ckL}, \mathsf{ckR}) \in M_R \times M_R^{r_\mathsf{Ped}} \times M_R^{r_\mathsf{Ped}} \times M_R^{r_\mathsf{Ped}} \times M_R^{n/2} \times M_R^{n/2}$, m *as* $(\mathsf{mL}, \mathsf{mR}) \in M_L(B_\mathsf{C})^{n/2} \times M_L(B_\mathsf{C})^{n/2}$, ρ *as* $(\rho_0, \rho\mathsf{L}, \rho\mathsf{R}) \in M_R^{r_\mathsf{Ped}} \times M_R^{r_\mathsf{Ped}} \times M_R^{r_\mathsf{Ped}}$, *and output* $\mathsf{cm} := (\langle \mathsf{mR}, \mathsf{ckL} \rangle + \langle \rho\mathsf{R}, \mathsf{ckL}_{\mathit{rand}} \rangle, \langle \mathsf{mL}, \mathsf{ckR} \rangle + \langle \rho\mathsf{L}, \mathsf{ckR}_{\mathit{rand}} \rangle, \langle \mathsf{mL}, \mathsf{mR} \rangle \mathsf{ck}_0 + \langle \rho_0, \mathsf{ck}_{\mathit{rand}} \rangle)$.
- $\mathsf{SP.Open}(\mathsf{ck}, \mathsf{m}, \rho, \mathsf{cm}, c)$: *Outputs* 1 *if it holds that* $c^2 \mathsf{cm} = (c\langle \mathsf{mR}, \mathsf{ckL} \rangle + c\langle \rho\mathsf{R}, \mathsf{ckL}_{\mathit{rand}} \rangle, c\langle \mathsf{mL}, \mathsf{ckR} \rangle + c\langle \rho\mathsf{L}, \mathsf{ckR}_{\mathit{rand}} \rangle, \langle \mathsf{mL}, \mathsf{mR} \rangle \mathsf{ck}_0 + \langle \rho_0, \mathsf{ck}_{\mathit{rand}} \rangle)$, *where* $\mathsf{ck} = (\mathsf{ck}_0, \mathsf{ck}_{\mathit{rand}}, \mathsf{ckL}_{\mathit{rand}}, \mathsf{ckR}_{\mathit{rand}}, \mathsf{ckL}, \mathsf{ckR}) \in M_R \times M_R^{r_\mathsf{Ped}} \times M_R^{r_\mathsf{Ped}} \times M_R^{r_\mathsf{Ped}} \times M_R^{n/2} \times M_R^{n/2}$, $\mathsf{m} = (\mathsf{mL}, \mathsf{mR}) \in M_L(B_{\max})^{n/2} \times M_L(B_{\max})^{n/2}$, $\rho = (\rho_0, \rho\mathsf{L}, \rho\mathsf{R}) \in M_R^{r_\mathsf{Ped}} \times M_R^{r_\mathsf{Ped}} \times M_R^{r_\mathsf{Ped}}$.

Remark 5.11. The assumption that M_L is a ring excludes the pairing setting, where M_L is a group. However, we capture this case by modifying the commitment scheme as follows. First, we use different message and key space for each half the message and key respectively. In particular, if $(\mathsf{ckL}, \mathsf{ckR}) \in M_R^{n/2} \times M_L^{n/2}$ and $(\mathsf{mL}, \mathsf{mR}) \in M_L^{n/2} \times M_R^{n/2}$, then both the Pedersen commitments $\langle \mathsf{mR}, \mathsf{ckL} \rangle$ and $\langle \mathsf{mL}, \mathsf{ckR} \rangle$, and the scalar-product $\langle \mathsf{mL}, \mathsf{mR} \rangle$ are well-defined.

In order to make the commitment scheme hiding, we modify the third component corresponding to the commitment of $\langle \mathsf{mL}, \mathsf{mR} \rangle$ by using an ElGamal encryption $(\langle \mathsf{mL}, \mathsf{mR} \rangle + \rho_0 \cdot \mathsf{ck}_{\mathsf{rand}}, \rho_0 \cdot \mathsf{G}_T)$, where G_T is the generator of the target group $\mathbb{G}_T$, $\mathsf{ck}_{\mathsf{rand}} = c \cdot \mathsf{G}_T \in M_T$ is public and c is part of the key, and $\rho_0 \leftarrow \mathbb{F}_q$. The commitment scheme is binding and hiding assuming that DDH is hard for the group $\mathbb{G}_T$ and the bilinear module assumption holds for $(M_L, M_R) = (\mathbb{G}_1, \mathbb{G}_2)$ and for $(M_L, M_R) = (\mathbb{G}_2, \mathbb{G}_1)$.

As in Pedersen commitments, the definition can be generalised to include different $\mathsf{CM.ExtendCK(ck)}$ functions.

Lemma 5.12. *The scalar-product commitment scheme is a bilinear commitment scheme, assuming the Bilinear Relation Assumption. Moreover, in all settings from Section 5.1 (discrete logarithm, pairing, GUO, lattice) such that $\mathcal{S}$ checks that $n\gamma_{M_L}(n \cdot (\gamma_R m(\mathcal{C}) D)^\ell \cdot B_{\mathsf{C}})^2 \leq B_{\max}$ [12] it is $(4, B_{\max}, C, D)$-invertible for the challenge space $\mathcal{C}$, where C and D are constants that depend on the setting (see proof).*

Proof. It is straightforward to verify that the scalar-product commitment scheme satisfies the correctness, hiding, and binding properties of a commitment scheme, based on the Bilinear Relation Assumption. The commitment scheme is also bilinear.

The rest of the proof is to establish the invertibility of the commitment scheme in the various settings.

Let $(\mathsf{ckL}, \mathsf{ckL}', \mathsf{ckR}, \mathsf{ckR}', \mathsf{ck}_0) \in \mathbb{K}^L \times \mathbb{K}^L \times \mathbb{K}^L \times \mathbb{K}^L \times \mathbb{K}_0$ be the commitment keys as in the invertibility definition. Also, let messages $(\mathsf{mL}^{(1)}, \mathsf{mR}^{(1)})$,..., $(\mathsf{mL}^{(4)}, \mathsf{mR}^{(4)})$, pairwise-distinct challenges $r_1, \dots, r_4 \in \mathcal{C}$, polynomial $q(X) = q_0 + q_1 X + q_2 X^2$ in $\mathbb{C}[X]$ and $c \in \mathbb{Z}$ be the output of $\mathcal{A}$. The experiment outputs 1 if there exists an $i \in \{1, 2, 3, 4\}$ such that

$$\mathsf{SP.Open}((\mathsf{ck}_0, \mathsf{ckL} + r_i \cdot \mathsf{ckL}', \mathsf{ckR} + r_i \cdot \mathsf{ckR}'), (\mathsf{mL}^{(i)}, \mathsf{mR}^{(i)}), 0, q(r_i), c) \neq 1.$$

Otherwise, we have that

$$c^2(q_0 + q_1 r_i + q_2 r_i^2) = (c\langle \mathsf{mR}^{(i)}, \mathsf{ckL} + r_i \cdot \mathsf{ckL}' \rangle, c\langle \mathsf{mL}^{(i)}, \mathsf{ckR} + r_i \cdot \mathsf{ckR}' \rangle, \langle \mathsf{mR}^{(i)}, \mathsf{mL}^{(i)} \rangle \mathsf{ck}_0). \quad (10)$$

As in the proof of Lemma 5.9, we can find representations of $C^2 \cdot c^2 q_0$, $C^2 \cdot c^2 q_1$, and $C^2 \cdot c^2 q_2$ in terms of ckL, ckL', ckR, and ckR'. The constants D and C in each setting are as follows.

Discrete logarithm setting. $C = 1$ and $D = 1$.

Pairing setting. As above.

Lattice setting. $C = 8$ and $D = 3d^2 D_3$. The constant D is used to upper bound the norm of messages appearing later in the proof.

GUO setting. $C = 1$ and $D = 3p^2 D_3$. The constant D is used to upper bound the norm of messages appearing later in the proof.

As in the proof of Lemma 5.9, for all settings we can write

$$C^2 \cdot c^2 q(X) = (C \cdot c(\langle \pi_{L,0}(X), \mathsf{ckL} \rangle + \langle \pi_{L,1}(X), \mathsf{ckL}' \rangle), C \cdot c(\langle \pi_{R,0}(X), \mathsf{ckR} \rangle + \langle \pi_{R,1}(X), \mathsf{ckR}' \rangle), \pi(X) \mathsf{ck}_0) \ ,$$

[12] The $\mathcal{S}$ guarantees that the bilinear relation assumption must be hard for messages up to the norm of $\langle \mathsf{mL}, \mathsf{mR} \rangle$ to imply invertibility.

where $\pi_{L,0}$,$\pi_{L,1}$,$\pi_{R,0}$,$\pi_{R,1}$,π are of the form $a + bX + cX^2$. By comparing the coefficients of ckL and ckL' in Equation (10), we conclude that $\pi_{L,0}(r_j)r_i = \pi_{L,1}(r_i)$ for each i, since otherwise for $m_1 = (\pi_{L,0}(r_i)||\pi_{L,1}(r_i))$ and $m_2 = (C\mathsf{mR}^{(i)}||Cr_i \cdot \mathsf{mR}^{(i)})$ it holds that

$$\mathsf{SP.BilinearCommit}(\mathsf{ck}_0, \mathsf{ckL}||\mathsf{ckL}', \mathsf{ckR}||\mathsf{ckR}'; 0, m_1; 0) = \mathsf{SP.BilinearCommit}(\mathsf{ck}_0, \mathsf{ckL}||\mathsf{ckL}', \mathsf{ckR}||\mathsf{ckR}'; 0, m_2; 0) \ ,$$

which would break the binding property of the commitment scheme. Note that $\|m_1\|, \|m_2\| \le D \max(\|\mathsf{mR}^{(i)}\|)$. A similar argument applies to ckR and ckR'.

Due to the degree bounds on the polynomials $\pi_{L,0}$ and $\pi_{L,1}$, the fact that this expression holds for the 4 different values of r_i implies that $\pi_{L,0}(X)$ is of the form $a + bX$. Further, for each of the four values of r_i, we have that $C\mathsf{mR}^{(i)} = \pi_{L,0}(r_i)$, otherwise for the two distinct messages $m_1 = (\pi_{L,0}(r_i)||r_i \cdot \pi_{L,0}(r_i))$ and $m_2 = (C\mathsf{mR}^{(i)}||Cr_i \cdot \mathsf{mR}^{(i)})$ it holds that

$$\mathsf{SP.BilinearCommit}(\mathsf{ck}_0, \mathsf{ckL}||\mathsf{ckL}', \mathsf{ckR}||\mathsf{ckR}'; 0, m_1; 0) = \mathsf{SP.BilinearCommit}(\mathsf{ck}_0, \mathsf{ckL}||\mathsf{ckL}', \mathsf{ckR}||\mathsf{ckR}'; 0, m_2; 0) \ ,$$

which breaks the binding property of the commitment scheme. A similar argument applies to $\pi_{R,0}$ and $\pi_{R,1}$. For the polynomial π, due to the degree bound and the fact that $\pi(r_i) = C^2\langle \mathsf{mR}^{(i)}, \mathsf{mL}^{(i)}\rangle = \langle \pi_{L,0}(r_i), \pi_{R,0}(r_i)\rangle$ for the 4 different values of r_i, it holds that $\pi(X) = \langle \pi_{L,0}(X), \pi_{R,0}(X)\rangle$.

Finally, $C^2 \cdot c^2 q(X) = (C \cdot c\, \pi_{L,0}(X)(\mathsf{ckL} + X \cdot \mathsf{ckL}'), C \cdot c\, \pi_{R,0}(X)(\mathsf{ckR} + X \cdot \mathsf{ckR}'), \langle \pi_{L,0}(X), \pi_{R,0}(X)\rangle \mathsf{ck}_0)$ and

$$C^2 \cdot c^2(q(1) + q(-1)) = 2(\langle C \cdot c\, M_R, \mathsf{ckL} + r_j \cdot \mathsf{ckL}'\rangle, \langle C \cdot c\, M_L, \mathsf{ckR} + r_j \cdot \mathsf{ckR}'\rangle, \langle M_L, M_R\rangle \mathsf{ck}_0)$$

with $M_R = (a_L||b_L)$ and $M_L = (a_R||b_R)$, where $\pi_{L,0}(X) = a_L + b_L X$ and $\pi_{R,0}(X) = a_R + b_R X$. Notice that in all settings $\|M_R\| \le D \max(\|\mathsf{mR}^{(i)}\|)$ and $\|M_L\| \le D \max(\|\mathsf{mL}^{(i)}\|)$. Hence, it holds that

$$\mathsf{SP.Open}((\mathsf{ck}_0, \mathsf{ckL}||\mathsf{ckL}', \mathsf{ckR}||\mathsf{ckR}'), (0, M_R), 0, 1/2 \cdot (q(1) + q(-1)), Cc) = 1 \ ,$$

Since the message M_L, M_R are efficiently computable, the invertibility experiment outputs 1. □

5.4 Compressed scalar-product commitments

In some settings, it is possible to compress the three distinct parts of scalar-product commitments into one. Then, the scalar-product protocol is similar to a Pedersen commitment, where the last coordinate corresponds to the scalar-product of the "first" and "second" half of the message. As in the scalar-product commitments, we restrict M_L to be a ring.

Definition 5.13 (compressed scalar-product commitments). *Let $(\mathcal{G}, \mathcal{K})$ algorithms as in the Bilinear Relation Assumption. The* **compressed scalar-product commitment scheme** *is defined via the following algorithms.*

- $\mathsf{CSP.Setup}(1^\lambda, n)$: *Sample an argument-friendly bilinear module $(R, M_L, M_R, M_T, e, \mathcal{C}, B_\mathsf{C}, B_{\max}) \leftarrow \mathcal{G}(1^\lambda, n+2)$ and a number r_Ped; set $\mathbb{K} := M_R$, $\mathbb{K}_0 := M_R^{r_\mathsf{Ped}+1}$, $\mathbb{M} := M_L(B_{\max})$, $\mathbb{R} := M_L(B_{\max})^{r_\mathsf{Ped}}$, and $\mathbb{C} = M_T$; output* $\mathsf{pp} := (\mathcal{M}, \mathbb{K}^n \times \mathbb{K}_0, \mathbb{M}^n, \mathbb{R}, \mathbb{C})$.
- $\mathsf{CSP.KeyGen}(\mathsf{pp})$: *Sample* $\mathsf{ck} \leftarrow \mathcal{K}(\mathcal{M})$ *where* $\mathsf{ck} \in M_R^n = \mathbb{K}_0 \times \mathbb{K}^n$.
- $\mathsf{CSP.Commit}(\mathsf{ck}; \mathsf{m}; \rho)$: *Parse* ck *as* $(\mathsf{ck}_0, \mathsf{ck}_{\mathit{rand}}, \mathsf{ckL}, \mathsf{ckR}) \in M_R \times M_R^{r_\mathsf{Ped}} \times M_R^{n/2} \times M_R^{n/2}$, m *as* $(\mathsf{mL}, \mathsf{mR}) \in M_L(B_\mathsf{C})^{n/2} \times M_L(B_\mathsf{C})^{n/2}$, $\rho \in M_L(B_\mathsf{C})^{r_\mathsf{Ped}}$, *and output* $\mathsf{cm} := \langle \mathsf{mR}, \mathsf{ckL}\rangle + \langle \mathsf{mL}, \mathsf{ckR}\rangle + \langle \mathsf{mL}, \mathsf{mR}\rangle \cdot \mathsf{ck}_0 + \langle \rho, \mathsf{ck}_{\mathit{rand}}\rangle$.

- $\mathsf{CSP.Open}(\mathsf{ck}, \mathsf{m}, \rho, \mathsf{cm}, c)$: *Output* 1 *if* $c^2 \cdot \mathsf{cm} = c\langle \mathsf{mR}, \mathsf{ckL}\rangle + c\langle \mathsf{mL}, \mathsf{ckR}\rangle + \langle \mathsf{mL}, \mathsf{mR}\rangle \cdot \mathsf{ck}_0 + c\langle \rho, \mathsf{ck}_{\mathit{rand}}\rangle$, *where* $\mathsf{ck} = (\mathsf{ck}_0, \mathsf{ck}_{\mathit{rand}}, \mathsf{ckL}, \mathsf{ckR}) \in M_R \times M_R^{r_{\mathsf{Ped}}} \times M_R^{n/2} \times M_R^{n/2}$, $\mathsf{m} = (\mathsf{mL}, \mathsf{mR}) \in M_L(B_{\max})^{n/2} \times M_L(B_{\max})^{n/2}$, $\rho \in M_L(B_{\max})^{r_{\mathsf{Ped}}}$.

The parameter r_{Ped} depends on the setting as in Section 5.2.

In the pairing instantiation, M_L is not a ring and using the slight modification of Section 5.3 does not result in a Pedersen-like commitment, so we do not consider the pairing setting in this section.

Lemma 5.14. *The compressed scalar-product commitment scheme is a bilinear commitment scheme, assuming the Bilinear Relation Assumption. Moreover, in the discrete logarithm, GUO, lattice settings from Section 5.1 such that $\mathcal{S}$ checks that $n\gamma_{M_L}(n \cdot (\gamma_R m(\mathcal{C})D)^{\ell} \cdot B_{\mathsf{C}})^2 \leq B_{\max}$, it is $(4, B_{\max}, C, D)$-invertible for challenge space $\mathcal{C}$, where C and D are constants that depend on the setting (see proof).*

The proof is similar to Lemma 5.12, so we provide only a proof sketch that highlights the main differences.

Proof sketch. The proof follows the same steps as in the proof of Lemma 5.12. We sketch the differences between the two proofs.

First, when comparing the coefficients of ckL, ckL', ckL, ckL', and ck_0 in contrast to the corresponding proof of Lemma 5.12 we define the messages $(m_{1,L}, m_{1,R}, m_{1,\mathsf{ck}_0}) = (\pi_{R,0}(r_j)||\pi_{R,1}(r_j), \pi_{L,0}(r_j)||\pi_{L,1}(r_j), \pi(r_j))$ and $(m_{2,L}, m_{2,R}, m_{2,\mathsf{ck}_0}) = (C \cdot \mathsf{mL}^{(j)}||C \cdot r_j \cdot \mathsf{mL}^{(j)}, C \cdot \mathsf{mR}^{(j)}||C \cdot r_j \cdot \mathsf{mR}^{(j)}, \langle C\mathsf{mL}^{(j)}, C\mathsf{mR}^{(j)}\rangle)$. If the messages $(m_{1,L}, m_{1,R}, m_{1,\mathsf{ck}_0})$ and $(m_{2,L}, m_{2,R}, m_{2,\mathsf{ck}_0})$ are distinct, then we get a break of the binding property of the generalised Pedersen commitment.

Similarly, we argue that $C\mathsf{mR}^{(j)} = \pi_{L,0}(r_j)$, $C\mathsf{mL}^{(j)} = \pi_{R,0}(r_j)$, and $\pi(r_j) = \langle \pi_{R,0}(r_j), \pi_{L,0}(r_j)\rangle$. We have to define the constant D appropriately in all settings so that the norms of the messages are less than $B_{\max}$.

Finally, $C^2 \cdot c^2 q(X) = C \cdot c\pi_{L,0}(X)(\mathsf{ckL} + X \cdot \mathsf{ckL}') + C \cdot c\pi_{R,0}(X)(\mathsf{ckR} + X \cdot \mathsf{ckR}') + \langle \pi_{L,0}(X), \pi_{R,0}(X)\rangle \mathsf{ck}_0$ and

$$C^2 \cdot c^2(q(1) + q(-1)) = 2(\langle C \cdot c \cdot M_R, \mathsf{ckL} + r_j \cdot \mathsf{ckL}'\rangle + \langle C \cdot c \cdot M_L, \mathsf{ckR} + r_j \cdot \mathsf{ckR}'\rangle + \langle M_L, M_R\rangle \mathsf{ck}_0$$

with $M_R = (a_L||b_L)$ and $M_L = (a_R||b_R)$, where $\pi_{L,0}(X) = a_L + b_L X$ and $\pi_{R,0}(X) = a_R + b_R X$. Notice that in all settings $\|M_R\| \leq D \max(\|\mathsf{mR}^{(i)}\|)$ and $\|M_L\| \leq D \max(\|\mathsf{mL}^{(i)}\|)$. Hence, it holds that

$$\mathsf{CSP.Open}((\mathsf{ck}_0, \mathsf{ck}_{\mathsf{rand}}, \mathsf{ckL}||\mathsf{ckL}', \mathsf{ckR}||\mathsf{ckR}'), (0, M_R), 0, 1/2 \cdot (q(1) + q(-1)), Cc) = 1 \ ,$$

Since the messages M_L, M_R are efficiently computable, the invertibility experiment outputs 1. □

6 Succinct argument for scalar products over rings

In this section, we give scalar-product protocols for rings defined over suitable argument-friendly bilinear modules. We will cover instances of scalar-products such as $\langle \vec{a}, \vec{b} \rangle = t \bmod I$ over M_L where M_L is itself a ring, and I is a suitable ideal of M_L. We start by discussing a simple rejection sampling algorithms used to sample random masks in our protocols.

6.1 Rejection sampling

Recall that to hide a message with norm at most B, the randomness for the generalised Pedersen commitment is sampled uniformly at random from $\mathbb{R}(B)$.

We state a lemma about sampling random masks over the integers.

Lemma 6.1. *Let $B, \kappa \in \mathbb{Z}$, and $x \in [-B, B]$. Consider the procedure that samples $y \leftarrow [-\kappa B, \kappa B]$, sets $z = x + y$ and outputs z if $z \in [-(\kappa-1)B, (\kappa-1)B]$, and outputs $\perp$ otherwise.*

The procedure outputs $\perp$ with probability at most $1/\kappa$. Conditioned on not outputting $\perp$, the distribution of $x + y$ is uniform in $[-(\kappa-1)B, (\kappa-1)B]$.

Proof. We have $x + y \in [x - \kappa B, x + \kappa B]$. The symmetric difference between $[x - \kappa B, x + \kappa B]$ and $[-(\kappa-1)B, (\kappa-1)B]$ has $2B$ elements. Thus, the probability of outputting $\perp$ is at most $\frac{2B}{2\kappa B+1} \leq \frac{B}{\kappa B} = \frac{1}{\kappa}$. □

The procedure in Lemma 6.1 allows us to hide secrets by adding randomly-sampled masks in our GUO and lattice instantiations, which use $\mathbb{Z}$-modules equipped with the infinity norm. For our discrete logarithm and pairing instantiations, which use cyclic modules with trivial norms, it is clear that the uniform distribution already perfectly hides secret values. Notice that if the procedure in Lemma 6.1 is applied to each element of a vector $\vec{x} \in \mathbb{Z}^N$, then the probability of outputting $\perp$ is at most N/κ. In the GUO setting, we set $\kappa = 2^\lambda$ for negligible completeness error. In the lattice setting, since the value of κ will feature in the norm bounds of extracted solutions, we cannot choose κ to be exponentially large, as the SIS problem is easy in such parameter regimes. In this setting, the scalar product argument reasons about vectors of length n defined over the ring $\mathbb{Z}[X]/\langle X^d + 1 \rangle$. Therefore, we set $\kappa = O(dn)$, which has size polynomial in the security parameter, for a constant probability of aborting.

One can reduce the completeness error in protocols by modifying the procedure in Lemma 6.1 so that it samples several random masks $\vec{y}_1, \ldots, \vec{y}_r$ for each $\vec{x}$, and avoids outputting $\perp$ by using any possible $\vec{y}_i$, so that the probability of outputting $\perp$ is at most $(N/\kappa)^r$.

6.2 Construction

Definition 6.2. *The* **committed scalar product** *relation $\mathcal{R}_{\mathrm{comSP}}(c, B_{\mathsf{C}})$ is the set of tuples*

$$(\mathbb{x}, \mathbb{w}) = \Big((\mathcal{M}, r_{\mathsf{Ped}}, I, \vec{\mathsf{G}}, \vec{\mathsf{G}}_0, \vec{\mathsf{H}}, \vec{\mathsf{H}}_0, \mathsf{U}, \vec{\mathsf{U}}_0, \mathsf{C}_0, \mathsf{C}_1, \mathsf{C}_t), (\vec{a}, \vec{b}, \vec{\rho}_0, \vec{\rho}_1, \vec{\rho}_t)\Big) \quad , \textit{where}$$

- *$\mathcal{M} = (R, M_L, M_R, M_T, e, \mathcal{C}, B_{\mathsf{C}}, B_{\max})$ is an argument-friendly bilinear module and $I \subseteq M_L$ is an ideal;*
- *$c \in R$;*
- *$\vec{a}, \vec{b} \in M_L^n(B_{\mathsf{C}})$;*

- $\vec{\mathsf{G}}, \vec{\mathsf{H}} \in M_R^n$;
- $\vec{\rho}_0, \vec{\rho}_1, \vec{\rho}_t \in M_L^{r_{\mathsf{Ped}}}(B_{\mathsf{C}})$;
- $\mathsf{U} \in M_R$, $\vec{\mathsf{G}}_0, \vec{\mathsf{H}}_0, \vec{\mathsf{U}}_0 \in M_R^{r_{\mathsf{Ped}}}$, *and* $\mathsf{C}_0, \mathsf{C}_1, \mathsf{C}_t \in M_T$

satisfying $\mathsf{Ped.Open}((\vec{\mathsf{G}}_0, \vec{\mathsf{G}}), \vec{a}, \vec{\rho}_0, \mathsf{C}_0, c) = 1$, $\mathsf{Ped.Open}((\vec{\mathsf{H}}_0, \vec{\mathsf{H}}), \vec{b}, \vec{\rho}_1, \mathsf{C}_1, c) = 1$ *and* $\mathsf{Ped.Open}((\vec{\mathsf{U}}_0, \mathsf{U}), \langle \vec{a}, \vec{b} \rangle \bmod I, \vec{\rho}_t, \mathsf{C}_t, c^2) = 1$.

Theorem 6.3. *Suppose that we have the following ingredients:*

- *an argument-friendly bilinear module* $(R, M_L, M_R, M_T, e, \mathcal{C}, B_{\mathsf{C}}, B_{\max})$ *satisfying the bilinear relation assumption, and such that* M_L *is a ring, and* $\mathcal{C}$ *is a sampling-set for* M_T*;*
- *an ideal* I *such that* $\mathcal{C}$ *is a strong sampling set for* $R_* = M_L/I$.

Then there is an honest-verifier zero-knowledge argument of knowledge for $\mathcal{R}_{\mathrm{comSP}}$ *over* R_**, supporting instances with* $n = 2^\ell$*, with communication complexity* $O(\log n_{\mathrm{row}})$ *elements of* M_T*, round complexity* $O(\log n)$*, prover and verifier complexity* $O(n)$ *bilinear product operations and* $O(n)$ *operations in* M_L.

Remark 6.4. The choice of ideal I used depends on the cryptographic instantiation of Theorem 6.3. In the lattice setting and GUO setting, we use $I = p\mathbb{Z}$ for some prime p. In each case, the prime p must be large enough to ensure that the challenge space $\mathcal{C}$ is a sampling set modulo p. In the discrete logarithm setting, we use $I = \{0\}$. In our analysis, we will set $m(R_*)$ to be a bound on the norms of representatives of R_* in M_L. This makes sense, because in each case, R_* is a finite ring. We will set $B_{\mathsf{C}} = m(R_*)$.

We state various bounds which will be used in our construction.

$$\begin{aligned} m_1 &:= \gamma_R m(\mathcal{C}) m(R_*) & & \\ m_2 &:= \gamma_R m(\mathcal{C})(m_3 + m(R_*)) & m_3 &:= \gamma_R n m(R_*)^2 \\ m_4 &:= 2\kappa\gamma_R n m(R_*) m_1 & m_5 &:= \gamma_R^2 m(\mathcal{C})^2 m_3 + \gamma_R m(\mathcal{C}) m_4 \end{aligned}$$

Construction 6.5. We construct an interactive argument for the relation $\mathcal{R}_{\mathrm{comSP}}(1, B_{\mathsf{C}})$. The prover $\mathbf{P}$ takes as input an instance $\mathbb{x} = (\mathcal{M}, r_{\mathsf{Ped}}, I, \vec{\mathsf{G}}, \vec{\mathsf{G}}_0, \vec{\mathsf{H}}, \vec{\mathsf{H}}_0, \mathsf{U}, \vec{\mathsf{U}}_0, \mathsf{C}_0, \mathsf{C}_1, \mathsf{C}_t)$, and witness $\mathbb{w} = (\vec{a}, \vec{b})$, while the verifier $\mathbf{V}$ takes as input the instance $\mathbb{x}$.

- The prover samples random masks $\vec{y}_a, \vec{y}_b \leftarrow M_L(\kappa m_1)^n$. The prover computes $v_2 := \langle \vec{a}, \vec{b} \rangle$, $v_1 := \langle \vec{a}, \vec{y}_b \rangle + \langle \vec{y}_a, \vec{b} \rangle$ and $v_0 := \langle \vec{y}_a, \vec{y}_b \rangle$. The prover samples random mask $h \leftarrow M_L(\kappa m_2)$.

  The prover samples commitment randomness and computes commitments as follows.

$$\begin{aligned} \vec{\sigma}_0 &\leftarrow \mathbb{R}(\kappa m_1) \ , & \mathsf{C}_2 &:= \mathsf{Ped.Commit}(\vec{\mathsf{G}}_0, \vec{\mathsf{G}}, \vec{y}_a; \vec{\sigma}_0) \ , \\ \vec{\sigma}_1 &\leftarrow \mathbb{R}(\kappa m_1) \ , & \mathsf{C}_3 &:= \mathsf{Ped.Commit}(\vec{\mathsf{H}}_0, \vec{\mathsf{H}}, \vec{y}_b; \vec{\sigma}_1) \ , \\ \vec{\sigma}_5 &\leftarrow \mathbb{R}(\kappa m_2) \ , & \mathsf{C}_7 &:= \mathsf{Ped.Commit}(\vec{\mathsf{U}}_0, \mathsf{U}, h; \vec{\sigma}_5) \ , \\ \vec{\sigma}_4 &\leftarrow \mathbb{R}(\kappa m_3) \ , & \mathsf{C}_6 &:= \mathsf{Ped.Commit}(\vec{\mathsf{U}}_0, \mathsf{U}, v_2; \vec{\sigma}_4) \ , \\ \vec{\sigma}_2 &\leftarrow \mathbb{R}(\kappa m_4) \ , & \mathsf{C}_4 &:= \mathsf{Ped.Commit}(\vec{\mathsf{U}}_0, \mathsf{U}, v_1; \vec{\sigma}_2) \ , \\ \vec{\sigma}_3 &\leftarrow \mathbb{R}(\kappa m_5) \ , & \mathsf{C}_5 &:= \mathsf{Ped.Commit}(\vec{\mathsf{U}}_0, \mathsf{U}, v_0; \vec{\sigma}_3) \ . \end{aligned} \tag{11}$$

  The prover sends $\mathsf{C}_7, \mathsf{C}_2, \mathsf{C}_3, \mathsf{C}_6, \mathsf{C}_4, \mathsf{C}_5 \in M_T$, and $h' := h \bmod I$ to the verifier.

- The verifier $\mathbf{V}$ sends random challenges $\gamma \leftarrow \mathcal{C}$ to the prover.
- The prover computes

$$
\begin{aligned}
\vec{z}_a &:= \gamma\vec{a} + \vec{y}_a \ , & \vec{z}_b &:= \gamma\vec{b} + \vec{y}_b \ , \\
\vec{\rho}_0' &:= \gamma\vec{\rho}_0 + \vec{\sigma}_0 \ , & \vec{\rho}_1' &:= \gamma\vec{\rho}_1 + \vec{\sigma}_1 \ , \\
\vec{\rho}_t' &:= \gamma^2\vec{\sigma}_4 + \gamma\vec{\sigma}_2 + \vec{\sigma}_3 \ , & \bar{v} &:= \gamma(\langle\vec{a},\vec{b}\rangle - (\langle\vec{a},\vec{b}\rangle \bmod I)) + h \ , \\
\vec{\sigma}' &:= \gamma(\vec{\sigma}_4 - \vec{\rho}_t) + \vec{\sigma}_5 \ .
\end{aligned}
$$

The prover aborts if any of the following conditions are not satisfied:

$$
\begin{aligned}
\|\vec{z}_a\|_{M_L} &\le (\kappa-1)m_1 \ , & \|\vec{z}_b\|_{M_L} &\le (\kappa-1)m_1 \ , \\
\|\vec{\rho}_0'\|_{\mathbb{R}} &\le (\kappa-1)m_1 \ , & \|\vec{\rho}_1'\|_{\mathbb{R}} &\le (\kappa-1)m_1 \ , \\
\|\vec{\rho}_t'\|_{\mathbb{R}} &\le (\kappa-1)m_5 \ , & \|\bar{v}\|_{M_L} &\le (\kappa-1)m_2 \ , \\
\|\vec{\sigma}'\|_{\mathbb{R}} &\le (\kappa-1)m_2 \ .
\end{aligned}
\tag{12}
$$

The prover sends $\vec{\rho}_0'$, $\vec{\rho}_1'$, $\vec{\rho}_t'$, $\bar{v}$, and $\vec{\sigma}'$ to the verifier.

- The verifier checks that

$$
\begin{aligned}
\|\vec{\rho}_0'\|_{\mathbb{R}} &\le (\kappa-1)m_1 \ , & \|\vec{\rho}_1'\|_{\mathbb{R}} &\le (\kappa-1)m_1 \ , \\
\|\vec{\rho}_t'\|_{\mathbb{R}} &\le (\kappa-1)m_5 \ , & \|\bar{v}\|_{M_L} &\le (\kappa-1)m_2 \ , \\
\|\vec{\sigma}'\|_{\mathbb{R}} &\le (\kappa-1)m_2 \ .
\end{aligned}
\tag{13}
$$

The verifier computes $\mathsf{T}' := \big(\gamma\mathsf{C}_0 + \mathsf{C}_2,\ \gamma\mathsf{C}_1 + \mathsf{C}_3,\ \gamma^2\mathsf{C}_t + \gamma\mathsf{C}_4 + \mathsf{C}_5\big)$.

This allows the verifier to compute

$$
\mathsf{T} := \mathsf{T}' - \Big(\langle\vec{\rho}_0', \vec{\mathsf{G}}_0\rangle,\ \langle\vec{\rho}_1', \vec{\mathsf{H}}_0\rangle,\ \langle\vec{\rho}_t', \vec{\mathsf{U}}_0\rangle\Big) = \mathsf{SP.Commit}((\vec{\mathsf{G}}, \vec{\mathsf{G}}_0, \vec{\mathsf{H}}, \vec{\mathsf{H}}_0, \mathsf{U}, \vec{\mathsf{U}}_0), (\vec{z}_a, \vec{z}_b), 0)
$$

from T' by removing the contributions of bases $\vec{\mathsf{G}}_0$, $\vec{\mathsf{H}}_0$ and $\vec{\mathsf{U}}_0$ from T'.

The prover and the verifier run the opening protocol of Construction 4.7 for $\mathcal{R}_{\mathrm{SC}}(1, (\kappa-1)m_1)$ with instance $\mathbb{x} = (\mathsf{SP}, \mathsf{pp}_{\mathsf{SP}}, \vec{\mathsf{G}}, \vec{\mathsf{G}}_0, \vec{\mathsf{H}}, \vec{\mathsf{H}}_0, \mathsf{U}, \vec{\mathsf{U}}_0, \mathsf{T})$ and witness $\mathbb{w} = (\vec{z}_a, \vec{z}_b)$ to show that

$$
\mathsf{SP.Open}((\vec{\mathsf{G}}, \vec{\mathsf{H}}, \mathsf{U}, \vec{\mathsf{U}}_0), (\vec{a}, \vec{b}, t), 0, \mathsf{T}, 1) = 1.
$$

The verifier checks that $\mathsf{Ped.Open}((\mathsf{U}, \vec{\mathsf{U}}_0), \bar{v}, \vec{\sigma}', \gamma(\mathsf{C}_6 - \mathsf{C}_t) + \mathsf{C}_7, 1) = 1$ and $\bar{v} \bmod I = h'$.

6.3 Proof of Theorem 6.3

The prover and verifier efficiency of Construction 6.5 are directly inherited from Construction 4.7.

Lemma 6.6. *The prover in Construction 6.5 performs the following operations: $O(n)$ bilinear operations between M_L and M_R; $O(n)$ operations in M_L; and $O(n)$ additions in M_T.*

Lemma 6.7. *The verifier in Construction 6.5 performs the following operations: $O(n)$ bilinear operations between M_L and M_R; $O(n)$ operations in M_L; and $O(n)$ additions in M_T.*

Lemma 6.8. *Construction 6.5 is complete for $\mathcal{R}_{\text{comSP}}(1, B_{\mathsf{C}})$ with completeness error $O(n/\kappa)$.*

Proof. Suppose that $((\mathcal{M}, r_{\mathsf{Ped}}, I, \vec{\mathsf{G}}, \vec{\mathsf{G}}_0, \vec{\mathsf{H}}, \vec{\mathsf{H}}_0, \mathsf{U}, \vec{\mathsf{U}}_0, \mathsf{C}_0, \mathsf{C}_1, \mathsf{C}_t), (\vec{a}, \vec{b}, \vec{\rho}_0, \vec{\rho}_1, \vec{\rho}_t)) \in \mathcal{R}_{\text{comSP}}(1, B_{\mathsf{C}})$.

First, we argue the prover aborts with probability at most $O(n/\kappa)$. Using the discussion after Lemma 6.1, the probability that any of the inequalities of Equation (12) are satisfied is at most $O(1/\kappa)$ for each inequality. In this case, the verifier checks of Equation (13) are all satisfied, and we have $\|\vec{z}_a\|_{M_L}, \|\vec{z}_b\|_{M_L} \leq (\kappa-1)m_1$.

Next, we argue that $(\mathbb{x}', \mathbb{w}') = \Big((\mathcal{M}, \vec{\mathsf{G}}, \vec{\mathsf{G}}_0, \vec{\mathsf{H}}, \vec{\mathsf{H}}_0, \mathsf{U}, \vec{\mathsf{U}}_0, \mathsf{T}), (\vec{z}_a, \vec{z}_b)\Big) \in \mathcal{R}_{\text{SC}}(1, m_1)$. By construction, we have $\langle \vec{z}_a, \vec{z}_b \rangle = \gamma^2 v_2 + \gamma v_1 + v_0$. Thus,

$$\begin{aligned}
\mathsf{T} &= \mathsf{T}' - \Big(\langle \vec{\rho}_0', \vec{\mathsf{G}}_0 \rangle,\ \langle \vec{\rho}_1', \vec{\mathsf{H}}_0 \rangle,\ \langle \vec{\rho}_t', \vec{\mathsf{U}}_0 \rangle\Big) \\
&= \Big(\gamma \mathsf{C}_0 + \mathsf{C}_2 - \langle \vec{\rho}_0', \vec{\mathsf{G}}_0 \rangle,\ \gamma \mathsf{C}_1 + \mathsf{C}_3 - \langle \vec{\rho}_1', \vec{\mathsf{H}}_0 \rangle,\ \gamma^2 \mathsf{C}_t + \gamma \mathsf{C}_4 + \mathsf{C}_5 - \langle \vec{\rho}_t', \vec{\mathsf{U}}_0 \rangle\Big) \\
&= \Big(\langle \vec{z}_a, \vec{\mathsf{G}} \rangle,\ \langle \vec{z}_b, \vec{\mathsf{H}} \rangle,\ \langle \vec{z}_a, \vec{z}_b \rangle \cdot \mathsf{U}\Big) \\
&= \mathsf{SP.Commit}((\vec{\mathsf{G}}, \vec{\mathsf{G}}_0, \vec{\mathsf{H}}, \vec{\mathsf{H}}_0, \mathsf{U}, \vec{\mathsf{U}}_0), (\vec{z}_a, \vec{z}_b), 0) \ .
\end{aligned}$$

By completeness of the generalised Pedersen commitment scheme, $(\vec{z}_a, \vec{z}_b)$ is a valid opening of T with respect to the appropriate bases, and will give a witness to $\mathcal{R}_{\text{SC}}(1, m_1)$. Construction 4.7 is complete and so the verifier of this subprotocol will accept.

It remains to show that the verifier checks that $\mathsf{Ped.Open}((\mathsf{U}, \vec{\mathsf{U}}_0), \bar{v}, \vec{\sigma}', \gamma(\mathsf{C}_6 - \mathsf{C}_t) + \mathsf{C}_7, 1) = 1$ and $\bar{v} \bmod I = h$ is accepted. Since

$$\begin{aligned}
\mathsf{C}_6 &= \mathsf{Ped.Commit}(\vec{\mathsf{U}}_0, \mathsf{U}, \langle \vec{a}, \vec{b} \rangle, \vec{\sigma}_4) \ , \\
\mathsf{C}_t &= \mathsf{Ped.Commit}(\vec{\mathsf{U}}_0, \mathsf{U}, \langle \vec{a}, \vec{b} \rangle \bmod I, \vec{\rho}_t) \ , \\
\mathsf{C}_7 &= \mathsf{Ped.Commit}(\vec{\mathsf{U}}_0, \mathsf{U}, h; \vec{\sigma}_5) \ , \\
\bar{v} &= \gamma(\langle \vec{a}, \vec{b} \rangle - (\langle \vec{a}, \vec{b} \rangle \bmod I)) + h \ , \\
\vec{\sigma}' &= \gamma(\vec{\sigma}_4 - \vec{\rho}_t) + \vec{\sigma}_5 \ ,
\end{aligned}$$

and by the homomorphic property of the Pedersen commitment scheme, it follows that $\gamma(\mathsf{C}_6 - \mathsf{C}_t) + \mathsf{C}_7 = \mathsf{Ped.Commit}(\vec{\mathsf{U}}_0, \mathsf{U}; \bar{v}, \vec{\sigma}')$. Therefore, the verifier's checks are accepting by the completeness property of the generalised Pedersen commitment scheme, and by reducing $\bar{v}$ modulo I. $\square$

Lemma 6.9. *Assuming the bilinear-relation assumption, there exists an efficient algorithm that given an instance $(\mathcal{M}, r_{\mathsf{Ped}}, I, \vec{\mathsf{G}}, \vec{\mathsf{G}}_0, \vec{\mathsf{H}}, \vec{\mathsf{H}}_0, \mathsf{U}, \vec{\mathsf{U}}_0, \mathsf{C}_0, \mathsf{C}_1, \mathsf{C}_t)$, and a $(3, 4^\ell)$-tree of accepting transcripts either extracts a witness $(\vec{a}, \vec{b})$ to $\mathcal{R}_{\text{comSP}}(c_{\mathsf{SP}}, B_{\mathsf{SP}})$ with c_{SP} and B_{SP} derived in the proof. For soundness we must have $\gamma_R n B_{\mathsf{SP}}^2$ be at most $B_{\max}$.*

Proof. Recall that Lemma 5.14 shows that compressed scalar-product commitments can be inverted with 4 openings. By Lemma 4.16, there is an efficient algorithm which takes the 4^ℓ-subtree of accepting transcripts for Construction 4.7 applied to $\mathcal{R}_{\text{SC}}(1, (\kappa-1)m_1)$ and produces either a non-trivial bilinear relation, or produces a witness $(\vec{z}_a, \vec{z}_b)$ to $\mathcal{R}_{\text{SC}}(c, B')$, for $c \in R$ which is not a zero-divisor for M_T or R_*, and $B' := n \cdot (\gamma_R m(\mathcal{C}) D)^\ell \cdot (\kappa-1)m_1$ satisfying

$$c^2 \cdot \mathsf{T} = \Big(c\langle \vec{z}_a, \vec{\mathsf{G}} \rangle,\ c\langle \vec{z}_b, \vec{\mathsf{H}} \rangle,\ \langle \vec{z}_a, \vec{z}_b \rangle \cdot \mathsf{U}\Big) \ .$$

This is done for each value of γ in the $(3, 4^\ell)$-tree of accepting transcripts.

By definition of T and T', we have

$$\begin{aligned} & c^2\left(\gamma \mathsf{C}_0 + \mathsf{C}_2,\ \gamma \mathsf{C}_1 + \mathsf{C}_3,\ \gamma^2 \mathsf{C}_6 + \gamma \mathsf{C}_4 + \mathsf{C}_5\right) \\ & = \Big(c \cdot \mathsf{Ped.Commit}(\vec{\mathsf{G}}_0, \vec{\mathsf{G}}, \vec{z}_a, c\vec{\rho}_0'),\ c \cdot \mathsf{Ped.Commit}(\vec{\mathsf{H}}_0, \vec{\mathsf{H}}, \vec{z}_b, c\vec{\rho}_1'),\ \mathsf{Ped.Commit}(\vec{\mathsf{U}}_0, \mathsf{U}, \langle \vec{z}_a, \vec{z}_b\rangle, c^2\vec{\rho}_t')\Big) \ . \end{aligned} \tag{14}$$

For each γ, we know that a copy of Equation (14) holds. Examining each component of Equation (14) gives

$$c(\gamma \mathsf{C}_1 + \mathsf{C}_3) = \mathsf{Ped.Commit}(\vec{\mathsf{H}}_0, \vec{\mathsf{H}}, \vec{z}_b, c\vec{\rho}_1') \ , \tag{15}$$

$$c(\gamma \mathsf{C}_0 + \mathsf{C}_2) = \mathsf{Ped.Commit}(\vec{\mathsf{G}}_0, \vec{\mathsf{G}}, \vec{z}_a, c\vec{\rho}_0') \ , \tag{16}$$

$$\text{and} \quad c^2(\gamma^2 \mathsf{C}_6 + \gamma \mathsf{C}_4 + \mathsf{C}_5) = \mathsf{Ped.Commit}(\vec{\mathsf{U}}_0, \mathsf{U}, \langle \vec{z}_a, \vec{z}_b\rangle, c^2\vec{\rho}_t') \ . \tag{17}$$

Now, considering Equation (15) and Equation (16) for two distinct challenge values γ and γ', taking the linear combinations $(1, -1)$ and $(\gamma, -\gamma')$, and setting $c_* := \gamma - \gamma'$, we obtain openings $\vec{a}$, $\vec{b}$, $\vec{y}_a$, $\vec{y}_b$, $\vec{\rho}_0$, $\vec{\rho}_1$, $\vec{\sigma}_0$, $\vec{\sigma}_1$ with norms at most $\gamma_R D_2 B'$ satisfying

$$\begin{aligned} c_* c \mathsf{C}_0 &= \langle \vec{a}, \vec{\mathsf{G}}\rangle + c\langle \vec{\rho}_0, \vec{\mathsf{G}}_0\rangle \\ c_* c \mathsf{C}_2 &= \langle \vec{y}_a, \vec{\mathsf{G}}\rangle + c\langle \vec{\sigma}_0, \vec{\mathsf{G}}_0\rangle \\ c_* c \mathsf{C}_1 &= \langle \vec{b}, \vec{\mathsf{H}}\rangle + c\langle \vec{\rho}_1, \vec{\mathsf{H}}_0\rangle \\ c_* c \mathsf{C}_3 &= \langle \vec{y}_b, \vec{\mathsf{H}}\rangle + c\langle \vec{\sigma}_1, \vec{\mathsf{H}}_0\rangle \ . \end{aligned}$$

We either have that $c_*\vec{z}_a = \gamma\vec{a} + \vec{y}_a$, $c_*\vec{z}_b = \gamma\vec{b} + \vec{y}_b$, $c_*\vec{\rho}_0' = \gamma\vec{\rho}_0 + \vec{\sigma}_0$ and $c_*\vec{\rho}_1' = \gamma\vec{\rho}_1 + \vec{\sigma}_1$ or we can use Equation (15) or Equation (16) to find a non-trivial bilinear relation and break the binding property of the Pedersen commitment scheme. Multiplying Equation (17) by c_*^2 and substituting in these values gives

$$(c_*c)^2(\gamma^2 \mathsf{C}_6 + \gamma \mathsf{C}_4 + \mathsf{C}_5) = \gamma^2\langle \vec{a}, \vec{b}\rangle \cdot \mathsf{U} + \gamma(\langle \vec{a}, \vec{y}_b\rangle + \langle \vec{y}_a, \vec{b}\rangle) \cdot \mathsf{U} + \langle \vec{y}_a, \vec{y}_b\rangle \cdot \mathsf{U} + (c_*c)^2\langle \vec{\rho}_t', \vec{\mathsf{U}}_0\rangle \tag{18}$$

Let $\mathbf{A} := \begin{pmatrix} \gamma_1^2 & \gamma_1 & 1 \\ \gamma_2^2 & \gamma_2 & 1 \\ \gamma_3^2 & \gamma_3 & 1 \end{pmatrix}$ and let $c'' := \det(\mathbf{A})$. Taking Equation (18) with respect to three distinct values of γ, taking the three values of $\vec{\rho}_t'$, and multiplying the linear system by the adjugate of $\mathbf{A}$ (to solve for the coefficient of γ^2) and then multiplying by c'' yields $\vec{\sigma}_4$ such that

$$(c_*cc'')^2 \mathsf{C}_6 = \langle c''\vec{a}, c''\vec{b}\rangle \cdot \mathsf{U} + (c_*cc'')^2\langle \vec{\sigma}_4, \vec{\mathsf{U}}_0\rangle \tag{19}$$

which means that the message $\langle c''\vec{a}, c''\vec{b}\rangle$ and randomness $(c_*cc'')^2\vec{\sigma}_4$ give a relaxed opening to C_6 with relaxation factor $c_{\mathsf{SP}} := (c_*cc'')^2$. The norm of $\vec{\sigma}_4$ is at most $(\kappa - 1)\gamma_R D_3 m_5$.

Finally, we show that C_t has a relaxed opening which is related to the opening of C_6 in M_L, considered modulo I. The verifier's check that $\mathsf{Ped.Open}((\mathsf{U}, \vec{\mathsf{U}}_0), \bar{v}, \vec{\sigma}', \gamma(\mathsf{C}_6 - \mathsf{C}_t) + \mathsf{C}_7, 1) = 1$ implies that

$$\gamma(\mathsf{C}_6 - \mathsf{C}_t) + \mathsf{C}_7 = \mathsf{Ped.Commit}(\mathsf{U}, \vec{\mathsf{U}}_0; \bar{v}, \vec{\sigma}') \ . \tag{20}$$

We also have that $\bar{v} \bmod I = h'$. Considering Equation (20) for challenges γ and γ', (with opening $\bar{v}'$ and $\vec{\sigma}''$ for γ') and subtracting one from the other, we find openings O and h, and randomness ϕ and $\vec{\sigma}_5$ such that

$$c_*(\mathsf{C}_6 - \mathsf{C}_t) = \mathsf{Ped.Commit}(\vec{\mathsf{U}}_0, \mathsf{U}; O; \phi) \ , \tag{21}$$

$$c_* \mathsf{C}_7 = \mathsf{Ped.Commit}(\vec{\mathsf{U}}_0, \mathsf{U}; h; \vec{\sigma}_5) \ , \tag{22}$$

where the norms of O, h, ϕ and $\vec{\sigma}_5$ are at most $\gamma_R D_2 m_2$. Furthermore, multiplying Equation (20) with c_* and substituting the openings of Equation (21) and Equation (22), we have that either $c_* \bar{v} = \gamma O + h$, $c_* \vec{\sigma}' = \gamma \phi + \vec{\sigma}_5$, $c_* \bar{v}' = \gamma' O + h$ and $c_* \vec{\sigma}'' = \gamma' \phi + \vec{\sigma}_5$, or we can use Equation (20) to find a non-trivial bilinear relation. Since $\bar{v} = \bar{v}' = h' \bmod I$, we have $\gamma O + h = \gamma' O + h \bmod I$. Taking linear combinations of these two equations shows that $c_* O = 0 \bmod I$ and $c_* h = c_* h' \bmod I$. Since $\mathcal{C}$ is a sampling set for M_L/I, we have $O = 0 \bmod I$ and $h = h' \bmod I$.

Rearranging Equation (21) and multiplying by $c_*(cc'')^2$ shows that

$$\begin{aligned}(c_* cc'')^2 \mathsf{C}_t &= (c_* cc'')^2 \mathsf{C}_6 - \mathsf{Ped.Commit}(\vec{\mathsf{U}}_0, \mathsf{U}; c_*(cc'')^2 O; c_*(cc'')^2 \phi) \\ &= \mathsf{Ped.Commit}(\mathsf{U}, \vec{\mathsf{U}}_0; \langle c''\vec{a}, c''\vec{b}\rangle, (c_* cc'')^2 \vec{\sigma}_4) - \mathsf{Ped.Commit}(\vec{\mathsf{U}}_0, \mathsf{U}; c_*(cc'')^2 O; c_*(cc'')^2 \phi)\end{aligned}$$

This means that message $\langle c''\vec{a}, c''\vec{b}\rangle - c_*(cc'')^2 O$ and randomness $(c_* cc'')^2 \vec{\sigma}_4 - c_*(cc'')^2 \phi$ give a relaxed opening to t, and since $O = 0 \bmod I$, C_t has a relaxed opening which is equal to the opening (modulo I) of C_6.

The norms of $\langle c''\vec{a}, c''\vec{b}\rangle - c_*(cc'')^2 O$ and $(c_* cc'')^2 \vec{\sigma}_4 - c_*(cc'')^2 \phi$ are at most $B_{\mathsf{SP}} = n \gamma_R^5 u_3^2 D_2^2 B'^2 + \gamma_R^5 u_2 u_3^2 D_2 \|c\|_R^2 m_2 + \gamma_R^7 u_2 u_3 \|c\|_R^2 (\kappa - 1) D_3 m_5$. This gives a witness to $\mathcal{R}_{\mathrm{comSP}}(c_{\mathsf{SP}}, B_{\mathsf{SP}})$, with $c_{\mathsf{SP}} = (c_* cc'')^2$. □

Lemma 6.10. *Construction 6.5 has semi-honest verifier zero-knowledge.*

Proof. We give a simulator for Construction 6.5.

1. Start simulating $\mathbf{V}$.
2. Sample openings $\vec{z}_a, \vec{z}_b \leftarrow M_L((\kappa-1)m_1)^n$ and $\bar{v} \leftarrow M_L((\kappa-1)m_2)^n$. Set $h' = \bar{v} \bmod I$.
3. Sample opening randomness $\vec{\rho}'_0, \vec{\rho}'_1 \leftarrow \mathbb{R}((\kappa-1)m_1)$, $\vec{\sigma}' \leftarrow \mathbb{R}((\kappa-1)m_2)$ and $\vec{\rho}'_t \leftarrow \mathbb{R}((\kappa-1)m_5)$.
4. Compute $\mathsf{T} = \Big(\langle \vec{z}_a, \vec{\mathsf{G}}\rangle,\ \langle \vec{z}_b, \vec{\mathsf{H}}\rangle,\ \langle \vec{z}_a, \vec{z}_b\rangle \cdot \mathsf{U}\Big)$. Compute $\mathsf{T}' = \mathsf{T} - \Big(\langle \vec{\rho}'_0, \vec{\mathsf{G}}_0\rangle,\ \langle \vec{\rho}'_1, \vec{\mathsf{H}}_0\rangle,\ \langle \vec{\rho}'_t, \vec{\mathsf{U}}_0\rangle\Big)$.
5. Compute commitments $\mathsf{C}_2, \mathsf{C}_3, \mathsf{C}_6, \mathsf{C}_4$ as follows:
$$\begin{aligned} \vec{\sigma}_0 &\leftarrow \mathbb{R}((\kappa-1)m_1) \ , & \mathsf{C}_2 &:= \mathsf{Ped.Commit}(\vec{\mathsf{G}}_0, \vec{\mathsf{G}}, \vec{0}; \vec{\sigma}_0) \ , \\ \vec{\sigma}_1 &\leftarrow \mathbb{R}((\kappa-1)m_1) \ , & \mathsf{C}_3 &:= \mathsf{Ped.Commit}(\vec{\mathsf{H}}_0, \vec{\mathsf{H}}, \vec{0}; \vec{\sigma}_1) \ , \\ \vec{\sigma}_4 &\leftarrow \mathbb{R}((\kappa-1)m_3) \ , & \mathsf{C}_6 &:= \mathsf{Ped.Commit}(\vec{\mathsf{U}}_0, \mathsf{U}, \vec{0}; \vec{\sigma}_4) \ , \\ \vec{\sigma}_2 &\leftarrow \mathbb{R}((\kappa-1)m_4) \ , & \mathsf{C}_4 &:= \mathsf{Ped.Commit}(\vec{\mathsf{U}}_0, \mathsf{U}, \vec{0}; \vec{\sigma}_2) \ . \end{aligned}$$
6. Compute $\mathsf{C}_5 = \mathsf{T}' - \big(\gamma \mathsf{C}_0 + \mathsf{C}_2,\ \gamma \mathsf{C}_1 + \mathsf{C}_3,\ \gamma^2 \mathsf{C}_t - \gamma \mathsf{C}_4\big)$.
   Compute $\mathsf{C}_7 = \mathsf{Ped.Commit}(\mathsf{U}, \vec{\mathsf{U}}_0; \bar{v}, \vec{\sigma}') - \gamma(\mathsf{C}_6 - \mathsf{C}_t)$.
7. Run the sumcheck protocol of Construction 4.7 for commitment $\mathsf{SP.Commit}()$ with instance $\mathbb{x} = (\mathcal{M}, \vec{\mathsf{G}}, \vec{\mathsf{G}}_0, \vec{\mathsf{H}}, \vec{\mathsf{H}}_0, \mathsf{U}, \vec{\mathsf{U}}_0, \mathsf{T})$ and witness $\mathbb{w} = (\vec{z}_a, \vec{z}_b)$.
8. Abort with probability equal to that of the honest prover, outputting only the commitments and h' in that case.

Now we argue that the simulated transcript is statistically indistinguishable from a transcript generated by an honest prover. Firstly, note that the simulator aborts with the same probability as the honest prover.

Based on Lemma 6.1, we have that openings $\vec{z}_a, \vec{z}_b \in M_L((\kappa-1)m_1)^n$, and $\bar{v} \in M_L((\kappa-1)m_2)^n$, and opening randomness $\vec{\rho}_0', \vec{\rho}_1' \leftarrow \mathbb{R}((\kappa-1)m_1)$, $\vec{\sigma}' \leftarrow \mathbb{R}((\kappa-1)m_2)$ and $\vec{\rho}_t' \leftarrow \mathbb{R}((\kappa-1)m_5)$, are uniformly distributed in a real protocol execution where the prover does not abort. This implies that the simulator produces distributions of these values which are statistically indistinguishable from those in a real protocol execution.

By the hiding property of the generalised Pedersen commitment scheme, $\mathsf{C}_2, \mathsf{C}_3, \mathsf{C}_6, \mathsf{C}_4$ are statistically indistinguishable from honestly generated commitments.

In both a real or simulated execution, all other simulated values are now fully determined: C_7 and C_5 by linear relations used in the protocol, h' by reducing $\bar{v}$ modulo I, and the rest of the transcript by executing Construction 4.7. The result follows. $\square$

7 Succinct argument for R1CS over rings

We construct zero-knowledge arguments for Rank 1 Constraint Satisfiability (R1CS) over rings. First, we define the R1CS relation. Then, we give the formal statement of Theorem 2 and the description of the interactive argument. In Lemma 7.4, Lemma 7.5, Lemma 7.6 we show that the protocol is complete, knowledge sound, and zero-knowledge. In Lemma 7.7, we give bounds on the number of operations performed by the verifier and prover.

Definition 7.1 (R1CS). *The indexed relation $\mathcal{R}_{\mathrm{R1CS}}$ is the set of all triples*

$$(\mathbb{i}, \mathbb{x}, \mathbb{w}) = \big((A, B, C), (R_*, A, B, C, M, n_{\mathrm{row}}, n_{\mathrm{col}}, n_{\mathrm{in}}, x), \vec{w}\big)$$

where R_* is a finite ring, A, B, C are matrices in $R_*^{n_{\mathrm{row}} \times n_{\mathrm{col}}}$, each having at most M non-zero entries, $x \in R_*^{n_{\mathrm{in}}}$ and $\vec{w}$ are vectors over R_*, and $\vec{z} := (x, \vec{w}) \in R_*^{n_{\mathrm{col}}}$ is a vector such that $A\vec{z} \circ B\vec{z} = C\vec{z}$. (Here "$\circ$" denotes the entry-wise product between two vectors.)

In this section, we give R1CS arguments for rings R_* defined over suitable argument-friendly bilinear modules. We will cover instances where M_L is itself a ring, I is a suitable ideal of M_L, and $R_* = M_L/I$. For notational simplicity, we handle the case where $n_{\mathrm{row}} = n_{\mathrm{col}}$.

Theorem 7.2. *Suppose that we have the following ingredients:*

- *an argument-friendly bilinear module $(R, M_L, M_R, M_T, e, \mathcal{C}, B_{\mathsf{C}}, B_{\max})$ satisfying the bilinear relation assumption, and with M_L a ring, and $\mathcal{C}$ is a sampling-set for M_T;*
- *an ideal I such that $\mathcal{C}$ is a strong sampling set for $R_* = M_L/I$;*
- *a zero-knowledge argument for $\mathcal{R}_{\mathrm{comSP}}$ (see Definition 6.2) over M_L and suitable ideal I, with constant c in the knowledge extractor (see Lemma 6.9).*

Then there is a zero-knowledge argument of knowledge for $\mathcal{R}_{\mathrm{R1CS}}$ over R_*, supporting instances with $n_{\mathrm{row}} = n_{\mathrm{col}} = 2^\ell$, with communication complexity $O(\log n_{\mathrm{row}})$ elements of M_T, round complexity $O(\log n_{\mathrm{row}})$, prover and verifier complexity $O(n_{\mathrm{row}})$ bilinear product operations and $O(n_{\mathrm{row}} + M)$ operations in M_L.

Construction 7.3. We construct an interactive argument for the relation $\mathcal{R}_{\mathrm{R1CS}}$. The prover $\mathbf{P}$ takes as input an instance $\mathbb{x} = (R_*, A, B, C, M, n_{\mathrm{row}}, n_{\mathrm{col}}, n_{\mathrm{in}}, x)$, and witness $\mathbb{w} = \vec{w}$, while the verifier $\mathbf{V}$ takes as input the instance $\mathbb{x}$.

- The prover $\mathbf{P}$ constructs a fully satisfying assignment of the R1CS instance $\vec{z} := (x, \vec{w}) \in M_L^{n_{\mathrm{col}}}$, and computes the vectors $\vec{z}_A := A\vec{z} \bmod I$, $\vec{z}_B := B\vec{z} \bmod I$, $\vec{z}_C := C\vec{z} \bmod I$ in $M_L^{n_{\mathrm{row}}}$.

  The prover computes commitments to messages $\vec{z}$, $\vec{z}_A$ and $\vec{z}_B$ as follows. Recall that elements of R_* have representatives with norms at most $m(R_*)$ in M_L.

$$\begin{aligned} \vec{\rho} &\leftarrow \mathbb{R}(m(R_*)) & \mathsf{C}_{\vec{z}} &\leftarrow \mathsf{Ped.Commit}(\vec{\mathsf{G}}_0, \vec{\mathsf{G}}, \vec{z}, \vec{\rho}) \ , \\ \vec{\rho}_A &\leftarrow \mathbb{R}(m(R_*)) & \mathsf{C}_{\vec{z}_A} &\leftarrow \mathsf{Ped.Commit}(\vec{\mathsf{G}}_0, \vec{\mathsf{G}}, \vec{z}_A, \vec{\rho}_A) \ , \\ \vec{\rho}_B &\leftarrow \mathbb{R}(m(R_*)) & \mathsf{C}_{\vec{z}_B} &\leftarrow \mathsf{Ped.Commit}(\vec{\mathsf{G}}_0, \vec{\mathsf{G}}, \vec{z}_B, \vec{\rho}_B) \ . \end{aligned}$$

  The prover sends $(\mathsf{C}_{\vec{z}}, \mathsf{C}_{\vec{z}_A}, \mathsf{C}_{\vec{z}_B}) \in M_T^3$ to the verifier.

- The verifier $\mathbf{V}$ sends uniformly random challenge vectors $(r_1, \ldots, r_{\log n_{\text{row}}}) \in \mathcal{C}^{\log n_{\text{row}}}$ and $(y_1, \ldots, y_{\log n_{\text{row}}}) \in \mathcal{C}^{\log n_{\text{row}}}$ to the prover.
- The prover $\mathbf{P}$ and verifier $\mathbf{V}$ compute the vectors $\vec{r} := \bigotimes_{i=1}^{\log n_{\text{row}}}(1, r_i)$, $\vec{y} := \bigotimes_{i=1}^{\log n_{\text{row}}}(1, y_i)$,

$$\vec{r}_A := \vec{r}^{\intercal} A \bmod I \ , \qquad \vec{r}_B := \vec{r}^{\intercal} B \bmod I \ , \qquad \vec{r}_C := \vec{r}^{\intercal} C \bmod I \in M_L^{n_{\text{col}}}.$$

  The prover $\mathbf{P}$ computes the elements and commitments

$$\begin{aligned} \alpha &:= \langle \vec{r}, \vec{z}_A \rangle \bmod I, & \vec{\rho}_\alpha &\leftarrow \mathbb{R}(m(R_*)) & \mathsf{C}_\alpha &\leftarrow \mathsf{Ped.Commit}(\vec{\mathsf{U}}_0, \mathsf{U}, \alpha, \vec{\rho}_\alpha), \\ \alpha'' &:= \langle \vec{r} \circ \vec{y}, \vec{z}_A \rangle \bmod I, & \vec{\rho}''_\alpha &\leftarrow \mathbb{R}(m(R_*)) & \mathsf{C}''_\alpha &\leftarrow \mathsf{Ped.Commit}(\vec{\mathsf{U}}_0, \mathsf{U}, \alpha'', \vec{\rho}''_\alpha), \\ \beta &:= \langle \vec{r}, \vec{z}_B \rangle \bmod I, & \vec{\rho}_\beta &\leftarrow \mathbb{R}(m(R_*)) & \mathsf{C}_\beta &\leftarrow \mathsf{Ped.Commit}(\vec{\mathsf{U}}_0, \mathsf{U}, \beta, \vec{\rho}_\beta), \\ \gamma &:= \langle \vec{r}, \vec{z}_C \rangle \bmod I, & \vec{\rho}_\gamma &\leftarrow \mathbb{R}(m(R_*)) & \mathsf{C}_\gamma &\leftarrow \mathsf{Ped.Commit}(\vec{\mathsf{U}}_0, \mathsf{U}, \gamma, \vec{\rho}_\gamma), \\ \vec{z}'_A &= \vec{r} \circ \vec{z}_A \bmod I & \vec{\rho}'_A &\leftarrow \mathbb{R}(m(R_*)) & \mathsf{C}'_{\vec{z}_A} &\leftarrow \mathsf{Ped.Commit}(\vec{\mathsf{H}}_0, \vec{\mathsf{H}}, \vec{z}'_A, \vec{\rho}'_A). \end{aligned}$$

  and sends the message $(\mathsf{C}_\alpha, \mathsf{C}''_\alpha, \mathsf{C}_\beta, \mathsf{C}_\gamma, \mathsf{C}'_{\vec{z}_A}) \in M_T^5$.
- The verifier samples a uniformly random challenge vector $\vec{s} \in \mathcal{C}^{n_{\text{in}}}$, and pads with zeroes to get $\vec{s}' \in \mathcal{C}^{n_{\text{col}}}$. The verifier sends $\vec{s}$ to the prover. The verifier computes $\sigma := \langle x, \vec{s} \rangle \bmod I$.
- The prover $\mathbf{P}$ and verifier $\mathbf{V}$ engage in several scalar-product sub-protocols, in parallel. Each scalar-product sub-protocol has instance $\mathbb{x}$ of the form $(\mathcal{M}, \vec{\mathsf{G}}, \vec{\mathsf{G}}_0, \vec{\mathsf{H}}, \vec{\mathsf{H}}_0, \mathsf{U}, \vec{\mathsf{U}}_0, X, Y, Z)$. The witnesses, values of X, Y and Z, and the purpose of each sub-protocol, are specified in the following table. For each protocol, the witnesses are reduced modulo I and have norm at most $m(R_*)$.

  Note that $\vec{r}$, $\vec{y}$, $\vec{r}_A$, $\vec{r}_B$, $\vec{r}_C$, $\vec{s}'$, and σ are known to $\mathbf{V}$, who will compute commitments to these values for themself, with respect to bases $\vec{\mathsf{G}}$, $\vec{\mathsf{H}}$ or U as required. This is signified in the table using a "#" symbol.

  Also, note that challenge vectors such as $\vec{r}$ have elements in R rather than M_L, but are easy to map into M_L simply by multiplying each element by 1_{M_L}. We do not do this explicitly in order to keep notation simple.

Witness	Instance commitments X, Y, Z	Checks	Purpose
$(\vec{z}_B, \vec{z}'_A, \vec{\rho}_B, \vec{\rho}'_A, \vec{\rho}_\gamma)$	$\mathsf{C}_{\vec{z}_B}, \mathsf{C}'_{\vec{z}_A}, \mathsf{C}_\gamma$	$\langle \vec{z}_B, \vec{z}'_A \rangle = \gamma \bmod I$	Hadamard check
$(\vec{y}, \vec{z}'_A, 0, \vec{\rho}'_A, \vec{\rho}''_\alpha)$	$\#, \mathsf{C}'_{\vec{z}_A}, \mathsf{C}''_\alpha$	$\langle \vec{y}, \vec{z}'_A \rangle = \alpha'' \bmod I$	Hadamard check
$(\vec{z}_A, \vec{r} \circ \vec{y}, \vec{\rho}_A, 0, \vec{\rho}''_\alpha)$	$\mathsf{C}_{\vec{z}_A}, \#, \mathsf{C}''_\alpha$	$\langle \vec{z}_A, \vec{r} \circ \vec{y} \rangle = \alpha'' \bmod I$	Hadamard check
$(\vec{z}, \vec{r}_A, \vec{\rho}, 0, \vec{\rho}_\alpha)$	$\mathsf{C}_{\vec{z}}, \#, \mathsf{C}_\alpha$	$\langle \vec{z}, \vec{r}_A \rangle = \alpha \bmod I$	Lincheck for A
$(\vec{z}_A, \vec{r}, \vec{\rho}_A, 0, \vec{\rho}_\alpha)$	$\mathsf{C}_{\vec{z}_A}, \#, \mathsf{C}_\alpha$	$\langle \vec{z}_A, \vec{r} \rangle = \alpha \bmod I$	Lincheck for A
$(\vec{z}, \vec{r}_B, \vec{\rho}, 0, \vec{\rho}_\beta)$	$\mathsf{C}_{\vec{z}}, \#, \mathsf{C}_\beta$	$\langle \vec{z}, \vec{r}_B \rangle = \beta \bmod I$	Lincheck for B
$(\vec{z}_B, \vec{r}, \vec{\rho}_B, 0, \vec{\rho}_\beta)$	$\mathsf{C}_{\vec{z}_B}, \#, \mathsf{C}_\beta$	$\langle \vec{z}_B, \vec{r} \rangle = \beta \bmod I$	Lincheck for B
$(\vec{z}, \vec{r}_C, \vec{\rho}, 0, \vec{\rho}_\gamma)$	$\mathsf{C}_{\vec{z}}, \#, \mathsf{C}_\gamma$	$\langle \vec{z}, \vec{r}_C \rangle = \gamma \bmod I$	Lincheck for C
$(\vec{z}, \vec{s}', \vec{\rho}, 0, 0)$	$\mathsf{C}_{\vec{z}}, \#, \#$	$\langle \vec{z}, \vec{s}' \rangle = \sigma \bmod I$	Partial assignment

Lemma 7.4 (completeness)**.** *Construction 7.3 is complete with completeness error $O(n/\kappa)$, where κ is the parameter used for masking in Section 6.*

Proof. Recall that the scalar-product sub-protocol of Construction 6.5 is complete (see Lemma 6.8). Let $(\mathbb{x}, \mathbb{w}) = \big((R_*, A, B, C, M, n_{\text{row}}, n_{\text{col}}, n_{\text{in}}, x), \vec{w}\big) \in \mathcal{R}_{\text{R1CS}}$. Then $\vec{z} := (x, \vec{w})$ satisfies $A\vec{z} \circ B\vec{z} = C\vec{z} \bmod I$. The prover $\mathbf{P}$ computes $\vec{z}_A = A\vec{z} \bmod I$ and similarly for B and C. Therefore $\vec{z}_A \circ \vec{z}_B = \vec{z}_C \bmod I$.

The prover also computes $\vec{z}'_A = \vec{r} \circ \vec{z}_A \bmod I$ and $\gamma = \langle \vec{r}, \vec{z}_C \rangle \bmod I$. For any choice of $\vec{r}$, we have

$$\begin{aligned} \langle \vec{z}_B, \vec{z}'_A \rangle &= \langle \vec{r}, \vec{z}_A \circ \vec{z}_B \rangle \\ &= \langle \vec{r}, \vec{z}_C \rangle = \gamma \bmod I \ . \end{aligned}$$

Thus, $\vec{z}_B$, $\vec{z}'_A$ and γ and their commitments give a member of the scalar-product relation $\mathcal{R}_{\text{comSP}}$ for suitable B_{R1CS}, so the first scalar-product protocol succeeds, except with probability $O(n/\kappa)$. Further, for every choice of $\vec{y}$, we have $\langle \vec{y}, \vec{z}'_A \rangle = \langle \vec{y}, \vec{r} \circ \vec{z}_A \rangle = \alpha'' \bmod I$. Therefore, the second and third scalar-product protocols succeed, except with probability $O(n/\kappa)$.

For every choice of $\vec{r}$ it holds that $\langle \vec{r}, \vec{z}_A \rangle = \langle \vec{r}, A\vec{z} \rangle = \langle \vec{r}^{\intercal} A, \vec{z} \rangle = \langle \vec{r}_A, \vec{z} \rangle \bmod I$, and therefore, the fourth and fifth scalar-product protocols succeed (the Lincheck for A), except with probability $O(n/\kappa)$. Similar reasoning applies to the Linchecks for B and C.

Finally we discuss the consistency check between the full assignment $\vec{z} = (x, \vec{w})$ and the partial assignment x. For every choice of $\vec{s}$, it holds that $\langle \vec{s}', \vec{z} \rangle = \langle \vec{s}, x \rangle = \sigma \bmod I$, which is the equation checked by the verifier. This means that $\vec{z}$, $\vec{s}'$, σ and their commitments are in the relation $\mathcal{R}_{\text{comSP}}$, so the final scalar product sub-protocol succeeds. □

Lemma 7.5 (knowledge soundness)**.** *There exists a PPT algorithm such that given an instance* $\mathbb{x} = (R_*, A, B, C, M, n_{\text{row}}, n_{\text{col}}, n_{\text{in}}, x)$*, and a* $(2^\ell, 2^\ell, 3, 4^\ell)$*-tree of accepting transcripts, either extracts a valid witness* $\mathbb{w} = \vec{w}$ *for the relation* $\mathcal{R}_{\text{R1CS}}$ *or a non-trivial bilinear relation, whenever* $\gamma_R n B_{\mathsf{SP}}^2 < B_{\max}$*. Here* B_{SP} *is the norm bound derived in Lemma 6.9.*

Proof. Fix values of $\vec{r}$, $\vec{y}$ and $\vec{s}$ (this corresponds to following some path from the root down to the $(2\ell + \log n_{\text{in}})$-th level of the tree of transcripts). By Lemma 6.9 there exists a probabilistic polynomial time algorithm which takes as input the $(3, 2, 2, 4^\ell)$-subtree of accepting transcripts (the full tree restricted to each scalar-product subprotocol) and outputs a witness to $\mathcal{R}_{\text{comSP}}(c, B_{\mathsf{SP}})$ (see Definition 6.2) for some c which is invertible modulo I by the properties of $\mathcal{C}$ and I, and $B' = 6D_3\gamma_R^2 m(\mathcal{C}) m(R_*)$.

Consider the first scalar-product subprotocol, run on commitments $\mathsf{C}_{\vec{z}_B}$, $\mathsf{C}'_{\vec{z}_A}$ and C_γ. We can extract vectors $\vec{z}_B, \vec{z}'_A \in M_L(B)^{n_{\text{col}}}$ and a scalar γ such that $c \cdot \mathsf{C}_{\vec{z}_B} = \langle \vec{z}_B, \vec{\mathsf{G}} \rangle + \langle \vec{\rho}_B, \vec{\mathsf{G}}_0 \rangle$, $c \cdot \mathsf{C}'_{\vec{z}_A} = \langle \vec{z}'_A, \vec{\mathsf{H}} \rangle + \langle \vec{\rho}'_A, \vec{\mathsf{H}}_0 \rangle$, and $c^2 \cdot \mathsf{C}_\gamma = \gamma \cdot \mathsf{U} + \langle \vec{\rho}_\gamma, \vec{\mathsf{U}}_0 \rangle$, with $\langle \vec{z}_B, \vec{z}'_A \rangle = \gamma \bmod I$.

For the second scalar-product protocol, run on honestly-made commitments to $\vec{y}$ and α'', and commitment $\mathsf{C}'_{\vec{z}_A}$, by applying the knowledge extractor and the binding property of the generalised Pedersen commitment scheme, we can deduce that $\langle c \cdot \vec{y}, \vec{z}'_A \rangle = c^2 \cdot \alpha'' \bmod I$, since the norm bounds on $c \cdot \vec{y}$ and $\vec{z}'_A$ are both less than $B_{\max}$.

Similarly, by applying the knowledge extractor for each scalar-product protocol and using the binding property to show that relaxed openings for commitments made by the verifier are multiples of the honestly committed values, we can extract $\vec{z}$, $\vec{z}_A$ and $\vec{z}_B$ with norms at most B_{R1CS}, and α, α'', β, and γ with norms at most $\gamma_R n B_{\mathsf{R1CS}}^2$ satisfying the following equations:

$$\begin{aligned} \langle \vec{z}_A, c \cdot \vec{r} \circ \vec{y} \rangle &= c^2 \cdot \alpha'' \bmod I \\ \langle \vec{z}, c \cdot \vec{r}_A \rangle &= c^2 \cdot \alpha \bmod I \\ \langle \vec{z}_A, c \cdot \vec{r} \rangle &= c^2 \cdot \alpha \bmod I \\ \langle \vec{z}, c \cdot \vec{r}_B \rangle &= c^2 \cdot \beta \bmod I \end{aligned}$$

$$\langle \vec{z}_B, c \cdot \vec{r} \rangle = c^2 \cdot \beta \bmod I$$
$$\langle \vec{z}, c \cdot \vec{r}_C \rangle = c^2 \cdot \gamma \bmod I$$
$$\langle \vec{z}_C, c \cdot \vec{r} \rangle = c^2 \cdot \gamma \bmod I$$
$$\langle \vec{z}, c \cdot \vec{s}' \rangle = c^2 \cdot \sigma \bmod I$$

We also know that $\langle c \cdot \vec{s}, c \cdot x \rangle = c^2 \cdot \sigma \bmod I$.

Equating the left-hand sides of the equations above which have related right-hand sides, and dividing by c^2 (c has an inverse because the set $\mathcal{C}$ is a strong sampling set for M_L/I), we can write

$$\langle \vec{y}, c^{-1} \cdot \vec{z}'_A \rangle = \langle c^{-1} \cdot \vec{z}_A, \vec{r} \circ \vec{y} \rangle = \langle c^{-1} \cdot \vec{z}_A \circ \vec{r}, \vec{y} \rangle \bmod I \ , \tag{23}$$
$$\langle c^{-1} \cdot \vec{z}_B, c^{-1} \cdot \vec{z}'_A \rangle = \langle c^{-1} \cdot \vec{z}, \vec{r}_C \rangle \bmod I \ , \tag{24}$$
$$\langle c^{-1} \cdot \vec{z}, \vec{r}_A \rangle = \langle c^{-1} \cdot \vec{z}_A, \vec{r} \rangle \bmod I \ , \tag{25}$$
$$\langle c^{-1} \cdot \vec{z}, \vec{r}_B \rangle = \langle c^{-1} \cdot \vec{z}_B, \vec{r} \rangle \bmod I \ , \tag{26}$$
$$\langle c^{-1} \cdot \vec{z}, \vec{s}' \rangle = \langle \vec{s}, x \rangle \bmod I \ . \tag{27}$$

Now, we show that $c^{-1} \cdot \vec{z}$ is a witness to the R1CS instance modulo I.

Consider Equation (23) for each accepting transcript at the $(2\ell + \log n_{\text{in}})$-th level. Since Equation (23) holds for a $2^{2 \log n_{\text{row}}}$-tree of values of $(y_1, \ldots, y_\ell), (r_1, \ldots, r_\ell)$ and the entries of $\vec{s}$, we can solve for the coefficients of y_i in $\langle \vec{y}, c^{-1} \cdot \vec{z}'_A \rangle$ and deduce that $c^{-1} \cdot \vec{z}'_A = c^{-1} \cdot \vec{r} \circ \vec{z}_A \bmod I$. This is always possible because e.g. $\langle \vec{y}, c^{-1} \cdot \vec{z}'_A \rangle$ is a multilinear polynomial in $(\vec{y}_1, \ldots, \vec{y}_{\log n_{\text{row}}})$, and solving for the coefficients amounts to solving linear equations in $\vec{y}_{\log n_{\text{row}}}$, then $\vec{y}_{\log n_{\text{row}}-1}$, and so on, recursively. Since $\mathcal{C}$ remains a strong sampling set modulo I, each linear equation is solvable modulo I.

Substituting $c^{-1} \cdot \vec{z}'_A = c^{-1} \cdot \vec{r} \circ \vec{z}_A$ into Equation (24) and applying the same technique shows that $c^{-1}\vec{z}_A \circ c^{-1}\vec{z}_B = C \cdot c^{-1}\vec{z} \bmod I$. Applying the same technique to Equation (25), Equation (26) and Equation (27) implies that $c^{-1}\vec{z}_A = A \cdot c^{-1}\vec{z} \bmod I$, $c^{-1}\vec{z}_B = B \cdot c^{-1}\vec{z} \bmod I$, and $c^{-1}\vec{z} = (x, \vec{w})$ for some vector $\vec{w}$, and thus $c^{-1}\vec{z}$ is a witness to the R1CS relation modulo I. □

Lemma 7.6 (zero-knowledge)**.** *Construction 7.3 has semi-honest verifier zero-knowledge.*

Proof. The simulator runs as follows:

- The simulator computes random commitments $\mathsf{C}_{\vec{z}_A}$, $\mathsf{C}'_{\vec{z}_A}$, $\mathsf{C}_{\vec{z}_B}$, $\mathsf{C}_{\vec{z}}$, C_α, C''_α, C_β, C_γ to zero messages, using randomness sampled uniformly from $\mathbb{R}(m(R_*))$.
- The simulator invokes the simulator of the scalar-product protocol for each subprotocol (given by Lemma 6.10).

Zero-knowledge follows from the hiding property of the generalised Pedersen commitment scheme, and the zero-knowledge property of the scalar-product protocol. □

Lemma 7.7. *The prover and verifier in Construction 7.3 performs the following operations: $O(n_{\text{row}})$ bilinear operations between M_L and M_R; $O(n_{\text{row}} + M)$ operations in M_L; and $O(n_{\text{row}})$ additions in M_T.*

Proof. The costs of Construction 7.3 are inherited from the costs of Construction 6.5, analysed in Lemma 6.6 and Lemma 6.7. □

8 Polynomial commitments over rings

We show how to use the scalar-product protocol over rings to get a polynomial commitment over rings. In particular, the polynomial commitment scheme enables a sender to commit to a polynomial p over a ring R and then later prove the correct evaluation of the polynomial at a desired point $\bmod I$ for an ideal $I \subseteq R$ as in Section 2.2.1. We first provide the definition of polynomial commitments, and then show the construction through the scalar-product sumcheck argument.

8.1 Definition

A polynomial commitment scheme consists of a binding commitment scheme for a message space $R[X]$ of polynomials over some ring R, and an interactive public-coin protocol between a PPT prover $\mathbf{P}$ and verifier $\mathbf{V}$. In the interactive protocol, both $\mathbf{P}$ and $\mathbf{V}$ have as input a commitment C, points $v, z \in R$, and a degree d. The prover additionally knows the opening of C to a secret polynomial $p(X) \in R[X]$ with $\deg(p(X)) \leq d$. The protocol convinces the verifier that $p(z) = v \bmod I$ for some ideal $I \subseteq R$. In a multivariate extension of polynomial commitments, the input $m > 1$ indicates the number of variables in the committed polynomial and $z \in R^m$.

Definition 8.1. *A polynomial commitment scheme is a tuple of algorithms* $(\mathbf{G}, \mathbf{Trim}, \mathbf{Commit})$ *and an interactive protocol* $z = (\mathbf{P}, \mathbf{V})$ *where*

- $\mathbf{G}(1^\lambda, D, m, \{uni, multi\}, \{Lagrange, monomials\}) \to \mathsf{pp}_{R,I}$*. On input a security parameter* λ *(in unary), a maximum degree bound* $D \in \mathbb{N}$*, the number of variables* m*, whether we consider univariate or multivariate polynomials and whether the polynomials are given in Lagrange or monomial basis,* $\mathbf{G}$ *samples public parameters* $\mathsf{pp}_{R,I}$*. The public parameters contain the ring* R*, and ideal* I *such that* $I \subseteq R$*. The committed polynomial belongs to* $R[\vec{X}]$*, but the evaluation is done* $\bmod I$*.*
- $\mathbf{Trim}(\mathsf{pp}_{R,I}, d) \to (\mathrm{ck}, \mathrm{rk})$*. On input public parameters* $\mathsf{pp}_{R,I}$*, and degree bound* $d \leq D$*,* $\mathbf{Trim}$ *deterministically computes a key pair* $(\mathrm{ck}, \mathrm{rk})$ *that is specialized to* d*.*
- $\mathbf{Commit}(\mathrm{ck}, p; r) \to \mathsf{C}$*. On input* ck*, a polynomial* p *over the ring* R*,* $\mathbf{Commit}$ *outputs a commitment* C *to the polynomial* p*. The randomness* r *is used if the commitment* C *is hiding.*
- $\mathbf{Eval} = (\mathbf{P}(\mathrm{ck}, p, \mathsf{C}, z, v; r_{\mathbf{P}}), \mathbf{V}(\mathrm{rk}, \mathsf{C}, z, v; r_{\mathbf{V}}))$*. The prover, with input* ck*, a polynomial* p*, a commitment* C*, an evaluation point* z*, a target point* v*, and randomness* $r_{\mathbf{P}}$*, and the verifier with input* rk*, a commitment* C*, an evaluation point* z*, a target point* v*, and randomness* $r_{\mathbf{V}}$*, engage in an interactive* public coin *protocol. At the end of the protocol,* $\mathbf{V}$ *outputs* 1 *if* $\mathbf{P}$ *attests that the polynomial* p *committed in* C *has degree at most* d *and evaluates* $\bmod I$ *to* v *at* z*.*

A polynomial commitment scheme must satisfy the following properties.

Completeness. For every maximum degree $D = \mathsf{poly}(\lambda) \in \mathbb{N}$ and every adversary $\mathcal{A}$

$$\Pr\left[\mathbf{V}(\mathrm{rk}, \mathsf{C}, z, v) = 1 \;\middle|\; \begin{array}{l} \mathsf{pp}_{R,I} \leftarrow \mathbf{G}(1^\lambda, D, m, \{uni, multi\}, \{Lagrange, monomials\}) \\ (p, d, z, r) \leftarrow \mathcal{A}(\mathsf{pp}_{R,I}) \\ (\mathrm{ck}, \mathrm{rk}) \leftarrow \mathbf{Trim}(\mathsf{pp}_{R,I}, d) \\ \mathsf{C} \leftarrow \mathbf{Commit}(\mathrm{ck}, p; r) \\ v = p(z) \bmod I \\ \mathbf{Eval} = (\mathbf{P}(\mathrm{ck}, p, \mathsf{C}, z, v; r), \mathbf{V}(\mathrm{rk}, \mathsf{C}, z, v)) \end{array}\right] = 1.$$

Extractability. There exists an efficient extractor χ and a function $\mathbf{t}$ such that for every maximum degree bound $D = \mathsf{poly}(\lambda) \in \mathbb{N}$ and polynomial-size adversary $\mathcal{A}$ the following holds.

$$\Pr\left[\begin{array}{l|l} T \text{ is an accepting } \mathbf{t}(d)\text{-tree for } \mathbf{Eval} & \mathsf{pp}_{R,I} \leftarrow \mathbf{G}(1^\lambda, D, m, \{uni, multi\}, \{Lagrange, monomials\}) \\ \Downarrow & (\mathsf{C}, d, z, v, T) \leftarrow \mathcal{A}(\mathsf{pp}_{R,I}) \\ p(z) = v \bmod I & (p, r) \leftarrow \chi(\mathsf{pp}_{R,I}, \mathsf{C}, d, z, v, T) \\ \mathsf{C} \leftarrow \mathbf{Commit}(\mathrm{ck}, p; r) & (\mathrm{ck}, \mathrm{rk}) \leftarrow \mathbf{Trim}(\mathsf{pp}_{R,I}, d) \end{array}\right] = 1.$$

Hiding. There exists a stateful polynomial-time simulator Sim such that, for every maximum degree bound $D = \mathsf{poly}(\lambda) \in \mathbb{N}$ and stateful polynomial-size adversary $\mathcal{A}$, the output distributions of the following games are statistically close.

$\mathrm{Real}(1^\lambda, D, \mathcal{A})$

1. $\mathsf{pp}_{R,I} \leftarrow \mathbf{G}(1^\lambda, D, m, \{uni, multi\}, \{Lagrange, monomials\})$
2. $(p, d) \leftarrow \mathcal{A}(\mathsf{pp}_{R,I})$
3. $(\mathrm{ck}, \mathrm{rk}) \leftarrow \mathbf{Trim}(\mathsf{pp}_{R,I}, d)$
4. Sample commitment randomness r
5. $\mathsf{C} \leftarrow \mathbf{Commit}(\mathrm{ck}, p; r)$
6. $z \leftarrow \mathcal{A}(\mathsf{C})$
7. Sample opening randomness $r_{\mathbf{P}}$ and $r_{\mathbf{V}}$
8. $\mathrm{tr} = \langle \mathbf{P}(\mathrm{ck}, p, \mathsf{C}, z, p(z); r_{\mathbf{P}}), \mathbf{V}(\mathrm{rk}, \mathsf{C}, z, p(z); r_{\mathbf{V}}) \rangle$
9. Output $(\mathsf{pp}_{R,I}, p, d, \mathsf{C}, z, \mathrm{tr})$

$\mathrm{Ideal}(1^\lambda, D, \mathcal{A})$

1. $(\mathsf{pp}_{R,I}, trap) \leftarrow \mathrm{Sim}.\mathbf{G}(1^\lambda, D, m, \{uni, multi\}, \{Lagrange, monomials\})$
2. $(p, d) \leftarrow \mathcal{A}(\mathsf{pp}_{R,I})$
3. $\mathsf{C} \leftarrow \mathrm{Sim}.\mathbf{Commit}(trap, d)$
4. $z \leftarrow \mathcal{A}(\mathsf{C})$
5. $\mathrm{tr} \leftarrow \mathrm{Sim}.\mathbf{Open}(\mathsf{C}, z, p(z))$
6. Output $(\mathsf{pp}_{R,I}, p, d, \mathsf{C}, z, \mathrm{tr})$

8.2 Polynomial commitment from scalar-product protocol

We construct a polynomial commitment for univariate and multilinear multivariate polynomials based on the scalar product protocol of Section 6. The polynomials can be given in either Lagrange or monomial basis. First, we show that the polynomial evaluation can be computed as a scalar product.

- A univariate polynomial in monomial basis of degree d, $p(X) = \sum_{i=0}^{d} a_i X^i$, is represented with the vector $\vec{a} = (a_0, \ldots, a_d)$. If $\vec{b} = (1, z, z^2, \ldots, z^d)$, then $\langle \vec{a}, \vec{b} \rangle = p(z)$.
- A multivariate multilinear polynomial in monomial basis is represented with a vector $\vec{a} = (a_0, \ldots, a_{2^m-1})$ where a_i with $i = (i_1, \ldots, i_m)$ is the coefficient of the monomial $\prod_{j=1}^{m} X_j^{i_j}$. If $\vec{b}$ is such that $\vec{b}_i = \prod_{j=1}^{m} z_j^{i_j}$ with $i = (i_1, \ldots, i_m)$, then $\langle \vec{a}, \vec{b} \rangle = p(z_1, \ldots, z_m)$.
- A univariate polynomial in Lagrange basis of degree d, $p(X) = \sum_{\omega \in \mathcal{H}} p(\omega)\ell_\omega(X)$ with $|\mathcal{H}| = d+1$, where $\ell_\omega(X) = \prod_{\omega' \neq \omega, \omega' \in \mathcal{H}} \frac{X-\omega'}{\omega-\omega'}$, is represented with the vector $\vec{a} = (p(\omega_1), \ldots, p(\omega_{d+1}))$ where $\mathcal{H} = \{\omega_i\}_{i \in [d+1]}$. If $\vec{b} = (\ell_{\omega_1}(z), \ldots, \ell_{\omega_{d+1}}(z))$, then $\langle \vec{a}, \vec{b} \rangle = p(z)$.
- A multivariate multilinear polynomial in Lagrange basis is written as $p(\vec{X}) = \sum_{\vec{\omega} \in \mathcal{H}} p(\vec{\omega})\ell_{\mathcal{H},\vec{\omega}}(X_1, \ldots, X_m)$ with $|\mathcal{H}| = 2^m + 1$ and $\mathcal{H} = \mathcal{H}^{(1)} \times \cdots \times \mathcal{H}^m$ where $\ell_{\mathcal{H},\vec{\omega}}(X_1, \ldots, X_m) = \prod_{i=1}^{m} \ell_{\mathcal{H}^{(i)},\omega_i}(X_i)$ and $\ell_{\mathcal{H}^{(i)},\omega_i}(X_i) = \prod_{\omega \in \mathcal{H}^{(i)}, \omega \neq \omega_i} \frac{X_i - \omega}{\omega_i - \omega}$. The polynomial p is represented with the vector $\vec{a} = (p(\vec{\omega}_1), \ldots, p(\vec{\omega}_{2^m}))$ where $\mathcal{H} = \{\vec{\omega}_i\}_{i \in [2^m]}$. If $\vec{b} = (\ell_{\mathcal{H},\vec{\omega}_1}(z_1, \ldots, z_m), \ldots, \ell_{\mathcal{H},\vec{\omega}_{2^m}}(z_1, \ldots, z_m))$, then $\langle \vec{a}, \vec{b} \rangle = p(z_1, \ldots, z_m)$.

We focus on the construction of polynomial commitment for univariate polynomials in monomial basis and the other cases follow similarly. The underlying binding commitment is the Pedersen commitment over an argument-friendly bilinear module. Then, the interactive protocol $\mathbf{Eval}$ is basically the scalar-product protocol of Section 6, where the part of the witness corresponding to the vector $\vec{b}$ is public and $\vec{\rho}_1 = 0$.

Construction 8.2. A polynomial commitment scheme for univariate polynomials in monomial basis is a tuple of algorithms $(\mathbf{G}, \mathbf{Trim}, \mathbf{Commit})$ and an interactive protocol $\mathbf{Eval} = (\mathbf{P}, \mathbf{V})$ as follows.

- $\mathbf{G}(1^\lambda, D, 1, uni, monomials)$ outputs $\mathsf{pp}_{R,I} = (\mathcal{M}, I, \vec{\mathsf{G}}', \vec{\mathsf{G}}_0, \vec{\mathsf{H}}', \vec{\mathsf{H}}_0, \mathsf{U}, \vec{\mathsf{U}}_0)$, where the elements of $\mathsf{pp}_{R,I}$ are sampled as in Construction 6.5.
- $\mathbf{Trim}(\mathsf{pp}_{R,I}, d)$ outputs $(\mathrm{ck}, \mathrm{rk})$ such that $\mathrm{ck} = \mathrm{rk} = (\vec{\mathsf{G}}, \vec{\mathsf{G}}_0, \vec{\mathsf{H}}, \vec{\mathsf{H}}_0, \mathsf{U}, \vec{\mathsf{U}}_0)$ where $\vec{\mathsf{G}}$ and $\vec{\mathsf{H}}$ are the first d elements of $\vec{\mathsf{G}}'$ and $\vec{\mathsf{H}}'$ respectively.
- $\mathbf{Commit}(\mathrm{ck}, p; \vec{\rho}_0)$ outputs a hiding Pedersen commitment for the polynomial p, represented as a vector $\vec{a} \in M_L$. In particular, the hiding commitment of p is of the form $\langle \vec{a}, \vec{\mathsf{G}} \rangle + \vec{\rho}_0 \cdot \vec{\mathsf{G}}_0 \in M_T$. We ignore the randomness $\vec{\rho}_0$ if we do not require a hiding polynomial commitment.
- $\mathbf{Eval} = (\mathbf{P}(\mathrm{ck}, p, \mathsf{C}_0, z, v; r_{\mathbf{P}}), \mathbf{V}(\mathrm{rk}, \mathsf{C}_0, z, v; r_{\mathbf{V}}))$ follows the protocol for committed scalar product of Section 6. Let $\vec{b} = (1, z, z^2, \ldots, z^d)$ and $\mathbb{x} = (\mathcal{M}, r_{\mathsf{Ped}}, I, \vec{\mathsf{G}}, \vec{\mathsf{G}}_0, \vec{\mathsf{H}}, \vec{\mathsf{H}}_0, \mathsf{U}, \vec{\mathsf{U}}_0, \mathsf{C}_0, \mathsf{C}_1, \mathsf{C}_t)$ with $\mathsf{C}_1 = \langle \vec{b}, \vec{\mathsf{H}} \rangle$, and $\mathsf{C}_t = \langle \vec{a}, \vec{b} \rangle \cdot \mathsf{U} + \vec{\rho}_t \cdot \vec{\mathsf{U}}_0$, the prover runs $\mathbf{P}_{\mathrm{comSP}}(\mathbb{x}, (\vec{a}, \vec{b}, \vec{\rho}_0, 0, \vec{\rho}_t))$, and the verifier runs $\mathbf{V}_{\mathrm{comSP}}(\mathbb{x})$.

Theorem 8.3. *Construction 8.2 is a polynomial commitment for univariate polynomials in monomial basis.*

sketch. We show that Construction 8.2 satisfies the completeness, extractability and hiding properties by directly applying the corresponding lemmas of Section 6. Completeness follows directly from Lemma 6.8.

For the extractability property, we run the extractor of Lemma 6.9 for $\mathbb{x} = (\mathcal{M}, r_{\mathsf{Ped}}, I, \vec{\mathsf{G}}, \vec{\mathsf{G}}_0, \vec{\mathsf{H}}, \vec{\mathsf{H}}_0, \mathsf{U}, \vec{\mathsf{U}}_0, \mathsf{C}_0, \mathsf{C}_1, \mathsf{C}_t)$ to recover $\vec{a}'$ and $\vec{b}'$ such that $c_{\mathsf{SP}}\mathsf{C}_0 = \langle \vec{a}', \vec{\mathsf{G}} \rangle + \langle \vec{\rho}_0, \vec{\mathsf{G}}_0 \rangle$, $c_{\mathsf{SP}}\mathsf{C}_1 = \langle \vec{b}', \vec{\mathsf{H}} \rangle + \langle \vec{\rho}_1, \vec{\mathsf{H}}_0 \rangle$, and $c_{\mathsf{SP}}^2\mathsf{C}_t = (\langle \vec{a}', \vec{b}' \rangle \bmod I) \cdot \mathsf{U} + \vec{\rho}_t \cdot \vec{\mathsf{U}}_0$. Because of the binding property, it must be that $\vec{b}' = c_{\mathsf{SP}}\vec{b}$ and $\vec{\rho}_1 = 0$. Finally, the polynomial p is equal to $c_{\mathsf{SP}}^{-1}\vec{a}' \bmod I$, which is well-defined since $c_{\mathsf{SP}} \bmod I$ is invertible in R_*.

For the hiding property, we use the simulator of Lemma 6.10 without the hiding factors for $\vec{b}$.

□

A Review of the sumcheck protocol

We state the sumcheck protocol [LFKN92] and describe the soundness guarantee that it offers.

Protocol 6: sumcheck protocol (Π_{SC})

The prover and verifier receive a polynomial $p \in \mathbb{F}[X_1, \ldots, X_\ell]$, subset $H \subseteq \mathbb{F}$, and claimed sum $\tau \in \mathbb{F}$. The prover wishes to convince the verifier that $\sum_{\omega_1,\ldots,\omega_\ell \in H} p(\omega_1, \ldots, \omega_\ell) = \tau$, and the verifier will only have to query p at a single (random) point.

If $\ell = 0$, then for $p \in \mathbb{F}$, the verifier checks that $p = \tau$. If $\ell > 0$ then the interactive reduction works as follows.

1. The prover $\mathbf{P}$ sends the polynomial $q(X) \in \mathbb{F}[X]$ to the verifier, computed as follows:
$$q(X) := \sum_{\omega_2,\ldots,\omega_\ell \in H} p(X, \omega_2, \ldots, \omega_\ell) \ .$$
2. The verifier $\mathbf{V}$ samples $r \leftarrow \mathbb{F}$ and sends r to the prover.
3. The verifier checks that $\tau = \sum_{\omega \in H} q(\omega)$. (If not, it rejects.)
4. The verifier outputs the new claimed sum $\tau' := q(r) \in \mathbb{F}$ for the new polynomial $p'(X_2, \ldots, X_\ell) := p(r, X_2, \ldots, X_\ell) \in \mathbb{F}[X_2, \ldots, X_\ell]$.

Lemma A.1. *For any polynomial $p \in \mathbb{F}[X_1, \ldots, X_\ell]$ of individual degree at most d, any $H \subseteq \mathbb{F}$ and any $\tau \in \mathbb{F}$, if $\sum_{\omega_1,\ldots,\omega_\ell \in H} p(\omega_1, \ldots, \omega_\ell) \neq \tau$, then the verifier of Protocol 6 accepts with probability at most $d\ell/|\mathbb{F}|$.*

Acknowledgments

This research was supported in part by a donation from the Ethereum Foundation. Part of the work was conducted while the first author was employed by UC Berkeley.

References

[AC20] Thomas Attema and Ronald Cramer. "Compressed Σ-Protocol Theory and Practical Application to Plug & Play Secure Algorithmics". In: *Proceedings of the 40th Annual International Cryptology Conference*. CRYPTO '20. 2020, pp. 513–543.

[ACDEY19] Mark Abspoel, Ronald Cramer, Ivan Damgård, Daniel Escudero, and Chen Yuan. "Efficient Information-Theoretic Secure Multiparty Computation over $\mathbb{Z}/p^k\mathbb{Z}$ via Galois Rings". In: *Proceedings of the 17th Theory of Cryptography Conference*. TCC '19. 2019, pp. 471–501.

[ACF20] Thomas Attema, Ronald Cramer, and Serge Fehr. *Compressing Proofs of k-Out-Of-n Partial Knowledge*. IACR Cryptology ePrint Archive, Report 2020/753. 2020.

[ACR20] Thomas Attema, Ronald Cramer, and Matthieu Rambaud. *Compressed Sigma-Protocols for Bilinear Circuits and Applications to Logarithmic-Sized Transparent Threshold Signature Schemes*. IACR Cryptology ePrint Archive, Report 2020/1447. 2020.

[AFGHO16] Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. "Structure-Preserving Signatures and Commitments to Group Elements". In: *Journal of Cryptology* 29 (2 2016), pp. 363–421.

[Abs+20] Mark Abspoel, Ronald Cramer, Ivan Damgård, Daniel Escudero, Matthieu Rambaud, Chaoping Xing, and Chen Yuan. "Asymptotically Good Multiplicative LSSS over Galois Rings and Applications to MPC over $\mathbb{Z}/p^k\mathbb{Z}$". In: *Proceedings of the 26th International Conference on the Theory and Application of Cryptology and Information Security*. ASIACRYPT '20. 2020, pp. 151–180.

[Adj] URL: `https://github.com/adjoint-io/bulletproofs`.

[BBBPWM18] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. "Bulletproofs: Short Proofs for Confidential Transactions and More". In: *Proceedings of the 39th IEEE Symposium on Security and Privacy*. S&P '18. 2018, pp. 315–334.

[BBCPGL18] Carsten Baum, Jonathan Bootle, Andrea Cerulli, Rafaël del Pino, Jens Groth, and Vadim Lyubashevsky. "Sub-linear Lattice-Based Zero-Knowledge Arguments for Arithmetic Circuits". In: *Proceedings of the 38th Annual International Cryptology Conference*. CRYPTO '18. 2018, pp. 669–699.

[BCCGP16] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. "Efficient Zero-Knowledge Arguments for Arithmetic Circuits in the Discrete Log Setting". In: *Proceedings of the 35th Annual International Conference on Theory and Application of Cryptographic Techniques*. EUROCRYPT '16. 2016, pp. 327–357.

[BCG20] Jonathan Bootle, Alessandro Chiesa, and Jens Groth. "Linear-Time Arguments with Sublinear Verification from Tensor Codes". In: *Proceedings of the 18th Theory of Cryptography Conference*. TCC '20. 2020, pp. 19–46.

[BCGGRS19] Eli Ben-Sasson, Alessandro Chiesa, Lior Goldberg, Tom Gur, Michael Riabzev, and Nicholas Spooner. "Linear-Size Constant-Query IOPs for Delegating Computation". In: *Proceedings of the 17th Theory of Cryptography Conference*. TCC '19. 2019, pp. 494–521.

[BCKLN14] Fabrice Benhamouda, Jan Camenisch, Stephan Krenn, Vadim Lyubashevsky, and Gregory Neven. "Better Zero-Knowledge Proofs for Lattice Encryption and Their Application to Group Signatures". In: *Proceedings of the 20th International Conference on the Theory and Application of Cryptology and Information Security*. ASIACRYPT '14. 2014, pp. 551–572.

[BCL20] Jonathan Bootle, Alessandro Chiesa, and Siqi Liu. *Zero-Knowledge Succinct Arguments with a Linear-Time Prover*. IACR Cryptology ePrint Archive, Report 2020/1527. 2020.

[BCOS20] Cecilia Boschini, Jan Camenisch, Max Ovsiankin, and Nicholas Spooner. "Efficient Post-quantum SNARKs for RSIS and RLWE and Their Applications to Privacy". In: *Proceedings of the 11th International Conference on Post-Quantum Cryptography*. PQCrypto '20. 2020, pp. 247–267.

[BCRSVW19] Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. "Aurora: Transparent Succinct Arguments for R1CS". In: *Proceedings of the 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. EUROCRYPT '19. Full version available at `https://eprint.iacr.org/2018/828`. 2019, pp. 103–128.

[BDFG20] Dan Boneh, Justin Drake, Ben Fisch, and Ariel Gabizon. *Halo Infinite: Recursive zk-SNARKs from any Additive Polynomial Commitment Scheme*. IACR Cryptology ePrint Archive, Report 2020/1536. 2020.

[BFHVXZ20] Rishabh Bhadauria, Zhiyong Fang, Carmit Hazay, Muthuramakrishnan Venkitasubramaniam, Tiancheng Xie, and Yupeng Zhang. "Ligero++: A New Optimized Sublinear IOP". In: *Proceedings of the 27th ACM Conference on Computer and Communications Security*. CCS '20. 2020, pp. 2025–2038.

[BFL91] László Babai, Lance Fortnow, and Carsten Lund. "Non-Deterministic Exponential Time has Two-Prover Interactive Protocols". In: *Computational Complexity* 1 (1991). Preliminary version appeared in FOCS '90., pp. 3–40.

[BFLS91] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. "Checking computations in polylogarithmic time". In: *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*. STOC '91. 1991, pp. 21–32.

[BFS20] Benedikt Bünz, Ben Fisch, and Alan Szepieniec. "Transparent SNARKs from DARK Compilers". In: *Proceedings of the 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. EUROCRYPT '20. 2020, pp. 677–706.

[BHRRS20] Alexander R. Block, Justin Holmgren, Alon Rosen, Ron D. Rothblum, and Pratik Soni. "Public-Coin Zero-Knowledge Arguments with (almost) Minimal Time and Space Overheads". In: *Proceedings of the 18th Theory of Cryptography Conference*. TCC '20. 2020, pp. 168–197.

[BISW17] Dan Boneh, Yuval Ishai, Amit Sahai, and David J. Wu. "Lattice-Based SNARGs and Their Application to More Efficient Obfuscation". In: *Proceedings of the 36th Annual International Conference on Theory and Applications of Cryptographic Techniques*. EUROCRYPT '17. 2017, pp. 247–277.

[BISW18] Dan Boneh, Yuval Ishai, Amit Sahai, and David J. Wu. "Quasi-Optimal SNARGs via Linear Multi-Prover Interactive Proofs". In: *Proceedings of the 37th Annual International Conference on Theory and Application of Cryptographic Techniques*. EUROCRYPT '18. 2018, pp. 222–255.

[BLNS20] Jonathan Bootle, Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. "A Non-PCP Approach to Succinct Quantum-Safe Zero-Knowledge". In: *Proceedings of the 40th Annual International Cryptology Conference*. CRYPTO '20. 2020, pp. 441–469.

[BMMTV19] Benedikt Bünz, Mary Maller, Pratyush Mishra, Nirvan Tyagi, and Psi Vesely. *Proofs for Inner Pairing Products and Applications*. Cryptology ePrint Archive, Report 2019/1177. 2019.

[CCHLRR18] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, and Ron D. Rothblum. *Fiat–Shamir From Simpler Assumptions*. Cryptology ePrint Archive, Report 2018/1004. 2018.

[CCKP19] Shuo Chen, Jung Hee Cheon, Dongwoo Kim, and Daejun Park. *Verifiable Computing for Approximate Computation*. IACR Cryptology ePrint Archive, Report 2019/762. 2019.

[CDESX18] Ronald Cramer, Ivan Damgård, Daniel Escudero, Peter Scholl, and Chaoping Xing. "SPD$\mathbb{Z}_{2^k}$: Efficient MPC mod 2^k for Dishonest Majority". In: *Proceedings of the 38th Annual International Cryptology Conference*. CRYPTO '18. 2018, pp. 769–798.

[CFFQR20] Matteo Campanelli, Antonio Faonio, Dario Fiore, Anaïs Querol, and Hadrián Rodríguez. *Lunar: a Toolbox for More Efficient Universal and Updatable zkSNARKs and Commit-and-Prove Extensions*. Cryptology ePrint Archive, Report 2020/1069. 2020.

[CHJKS20] Heewon Chung, Kyoohyung Han, Chanyang Ju, Myungsun Kim, and Jae Hong Seo. *Bulletproofs+: Shorter Proofs for Privacy-Enhanced Distributed Ledger*. Cryptology ePrint Archive, Report 2020/735. 2020.

[CHMMVW20] Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Noah Vesely, and Nicholas Ward. "Marlin: Preprocessing zkSNARKs with Universal and Updatable SRS". In: *Proceedings of the 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. EUROCRYPT '20. 2020, pp. 738–768.

[CMS19] Alessandro Chiesa, Peter Manohar, and Nicholas Spooner. "Succinct Arguments in the Quantum Random Oracle Model". In: *Proceedings of the 17th Theory of Cryptography Conference*. TCC '19. 2019, pp. 1–29.

[CMT12] Graham Cormode, Michael Mitzenmacher, and Justin Thaler. "Practical Verified Computation with Streaming Interactive Proofs". In: *Proceedings of the 4th Symposium on Innovations in Theoretical Computer Science*. ITCS '12. 2012, pp. 90–112.

[COS20] Alessandro Chiesa, Dev Ojha, and Nicholas Spooner. "Fractal: Post-Quantum and Transparent Recursive Proofs from Holography". In: *Proceedings of the 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. EUROCRYPT '20. 2020, pp. 769–793.

[CY20] Alessandro Chiesa and Eylon Yogev. "Barriers for Succinct Arguments in the Random Oracle Model". In: *Proceedings of the 18th Theory of Cryptography Conference*. TCC '20. 2020, pp. 47–76.

[GH98] Oded Goldreich and Johan Håstad. "On the complexity of interactive proofs with bounded communication". In: *Information Processing Letters* 67.4 (1998), pp. 205–214.

[GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. "Delegating Computation: Interactive Proofs for Muggles". In: *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*. STOC '08. 2008, pp. 113–122.

[GMNO18] Rosario Gennaro, Michele Minelli, Anca Nitulescu, and Michele Orrù. "Lattice-Based zk-SNARKs from Square Span Programs". In: *Proceedings of the 25th ACM Conference on Computer and Communications Security*. CCS '18. 2018, pp. 556–573.

[GN08] Nicolas Gama and Phong Q. Nguyen. "Predicting Lattice Reduction". In: *Proceedings of the 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. EUROCRYPT '08. 2008, pp. 31–51.

[GT20] Ashrujit Ghoshal and Stefano Tessaro. *Tight State-Restoration Soundness in the Algebraic Group Model*. Cryptology ePrint Archive, Report 2020/1351. 2020.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. "A Pseudorandom Generator from any One-way Function". In: *SIAM Journal on Computing* 28.4 (1999), pp. 1364–1396.

[JT20] Joseph Jaeger and Stefano Tessaro. "Expected-Time Cryptography: Generic Techniques and Applications to Concrete Soundness". In: *Proceedings of the 18th Theory of Cryptography Conference*. TCC '20. 2020, pp. 414–443.

[Kil92] Joe Kilian. "A note on efficient zero-knowledge proofs and arguments". In: *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*. STOC '92. 1992, pp. 723–732.

[LFKN92] Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. "Algebraic Methods for Interactive Proof Systems". In: *Journal of the ACM* 39.4 (1992), pp. 859–868.

[LMR19] Russell W. F. Lai, Giulio Malavolta, and Viktoria Ronge. "Succinct Arguments for Bilinear Group Arithmetic: Practical Structure-Preserving Cryptography". In: *Proceedings of the 26th ACM Conference on Computer and Communications Security*. CCS '19. 2019, pp. 2057–2074.

[Lee20] Jonathan Lee. *Dory: Efficient, Transparent arguments for Generalised Inner Products and Polynomial Commitments*. Cryptology ePrint Archive, Report 2020/1274. 2020.

[Mei13] Or Meir. "IP = PSPACE Using Error-Correcting Codes". In: *SIAM Journal on Computing* 42.1 (2013), pp. 380–403.

[Mon] URL: `https://github.com/monero-project/monero/tree/master/src/ringct`.

[PLS19] Rafaël del Pino, Vadim Lyubashevsky, and Gregor Seiler. "Short Discrete Log Proofs for FHE and Ring-LWE Ciphertexts". In: *Proceedings of the 22nd International Conference on Practice and Theory of Public-Key Cryptography*. PKC '19. 2019, pp. 344–373.

[Piv] *PIVX Implementation of Bulletproofs*. `https://github.com/PIVX-Project/PIVX/tree/Bulletproofs/src/libzerocoin`.

[RR20] Noga Ron-Zewi and Ron Rothblum. "Local Proofs Approaching the Witness Length". In: *Proceedings of the 61st Annual IEEE Symposium on Foundations of Computer Science*. FOCS '20. 2020.

[RV09] Guy N. Rothblum and Salil Vadhan. "Are PCPs Inherent in Efficient Arguments?" In: *Proceedings of the 24th IEEE Annual Conference on Computational Complexity*. CCC '09. 2009, pp. 81–92.

[Set20] Srinath Setty. "Spartan: Efficient and General-Purpose zkSNARKs Without Trusted Setup". In: *Proceedings of the 40th Annual International Cryptology Conference*. CRYPTO '20. 2020, pp. 704–737.

[Tha13] Justin Thaler. "Time-Optimal Interactive Proofs for Circuit Evaluation". In: *Proceedings of the 33rd Annual International Cryptology Conference*. CRYPTO '13. 2013, pp. 71–89.

[VSBW13] Victor Vu, Srinath Setty, Andrew J. Blumberg, and Michael Walfish. "A hybrid architecture for interactive verifiable computation". In: *Proceedings of the 34th IEEE Symposium on Security and Privacy*. Oakland '13. 2013, pp. 223–237.

[WTSTW18] Riad S. Wahby, Ioanna Tzialla, Abhi Shelat, Justin Thaler, and Michael Walfish. "Doubly-efficient zkSNARKs without trusted setup". In: *Proceedings of the 39th IEEE Symposium on Security and Privacy*. S&P '18. 2018, pp. 926–943.

[Wah+17] Riad S. Wahby, Ye Ji, Andrew J. Blumberg, Abhi Shelat, Justin Thaler, Michael Walfish, and Thomas Wies. "Full Accounting for Verifiable Outsourcing". In: *Proceedings of the 24th ACM Conference on Computer and Communications Security*. CCS '17. 2017, pp. 2071–2086.

[XZZPS19] Tiancheng Xie, Jiaheng Zhang, Yupeng Zhang, Charalampos Papamanthou, and Dawn Song. "Libra: Succinct Zero-Knowledge Proofs with Optimal Prover Computation". In: *Proceedings of the 39th Annual International Cryptology Conference*. CRYPTO '19. 2019, pp. 733–764.

[ZGKPP17] Yupeng Zhang, Daniel Genkin, Jonathan Katz, Dimitrios Papadopoulos, and Charalampos Papamanthou. "vSQL: Verifying Arbitrary SQL Queries over Dynamic Outsourced Databases". In: *Proceedings of the 38th IEEE Symposium on Security and Privacy*. S&P '17. 2017, pp. 863–880.

[ZXZS] Jiaheng Zhang, Tiancheng Xie, Yupeng Zhang, and Dawn Song. "Transparent Polynomial Delegation and Its Applications to Zero Knowledge Proof". In: *Proceedings of the 41st IEEE Symposium on Security and Privacy*. S&P '20, pp. 859–876.

[dalek18] dalek cryptography. *A pure-Rust implementation of Bulletproofs using Ristretto*. 2018. URL: `https://github.com/dalek-cryptography/bulletproofs`.