Sumcheck Arguments and their Applications [⋆]

Jonathan Bootle[1], Alessandro Chiesa[2] and Katerina Sotiraki[2]

[1] IBM Research – Zurich, Switzerland
`jbt@zurich.ibm.com`
[2] UC Berkeley, USA
`alexch@berkeley.edu`
`katesot@berkeley.edu`

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).
We define a class of sumcheck-friendly commitment schemes over modules that captures many examples of interest, and show that the sumcheck protocol applied to a 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 (discrete logarithms, pairings, groups of unknown order, lattices), providing one 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

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 arguing the security of the Fiat–Shamir transformation in the plain model [CCHLRR18], which is notoriously hard to analyze for other interactive proofs. Moreover, variants of the sumcheck protocol have spawned lines of research: the univariate sumcheck [BCRSVW19] was used in numerous succinct

[⋆] The full version of this paper is available at `https://eprint.iacr.org/2021/333`.

arguments [BCGGRS19; ZXZS20; CHMMVW20; COS20; CFFQR20; BFHVXZ20]; and the sumcheck protocol for tensor codes [Mei13] was used to obtain probabilistic proofs with linear-size proofs [BCGRS17; RR20] and 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 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 in 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 bilinear groups [LMR19], unknown-order groups [BFS20], and lattices [BLNS20]. They have also been formulated in more abstract settings: [BMMTV19] study sufficient properties of commitment schemes that enable folding techniques; and [AC20; ACF20; ACR20; BDFG20] study folding techniques for 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, *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 (prime-order groups, bilinear groups, unknown-order groups, lattices) and also enables us to answer open questions raised in prior works. We elaborate on these contributions below, and summarize the underlying technical ideas in Section 2.

(1) Sumcheck arguments. Recall that the sumcheck protocol is an interactive proof for statements of the form $\sum_{\underline{\omega} \in H^\ell} p(\underline{\omega}) = \tau$ 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).Let $\Sigma[R, M, H, \ell, \tau, \mathcal{C}, p]$ denote the sumcheck protocol for the statement $\sum_{\underline{\omega} \in H^\ell} p(\underline{\omega}) = \tau$ when $H \subseteq R$, $\tau \in M$, and $p \in M[X_1, \ldots, X_\ell]$, and the verifier uses the challenge set $\mathcal{C} \subseteq R$ to sample each round's challenge. (We explain later on in Section 2.1 why the sumncheck protocol over modules involves a given challenge set for the verifier.)

A *sumcheck argument* is, informally, a sumcheck protocol used to succinctly prove knowledge of openings for certain commitments (you run the sumcheck reduction followed by a cryptographic analogue of the consistency check). We say that a commitment scheme CM is *sumcheck-friendly* if the statement "I know m of length n such that $\mathsf{CM.Commit}\,(\mathsf{ck}, \mathsf{m}) = \mathsf{cm}$" can be rewritten as the statement "I know m of length n such that $\sum_{\underline{\omega} \in \{-1,1\}^{\log n}} f_{\mathsf{CM}}(p_{\mathsf{m}}(\underline{\omega}), p_{\mathsf{ck}}(\underline{\omega})) = \mathsf{cm}$" where the message polynomial $p_{\mathsf{m}}(\underline{X})$ is over an R-module $\mathbb{M}$, the key polynomial $p_{\mathsf{ck}}(\underline{X})$ is over an R-module $\mathbb{K}$, and the combiner function f_{CM} maps $\mathbb{M} \times \mathbb{K}$ to an R-module $\mathbb{C}$ (and is such that $f_{\mathsf{CM}}(p_{\mathsf{m}}(\underline{X}), p_{\mathsf{ck}}(\underline{X}))$ is a polynomial over $\mathbb{C}$). We observe that commitment schemes of interest are sumcheck-friendly, including various forms of Pedersen commitments (we elaborate on this later). Our main result is to construct a knowledge extractor for the sumcheck protocol applied to such statements, provided CM is *invertible* (a certain property that we discuss later on).

Theorem 1 (informal)**.** *Let* CM *be a sumcheck-friendly commitment scheme that is invertible. Let* cm *be a commitment to a message* m *using a commitment key* ck*. Then (a straightforward extension of)*

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

is an interactive argument of knowledge for an opening to cm *with respect to* ck *with knowledge error* $O(\frac{\log n}{|\mathcal{C}|})$*, where the polynomial in the numerator depends on* CM*. The round complexity is* $O(\log n)$ *and the communication complexity is* $O(\log n)$ *elements in* $\mathbb{C}$*. Moreover, if* f_{CM} *is a bilinear function, then the prover and verifier complexity is dominated by* $O(n)$ *operations in* $\mathbb{C}$*.*

The above informal statement omits many technical details, such as commitment randomness and relaxed notions of commitment opening necessary to express settings over lattices. Moreover, the informal statement fixes certain choices (such as choosing the summation domain $H = \{-1, 1\}$ and $\ell = \log n$ variables).

As we demonstrate in the full version of this paper, 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 from CM's invertibility. In turn invertibility may hold unconditionally or under certain hardness assumptions (we give examples of this in Section 2.3.2).[3]

(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 prior lines of work for secret-sharing schemes and multiparty computation protocols [CFIK03; CDESX18; ACDEY19; Abs+20] for supporting computations defined over rings.

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 $R_\bullet$ 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 $R_\bullet$, coefficient matrices $A, B, C \in R_\bullet^{n\times n}$ each containing at most $m = \Omega(n)$ non-zero entries, and an instance vector $\underline{x}$ over $R_\bullet$, is there a witness vector $\underline{w}$ over $R_\bullet$ such that $\underline{z} := (\underline{x}, \underline{w}) \in R_\bullet^n$ and $A\underline{z} \circ B\underline{z} = C\underline{z}$? (Here "$\circ$" denotes the entry-wise product of vectors over $R_\bullet$.)*

Theorem 2 (informal)**.** *Let R be a ring, M be an R-module, $\mathcal{C} \subseteq R$ a challenge space, and $I \subseteq R$ an ideal. If pairwise differences in $\mathcal{C}$ have suitable pseudoinverses in $R_\bullet := R/I$ and the bilinear relation assumption holds over M, then there is a zero-knowledge succinct argument of knowledge for the R1CS problem over $R_\bullet$. For $n \times n$ coefficient matrices with at most m non-zero entries, the argument has knowledge error $O(\frac{\log n}{|\mathcal{C}|})$, round complexity $O(\log n)$, communication complexity $O(\log n)$ elements of M and $O(1)$ elements of R, and prover and verifier complexity dominated by $O(m)$ operations in R and $O(n)$ operations in 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.[4]

Our Theorem 2 directly implies a solution to this open question. This may be surprising because the knowledge extractor for a sumcheck argument over lattices finds only a relaxed opening of a (sumcheck-friendly and invertible) commitment; this relaxed extraction occurs in many other lattice-based arguments of knowledge. This notwithstanding we still derive from it a knowledge extractor for the R1CS problem.

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

[4] 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.

Corollary 1 (informal)**.** *Let $R := \mathbb{Z}[X]/\langle X^d + 1\rangle$ for d a power of 2. Let p and q be primes with q sufficiently larger than p. Assuming hardness of the SIS problem over R/qR, there is an argument of knowledge for R1CS over $R_\bullet := R/pR$ with knowledge-soundness error $O(\frac{\log n}{d})$, round complexity $O(\log n)$, communication complexity dominated by $O(\log n)$ elements of R/qR, and prover and verifier complexity dominated by $O(m)$ operations in $R_\bullet$ and $O(n)$ operations in R/qR.*

Our new lattice-based argument system shows that one can succinctly prove general relations over rings pertinent to lattice cryptography, 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 post-quantum succinct arguments are from hash functions [CMS19; CMSZ21] 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 sumcheck-friendly commitment CM in Theorem 1 we obtain succinct arguments of knowledge for different relations of interest, as we now explain.

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 *generalized 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 generalized Pedersen commitment scheme thus yields succinct protocols for opening Pedersen commitments in different settings, such as prime-order groups, bilinear groups, unknown-order groups, and lattices.

We also study instantiations that capture richer functionalities.

- *Linear-function commitments:* the commitment includes a commitment to the scalar product of a public (query) message and a secret message. This draws inspiration from [AC20] which considers linear-function commitments in the prime-order group setting, bilinear group setting, and strong RSA setting.
- *Scalar-product commitments:* the commitment includes a commitment to the scalar product of two secret parts of the message. This draws inspiration from [BCCGP16; BBBPWM18; BMMTV19] which consider bilinear commitment schemes for prime-order or bilinear groups. 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*.

	prime-order groups (DL assumption)	bilinear groups (double-pairing assumption)	unknown-order groups (order assumption)	ideal lattices (SIS assumption)
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-friendly commitment	**sumcheck arguments from this work**			

Fig. 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 (along with corresponding sufficient cryptographic assumptions). 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 and settings 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. Below we provide 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 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 where the sumcheck prover sends polynomials expressed in a *different monomial basis*.
- [CHJKS20] gives weighted inner-product arguments to improve concrete efficiency. This corresponds to a sumcheck argument for *weighted-sums of polynomial evaluations* (see the full version of the paper for details).

– [PLS19] gives a zero-knowledge version of folding techniques that achieves better concrete efficiency by using less prover randomness. This relates to *derandomizing a zero-knowledge sumcheck argument*.
– [BMMTV19; BFS20] consider subprotocols for delegating expensive verifier computation to the prover. This corresponds to *delegating polynomial evaluation*, to help the verifier outsource evaluating the commitment key polynomial. Sumcheck arguments neatly conceptualize the role of polynomials in folding protocols and simplify the task of applying delegation protocols in other settings (see discussed further in the full version).
– Like [BMMTV19; ACF20; BDFG20], sumcheck arguments capture optimizations of folding techniques that compress several target commitment values into one (e.g., the optimization from [BCCGP16] to [BBBPWM18]) as sumcheck arguments applied to alternative commitment schemes.

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

Looking ahead, the new perspective offered by sumcheck arguments, with the sumcheck protocol at their core, makes it easier to explore new design options and optimizations for succinct arguments, especially so for those that have already been studied for the (information-theoretic) sumcheck protocol.

Existing analyses of the sumcheck protocol may also inspire analogous ones for sumcheck arguments. For example, the sumcheck protocol can be made non-interactive via the Fiat–Shamir transformation, where the verifier's messages are replaced by the outputs of a hash function. Jawale et al. [JKKZ20] show that the result is a non-interactive argument provided the hash function is *lossy correlation-intractable* (and construct such hash functions based on the LWE assumption). This seems to provide a starting point for studying the security of sumcheck arguments under the Fiat–Shamir transformation.

1.3 Related work

Folding techniques. Figure 1 summarizes the main relationship between sumcheck arguments for sumcheck-friendly 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, Cramer, and Fehr [ACF20] present folding techniques for pre-images of general group homomorphisms over prime-order groups. These were extended 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 using computational assumptions.

Both general group homomorphisms and doubly-homomorphic commitment schemes are special cases of sumcheck-friendly commitment schemes. Our work also finds the same distinction between proofs and arguments: our sumcheck argument for "linear" commitment schemes such as the generalized Pedersen commitment scheme (and linear-function commitments) do not require computational assumptions, whereas our sumcheck argument for "quadratic" commitment schemes require computational assumptions.

Reductions from NP-complete problems. Attema and Cramer [AC20] construct zero-knowledge succinct arguments for NP-complete relations by (i) using secret-sharing techniques to interactively reduce NP statements to linear relations (under computational assumptions), and then (ii) relying on succinct arguments for linear relations. This "linearization" requires the prover to perform polynomial arithmetic on high-degree polynomials, and hence an efficient realization would likely rely on FFTs. FFTs require linear space-complexity for the prover, and prevent the prover from being implemented in logarithmic space as in the sumcheck protocol [CMT12] or other succinct arguments based on folding protocols [BHRRS20]. In contrast, we reduce NP statements to bilinear relations such as scalar-product relations, and then rely on succinct arguments for scalar products; this reduction can be performed via a linear number of cryptographic operations, and without relying on FFTs.

1.4 Concurrent work

Attema, Cramer, and Kohl [ACK21] construct zero-knowledge succinct arguments for NP based on the SIS assumption, using folding techniques for lattices. As with [AC20], their construction uses secret-sharing techniques which are likely to rely on FFTs and lead to a prover with large space complexity. Moreover, the techniques in [ACK21] are for lattices, while our techniques based on sumcheck arguments provide a general framework in which lattices are a special case. Additionally, [ACK21] give a detailed analysis of the knowledge error of their lattice-based folding techniques, which was not present in [BLNS20], and establish that the knowledge error can be reduced using parallel repetition.

Albrecht and Lai [AL21] study a variant of the folding techniques in [BLNS20], instantiated in a different choice of ring which offers exact proofs (rather than proofs with relaxed knowledge extraction) and various efficiency advantages. Like [ACK21], they also analyze the knowledge error of their folding techniques, and prove results relating relaxed extraction to ring structure. We are optimistic that their ideas can be incorporated into our sumcheck-based framework.

Ganesh, Nitulescu, and Soria-Vazquez [GNS21] model NP relations over rings and give a generic construction of designated-verifier zero-knowledge SNARKs using techniques related to prior lattice-based SNARK constructions [BISW17; BISW18; GMNO18].

Block et al. [BHRRS21] study a variant of the commitment scheme of [BFS20] in groups of unknown order that is compatible with a streaming formalism, and give space-efficient arguments for NP languages. We are optimistic that their ideas can be incorporated into our sumcheck-based framework.

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 sumcheck protocol for polynomials over modules. Then in Section 2.2 we present a succinct zero-knowledge argument for Pedersen commitments based on the sumcheck protocol. In Section 2.3 we show how to lift this protocol to any "sumcheck-friendly" commitment, but still in the setting of prime-order groups. Finally in Section 2.4 we explain the main considerations in generalizing further to commitments over rings, and in Section 2.5 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 2.6 we discuss a generic scalar-product protocol built from sumcheck arguments, and then in Section 2.7 we explain how it enables us to obtain zero-knowledge succinct arguments for R1CS over rings (Theorem 2 and in particular Corollary 1). In the full version, we also discuss how we obtain polynomial commitment schemes over rings from sumcheck arguments.

2.1 Sumcheck protocol over modules

The sumcheck protocol [LFKN92] directly extends to work with polynomials over *modules*. The prover P_{SC} and verifier V_{SC} receive a sumcheck instance $\mathbb{x}_{\mathrm{SC}} = (R, M, H, \ell, \tau, \mathcal{C})$, where R is a ring, M is a module over R, H is a subset of R, ℓ is a number of variables, $\tau \in M$ is a claimed sum, and $\mathcal{C} \subseteq R$ is a sampling set (more about this below). The prover P_{SC} additionally receives a polynomial $p \in M[X_1, \ldots, X_\ell]$ such that $\sum_{\underline{\omega} \in H^\ell} p(\underline{\omega}) = \tau$. The protocol has ℓ rounds and works as follows.

1. For $i = 1, \ldots, \ell$:
   (a) P_{SC} sends to V_{SC} the polynomial $q_i(X) := \sum_{\omega_{i+1},\ldots,\omega_\ell \in H} p(r_1, \ldots, r_{i-1}, X, \omega_{i+1}, \ldots, \omega_\ell) \in M[X]$;
   (b) V_{SC} sends to P_{SC} a random challenge $r_i \leftarrow \mathcal{C}$.
2. V_{SC} checks that $\sum_{\omega_1 \in H} q_1(\omega_1) = \tau$ and, for $i \in \{2, \ldots, \ell\}$, that $\sum_{\omega_i \in H} q_i(\omega_i) = q_{i-1}(r_{i-1})$.
3. If the checks pass then V_{SC} sets $v := q_\ell(r_\ell) \in M$ and outputs the tuple $((r_1, \ldots, r_\ell), v)$.

The security guarantee of the sumcheck protocol, which requires $\mathcal{C}$ to be a sampling set, is given below.

Definition 1. *We say that $\mathcal{C} \subseteq R$ is a* **sampling set** *for the R-module M if for every distinct $c_1, c_2 \in \mathcal{C}$ the map that sends $m \in M$ to $(c_1 - c_2) \cdot m \in M$ is injective.*

Lemma 1. *Let $\mathbb{x}_{\mathrm{SC}} = (R, M, H, \ell, \tau, \mathcal{C})$ be a sumcheck instance and a polynomial $p \in M[X_1, \ldots, X_\ell]$ of total degree d. If $\mathcal{C}$ is a sampling set for M then the following holds.*
– Completeness. *If $\sum_{\underline{\omega} \in H^\ell} p(\underline{\omega}) = \tau$ then $\Pr_{\underline{r} \leftarrow \mathcal{C}^\ell}[\langle P_{\mathrm{SC}}(\mathbb{x}_{\mathrm{SC}}, p), V_{\mathrm{SC}}(\mathbb{x}_{\mathrm{SC}}; \underline{r})\rangle = (\underline{r}, p(\underline{r}))] = 1$.*
– Soundness. *If $\sum_{\underline{\omega} \in H^\ell} p(\underline{\omega}) \neq \tau$ then, for every $\tilde{P}_{\mathrm{SC}}$, $\Pr_{\underline{r} \leftarrow \mathcal{C}^\ell}[\langle \tilde{P}_{\mathrm{SC}}, V_{\mathrm{SC}}(\mathbb{x}_{\mathrm{SC}}; \underline{r})\rangle = (\underline{r}, p(\underline{r}))] < \frac{\ell d}{|\mathcal{C}|}$.*

Above $\langle A, V_{\mathrm{SC}}(\mathbb{x}_{\mathrm{SC}}; \underline{r})\rangle$ is the output of $V_{\mathrm{SC}}(\mathbb{x}_{\mathrm{SC}})$ when interacting with algorithm A using randomness $\underline{r}$.

The lemma directly follows from a generalization of the Schwartz–Zippel lemma over modules.

Lemma 2. *Let R be a ring, M an R-module, and $f \in M[X_1, \ldots, X_\ell]$ a non-zero polynomial of total degree D. If $\mathcal{C}$ is a sampling set for M then $\Pr_{\underline{r} \leftarrow \mathcal{C}^\ell}[f(\underline{r}) = 0] \leq \frac{D}{|\mathcal{C}|}$.*

The proof of Lemma 2 follows the same approach 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 in turn is used in the base case and in the inductive step.

The sumcheck protocol in the special case when $M = R$ has been used before, e.g., in [CCKP19].

2.2 Sumcheck argument for Pedersen commitments

We describe a cryptographic protocol for proving knowledge of an opening of a Pedersen commitment, whose main subroutine is the sumcheck protocol. We refer to such a protocol as a *sumcheck argument*. Note that 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.

Definition 2. *We index the entries of a vector $\underline{v}$ of length $n = 2^\ell$ via binary strings $(i_1, \ldots, i_\ell) \in \{0,1\}^\ell$, and define the corresponding polynomial $p_{\underline{v}}(X_1, \ldots, X_\ell) := \sum_{i_1,\ldots,i_\ell \in \{0,1\}} v_{i_1,\ldots,i_\ell} X_1^{i_1} \cdots X_\ell^{i_\ell}$.*

Protocol 1: sumcheck argument for Pedersen commitments

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

The prover and verifier engage in a sumcheck protocol for the instance

$$\mathbb{x}_{\mathrm{SC}} := (R = \mathbb{F},\ M = \mathbb{G},\ H = \{-1, 1\},\ \ell = \log n,\ \tau = 2^\ell \mathsf{C},\ \mathcal{C} = \mathbb{F})$$

where the prover uses the polynomial $p(\underline{X}) := p_{\underline{a}}(\underline{X}) \cdot p_{\underline{\mathsf{G}}}(\underline{X})$. After the end of the sumcheck protocol, the prover learns $\underline{r} \in \mathbb{F}^\ell$ and the verifier learns $(\underline{r}, v) \in \mathbb{F}^\ell \times \mathbb{G}$. Then the prover computes and sends $p_{\underline{a}}(\underline{r}) \in \mathbb{F}$ to the verifier, and the verifier computes $p_{\underline{\mathsf{G}}}(\underline{r}) \in \mathbb{G}$ and checks that $p_{\underline{a}}(\underline{r}) \cdot p_{\underline{\mathsf{G}}}(\underline{r}) = v$.

We begin by explaining why Protocol 1 is mathematically well-defined. The "multiplication" operation implicit in the expression $p_{\underline{a}}(\underline{X}) \cdot p_{\underline{\mathsf{G}}}(\underline{X})$, 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 $\mathbb{F} \times \mathbb{G} \to \mathbb{G}$. For example, consider the polynomials

$p_1(X) = a + a' \cdot X \in \mathbb{F}[X]$ and $p_2(X) = \mathsf{G} + X \cdot \mathsf{G}' \in \mathbb{G}[X]$, and let $r \in \mathbb{F}$. The product of $p_1(r)$ and $p_2(r)$ can be written as follows:

$$\begin{aligned} p_1(r) \cdot p_2(r) &= (a + a'r) \cdot (\mathsf{G} + r \cdot \mathsf{G}') = a \cdot (\mathsf{G} + r \cdot \mathsf{G}') + a'r \cdot (\mathsf{G} + r \cdot \mathsf{G}') \\ &= a \cdot \mathsf{G} + ar \cdot \mathsf{G}' + a'r \cdot \mathsf{G} + a'r^2 \cdot \mathsf{G}' \\ &= a \cdot \mathsf{G} + r \cdot (a \cdot \mathsf{G}' + a' \cdot \mathsf{G}) + r^2 \cdot (a' \cdot \mathsf{G}') \ , \end{aligned}$$

where the second and third equalities follow from the *bilinear properties* of scalar multiplication.[5] This holds for any $r \in \mathbb{F}$, and so it makes sense to define the "scalar multiplication" of $p_1(X)$ and $p_2(X)$:

$$p_1(X) \cdot p_2(X) = (a + a'X) \cdot (\mathsf{G} + X \cdot \mathsf{G}') := a \cdot \mathsf{G} + X \cdot (a \cdot \mathsf{G}' + a' \cdot \mathsf{G}) + X^2 \cdot (a' \cdot \mathsf{G}') \ .$$

The polynomial $p_{\underline{a}}(\underline{X}) \cdot p_{\underline{\mathsf{G}}}(\underline{X})$, whose coefficients lie in $\mathbb{G}$, is defined this way.

Completeness of Protocol 1 follows from the fact that $\sum_{\underline{\omega} \in \{-1,1\}^n} p_{\underline{a}}(\underline{\omega}) \cdot p_{\underline{\mathsf{G}}}(\underline{\omega}) = 2^\ell \langle \underline{a}, \underline{\mathsf{G}} \rangle$. Indeed, each contribution to $\sum_{\underline{\omega} \in \{-1,1\}^\ell} p_{\underline{a}}(\underline{\omega}) \cdot p_{\underline{\mathsf{G}}}(\underline{\omega})$ corresponds to the monomials of $p_{\underline{a}}(\underline{X}) \cdot p_{\underline{\mathsf{G}}}(\underline{X})$ of the form $X_1^{2i_1} \cdots X_\ell^{2i_\ell}$. The coefficient of $X_1^{2i_1} \cdots X_\ell^{2i_\ell}$ in $p_{\underline{a}}(\underline{X}) \cdot p_{\underline{\mathsf{G}}}(\underline{X})$ arises from a multiplication of the monomials in the terms $a_{i_1,\ldots,i_\ell} X_1^{i_1} \cdots X_\ell^{i_\ell}$ and $\mathsf{G}_{i_1,\ldots,i_\ell} X_1^{i_1} \cdots X_\ell^{i_\ell}$, which multiply to give $a_{i_1,\ldots,i_\ell} \cdot \mathsf{G}_{i_1,\ldots,i_\ell} \cdot X_1^{2i_1} \cdots X_\ell^{2i_\ell}$. Thus, $\sum_{\underline{\omega} \in \{-1,1\}^\ell} p_{\underline{a}}(\underline{\omega}) \cdot p_{\underline{\mathsf{G}}}(\underline{\omega}) = 2^\ell \langle \underline{a}, \underline{\mathsf{G}} \rangle$.

The security guarantee of Protocol 1 is different from that of the sumcheck protocol. The sumcheck protocol has a soundness guarantee: if the polynomial p does not have the claimed sum τ then the verifier accepts with small probability. In contrast, Protocol 1 has a **knowledge soundness** guarantee: there exists an extractor that, given a suitable collection of accepting transcripts for a given commitment key $\underline{\mathsf{G}}$ and commitment C, efficiently finds an opening $\underline{a}$ such that $\mathsf{C} = \langle \underline{a}, \underline{\mathsf{G}} \rangle$.

This difference makes sense: any given Pedersen commitment C can *always* be expressed as a scalar product of some opening $\underline{a}$ and the commitment key generators $\underline{\mathsf{G}}$; in fact, there are many different possible openings $\underline{a}$ for which this is true! Therefore, soundness is not a meaningful notion for Protocol 1.

The security guarantee is summarized by the following lemma, whose proof we sketch in Section 2.2.1.

Lemma 3 (informal). *Protocol 1 satisfies the following for every key* $\underline{\mathsf{G}} \in \mathbb{G}^n$ *and commitment* $\mathsf{C} \in \mathbb{G}$.
- Completeness. *For every* $\underline{a} \in \mathbb{F}^n$ *such that* $\mathsf{C} = \langle \underline{a}, \underline{\mathsf{G}} \rangle$, $\Pr[\langle \mathbf{P}(\underline{\mathsf{G}}, \mathsf{C}, \underline{a}), \mathbf{V}(\underline{\mathsf{G}}, \mathsf{C}) \rangle = 1] = 1$.
- Knowledge soundness. *Given a suitable tree of accepting transcripts for* $\mathbf{V}(\underline{\mathsf{G}}, \mathsf{C})$*, one can efficiently extract an opening* $\underline{a} \in \mathbb{F}^n$ *such that* $\mathsf{C} = \langle \underline{a}, \underline{\mathsf{G}} \rangle$.

Perhaps surprisingly, Protocol 1 is *equivalent* to the "split-and-fold" knowledge protocol for Pedersen commitments introduced in [BCCGP16] (we describe this equivalence in the full version of the paper). Moreover, knowledge soundness can be established without relying on any computational assumptions, a fact that was noted for the "split-and-fold" knowledge protocol in [ACF20; BDFG20].

[5] 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 $a \cdot (\mathsf{G} + \mathsf{G}') = a \cdot \mathsf{G} + a \cdot \mathsf{G}'$.

2.2.1 Proof sketch of Lemma 3

We discuss knowledge soundness. The extractor takes as input 3^ℓ accepting transcripts arranged in a 3-ary tree of depth ℓ, with each path from the root to the leaf identified by a choice of verifier randomness $r_1, \dots, r_\ell \in \mathbb{F}$. For $i \in [\ell]$, the node at layer $i-1$ corresponding to path $r_1, \dots, r_{i-1} \in \mathbb{F}$ is labeled with the message sent by the prover given challenges $r_1, \dots, r_{i-1}$ and has three children nodes each corresponding to a distinct challenge $r_i^{(j)} \in \mathbb{F}$. For $i \in [\ell]$, a prover message for the layer $i-1$ is a quadratic polynomial $q_i[r_1, \dots, r_{i-1}] \in \mathbb{G}[X]$ sent by the prover in the sumcheck protocol given challenges $r_1, \dots, r_{i-1}$; and a prover message for the layer ℓ is an opening $w[r_1, \dots, r_\ell] \in \mathbb{F}$ sent by the prover after the sumcheck protocol. Since transcripts are accepting, we know that: $\sum_{\omega_1 \in \{-1,1\}} q_1(\omega_1) = 2^\ell \mathsf{C}$; for $i \in \{2, \dots, \ell\}$, $\sum_{\omega \in \{-1,1\}} q_i[r_1, \dots, r_{i-1}](\omega) = q_{i-1}[r_1, \dots, r_{i-2}](r_{i-1})$; and $w[r_1, \dots, r_\ell] \cdot p_{\underline{\mathsf{G}}}(r_1, \dots, r_\ell) = q_\ell[r_1, \dots, r_{\ell-1}](r_\ell)$.

The extractor works inductively, processing each layer of the tree starting from the ℓ-th layer and moving upwards towards the root. For $i = \ell, \dots, 1$ and for every path $(r_1, \dots, r_{i-1}) \in \mathbb{F}^{i-1}$ in the transcript tree with children $\{r_i^{(j)}\}_{j \in [3]}$, the extractor works as follows.

1. Let $\underline{\mathsf{G}}[r_1, \dots, r_{i-1}] \in \mathbb{G}^{n/2^{i-1}}$ be the coefficients of $p_{\underline{\mathsf{G}}}(r_1, \dots, r_{i-1}, X_i, \dots, X_\ell)$, and let $\underline{\mathsf{G}}_0[r_1, \dots, r_{i-1}]$ and $\underline{\mathsf{G}}_1[r_1, \dots, r_{i-1}]$ be the coefficients for monomials without X_i and with X_i respectively. For $j \in [3]$, let $\underline{\mathsf{G}}'[r_1, \dots, r_{i-1}, r_i^{(j)}] := \underline{\mathsf{G}}[r_1, \dots, r_{i-1}] + r_i^{(j)} \cdot \underline{\mathsf{G}}_1[r_1, \dots, r_{i-1}] \in \mathbb{G}^{n/2^i}$ be the coefficients of $p_{\underline{\mathsf{G}}}(r_1, \dots, r_{i-1}, r_i^{(j)}, X_{i+1}, \dots, X_\ell)$.
2. We inductively know, for each $j \in [3]$, an opening $w[r_1, \dots, r_{i-1}, r_i^{(j)}] \in \mathbb{F}^{n/2^i}$ to the commitment $q_i[r_1, \dots, r_{i-1}](r_i^{(j)}) \in \mathbb{G}$ with respect to the key $\underline{\mathsf{G}}'[r_1, \dots, r_{i-1}, r_i^{(j)}]$:
$$\begin{aligned}
\langle w[r_1, \dots, r_{i-1}, r_i^{(1)}], \underline{\mathsf{G}}'[r_1, \dots, r_{i-1}, r_i^{(1)}] \rangle &= q_i[r_1, \dots, r_{i-1}](r_i^{(1)}) \ , \\
\langle w[r_1, \dots, r_{i-1}, r_i^{(2)}], \underline{\mathsf{G}}'[r_1, \dots, r_{i-1}, r_i^{(2)}] \rangle &= q_i[r_1, \dots, r_{i-1}](r_i^{(2)}) \ , \\
\langle w[r_1, \dots, r_{i-1}, r_i^{(3)}], \underline{\mathsf{G}}'[r_1, \dots, r_{i-1}, r_i^{(3)}] \rangle &= q_i[r_1, \dots, r_{i-1}](r_i^{(3)}) \ .
\end{aligned}$$
3. Since the polynomial $q_i[r_1, \dots, r_{i-1}]$ is quadratic, we can use linear algebra on the above three equations to compute a quadratic polynomial $\pi[r_1, \dots, r_{i-1}] \in \mathbb{F}^{n/2^{i-1}}[X]$ such that $\langle \pi[r_1, \dots, r_{i-1}](X), \underline{\mathsf{G}}[r_1, \dots, r_{i-1}] \rangle = q_i[r_1, \dots, r_{i-1}](X)$. Then we can obtain an opening $w[r_1, \dots, r_{i-1}] \in \mathbb{F}^{n/2^{i-1}}$ such that $\langle w[r_1, \dots, r_{i-1}], \underline{\mathsf{G}}[r_1, \dots, r_{i-1}] \rangle = \sum_{\omega \in \{-1,1\}} q_i[r_1, \dots, r_{i-1}](\omega)$. Observe that:
   - If $i > 1$, the verifier's checks imply that $\sum_{\omega \in \{-1,1\}} q_i[r_1, \dots, r_{i-1}](\omega) = q_{i-1}[r_1, \dots, r_{i-2}](r_{i-1})$, and so $w[r_1, \dots, r_{i-1}]$ is an opening to the commitment $q_{i-1}[r_1, \dots, r_{i-2}](r_{i-1})$ under the key $\underline{\mathsf{G}}[r_1, \dots, r_{i-1}]$.
   - If $i = 1$ (this is the last iteration) then the verifier's checks imply that $\sum_{\omega_1 \in \{-1,1\}} q_1(\omega_1) = 2^\ell \mathsf{C}$, and so w is an opening to the commitment $2^\ell \mathsf{C}$ under the key $\underline{\mathsf{G}}$. Dividing by 2^ℓ yields the desired opening.

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 for the general setting of sumcheck-friendly commitments (see Sections 2.3 and 2.4).

2.3 Sumcheck argument for sumcheck-friendly commitments

We explain how to formulate a sumcheck argument for proving knowledge of an opening for any commitment scheme that satisfies certain functionality and security properties. We proceed in two steps: in Section 2.3.1 we focus on the special case of scalar product protocols under Pedersen commitments to gain intuition, and then in Section 2.3.2 we extend this to apply to a *sumcheck-friendly* commitment.

2.3.1 Scalar-products under Pedersen commitments

In Section 2.2 we have seen how to construct a sumcheck argument for Pedersen commitments. We now write a sumcheck argument that proves knowledge of openings of *two* Pedersen commitments such that the scalar product of the two openings is a publicly-known value. That is, we obtain a knowledge protocol for the commitment scheme CM that, given a commitment key $(\underline{G}, \underline{H})$, maps a message $(\underline{a}, \underline{b})$ to

$$\mathsf{CM.Commit}\Big((\underline{G}, \underline{H}), (\underline{a}, \underline{b})\Big) := (\langle \underline{a}, \underline{G} \rangle, \langle \underline{b}, \underline{H} \rangle, \langle \underline{a}, \underline{b} \rangle) \ .$$

Protocol 2: sumcheck argument for scalar-products under Pedersen commitments

For $n = 2^\ell$, the prover and verifier receive as input commitment keys $\underline{G}, \underline{H} \in \mathbb{G}^n$, commitments $\mathsf{C}_a, \mathsf{C}_b \in \mathbb{G}$ and target value $t \in \mathbb{F}$. The prover also receives as input openings $\underline{a}, \underline{b} \in \mathbb{F}^n$ such that $\mathsf{C}_a = \langle \underline{a}, \underline{G} \rangle$, $\mathsf{C}_b = \langle \underline{b}, \underline{H} \rangle$ and $t = \langle \underline{a}, \underline{b} \rangle$. (I.e., such that $\mathsf{CM.Commit}\big((\underline{G}, \underline{H}), (\underline{a}, \underline{b})\big) = (\mathsf{C}_a, \mathsf{C}_b, t)$.)

The prover and verifier engage in a sumcheck protocol for the instance $\mathbb{x}_{\mathrm{SC}} := (R = \mathbb{F}, M = \mathbb{G} \times \mathbb{G} \times \mathbb{F}, H = \{-1, 1\}, \ell = \log n, \tau = (2^\ell \mathsf{C}_a, 2^\ell \mathsf{C}_b, 2^\ell t), \mathcal{C} = \mathbb{F})$ where the prover uses the polynomial $p(\underline{X}) := \Big(p_{\underline{a}}(\underline{X}) \cdot p_{\underline{G}}(\underline{X}),\ p_{\underline{b}}(\underline{X}) \cdot p_{\underline{H}}(\underline{X}),\ p_{\underline{a}}(\underline{X}) \cdot p_{\underline{b}}(\underline{X})\Big) \in (\mathbb{G} \times \mathbb{G} \times \mathbb{F})[X_1, \ldots, X_\ell]$.

After the end of the sumcheck protocol, the prover learns $\underline{r} \in \mathbb{F}^\ell$ and the verifier learns $(\underline{r}, v) \in \mathbb{F}^\ell \times (\mathbb{G} \times \mathbb{G} \times \mathbb{F})$. Then the prover computes and sends $p_{\underline{a}}(\underline{r}), p_{\underline{b}}(\underline{r}) \in \mathbb{F}$ to the verifier, and the verifier computes $p_{\underline{G}}(\underline{r}), p_{\underline{H}}(\underline{r}) \in \mathbb{G}$ and checks that $(p_{\underline{a}}(\underline{r}) \cdot p_{\underline{G}}(\underline{r}),\ p_{\underline{b}}(\underline{r}) \cdot p_{\underline{H}}(\underline{r}),\ p_{\underline{a}}(\underline{r}) \cdot p_{\underline{b}}(\underline{r})) = v$.

Similarly to Section 2.2, the first and second components of the polynomial $p(\underline{X})$ are well-defined because of the bilinearity of scalar multiplication from $\mathbb{F} \times \mathbb{G}$ to $\mathbb{G}$; moreover, the third component of $p(\underline{X})$ is well-defined because it involves the product of two polynomials over $\mathbb{F}$.

Protocol 2 is complete because

$$\sum_{\underline{\omega}\in\{-1,1\}^\ell} p_{\underline{a}}(\underline{\omega}) p_{\underline{G}}(\underline{\omega}) = 2^\ell \langle \underline{a}, \underline{G} \rangle \,, \quad \sum_{\underline{\omega}\in\{-1,1\}^\ell} p_{\underline{b}}(\underline{\omega}) p_{\underline{H}}(\underline{\omega}) = 2^\ell \langle \underline{b}, \underline{H} \rangle \,, \quad \sum_{\underline{\omega}\in\{-1,1\}^\ell} p_{\underline{a}}(\underline{\omega}) p_{\underline{b}}(\underline{\omega}) = 2^\ell \langle \underline{a}, \underline{b} \rangle \ .$$

Moreover, one can show that Protocol 2 satisfies the following knowledge-soundness property: there exists an extractor that, given a suitable collection of accepting transcripts for a given commitment key $(\underline{G}, \underline{H})$ and commitment $\mathsf{C} = (\mathsf{C}_a, \mathsf{C}_b, t)$, efficiently finds an opening $(\underline{a}, \underline{b})$ such that $\mathsf{C} = \mathsf{CM.Commit}((\underline{G}, \underline{H}), (\underline{a}, \underline{b}))$, assuming that the discrete logarithm problem is hard over $\mathbb{G}$. Proving knowledge soundness follows a similar approach to that for Protocol 1 sketched in Section 2.2.1. The main difference is that "inverting" from a level to the previous one involves not only solving linear equations to find openings of commitments corresponding to the first two components of the polynomial $p(\underline{X})$, but also arguing that the scalar-product of these openings equals the third component of the polynomial $p(\underline{X})$. This step relies on the hardness of the discrete logarithm problem over $\mathbb{G}$ (which one may have assumed anyway to make the commitment binding). This is different from Protocol 1, where no assumptions were necessary to establish knowledge soundness, and intuitively is because the commitment scheme involves a quadratic, rather than linear, computation on the message.

2.3.2 Extending to any sumcheck-friendly commitment

The commitments used in Protocols 1 and 2 are examples of a *sumcheck-friendly* commitment scheme. Below we give an informal definition (which omits technicalities such as how commitment randomness is handled).

Definition 3 (informal)**.** *Let $\mathbb{F}$ be a prime-order field and let $\mathbb{M}, \mathbb{K}, \mathbb{C}$ be $\mathbb{F}$-linear spaces. A commitment scheme* CM *is* **sumcheck-friendly** *if there exists an efficient function $f_{\mathsf{CM}} \colon \mathbb{M} \times \mathbb{K} \to \mathbb{C}$ such that for every commitment key* ck *and message* m *it holds that* $\mathsf{CM.Commit}(\mathsf{ck}, \mathsf{m}) = \sum_{\underline{\omega}\in H^\ell} f_{\mathsf{CM}}(p_{\mathsf{m}}(\underline{\omega}), p_{\mathsf{ck}}(\underline{\omega}))$ *where: (i) $H \subseteq \mathbb{F}$ is a domain and $\ell \in \mathbb{N}$ a number of variables; (ii) $p_{\mathsf{m}}(\underline{X}) \in \mathbb{M}[\underline{X}]$ can be efficiently obtained from the message* m *(and, conversely,* m *can be efficiently obtained from $p_{\mathsf{m}}(\underline{X})$); (iii) $p_{\mathsf{ck}}(\underline{X}) \in \mathbb{K}[\underline{X}]$ can be efficiently obtained from the commitment key* ck*; (iv) $f_{\mathsf{CM}}(p_{\mathsf{m}}(\underline{X}), p_{\mathsf{ck}}(\underline{X})) \in \mathbb{C}[\underline{X}]$ is a polynomial.*

We can obtain an opening protocol for CM via a sumcheck argument.

Protocol 3: sumcheck argument for sumcheck-friendly commitments

For $n = 2^\ell$, the prover and verifier receive as input commitment key ck and commitment cm. The prover also receives as input an opening m such that $\mathsf{cm} = \mathsf{CM.Commit}(\mathsf{ck}, \mathsf{m})$.

The prover and verifier engage in a sumcheck protocol for the instance

$$\mathbb{x}_{\text{SC}} := (R = \mathbb{F}, M = \mathbb{C}, H, \ell, \tau = \mathsf{cm}, \mathcal{C} = \mathbb{F})$$

where the prover uses the polynomial $p_{\mathsf{m},\mathsf{ck}}(\underline{X}) := f_{\mathsf{CM}}(p_{\mathsf{m}}(\underline{X}), p_{\mathsf{ck}}(\underline{X}))$. At the end of the sumcheck protocol, the prover learns $\underline{r} \in \mathbb{F}^\ell$ and the verifier learns

$(\underline{r}, v) \in \mathbb{F}^\ell \times \mathbb{C}$. Then the prover computes and sends $p_{\mathsf{m}}(\underline{r})$ to the verifier, and the verifier computes $p_{\mathsf{ck}}(\underline{r})$ and checks that $f_{\mathsf{CM}}(p_{\mathsf{m}}(\underline{r}), p_{\mathsf{ck}}(\underline{r})) = v$.

The above opening protocol for the sumcheck-friendly commitment scheme CM has perfect completeness, and also has knowledge soundness if CM is *invertible* (a property that we discuss shortly).

Theorem 3 (informal)**.** *Let* CM *be a sumcheck-friendly commitment scheme. If* CM *is invertible then Protocol 3 is an opening protocol for* CM*: there exists an extractor that given a key* ck*, commitment* cm*, and a suitable tree of accepting transcripts for* $(\mathsf{ck}, \mathsf{cm})$*, finds an opening* m *such that* $\mathsf{cm} = \mathsf{CM.Commit}(\mathsf{ck}, \mathsf{m})$.

Completeness. The sumcheck-friendly property tells us that $\mathsf{cm} = \sum_{\underline{\omega} \in H^\ell} f_{\mathsf{CM}}(p_{\mathsf{m}}(\underline{\omega}), p_{\mathsf{ck}}(\underline{\omega}))$, so the completeness of Protocol 3 follows from the completeness of the sumcheck protocol.

Knowledge soundness. Since m can be efficiently obtained from $p_{\mathsf{m}}(\underline{X})$, it suffices for the extractor to recover, from the tree of transcripts, a polynomial $p_{\mathsf{m}}(\underline{X})$ such that $\mathsf{cm} = \sum_{\underline{\omega} \in H^\ell} f_{\mathsf{CM}}(p_{\mathsf{m}}(\underline{\omega}), p_{\mathsf{ck}}(\underline{\omega}))$.

The proof strategy is similar to the one described in Section 2.2.1: the extractor proceeds layer by layer, starting from the leaf layer of the tree of transcripts and continuing to the root; for each node in a particular layer, the extractor computes a polynomial obtained from the polynomials associated to the node's children. The desired polynomial $p_{\mathsf{m}}(\underline{X})$ is the polynomial associated to the root of the tree.

The invertibility property facilitates progress from children to parents, and states that given enough openings for a commitment of a layer one can find an opening of a commitment of the previous layer.

Definition 2 (informal). CM *is* ***K-invertible*** *if there exists an efficient algorithm* $\mathcal{I}$ *satisfying the following. Suppose that* $\mathcal{I}$ *receives* $i \in [\ell]$*, challenge vector* $(r_1, \dots, r_{i-1}) \in \mathbb{F}^{i-1}$*, distinct challenges* $r_i^{(1)}, \dots, r_i^{(K)} \in \mathbb{F}$*, opening polynomials* $p_1, \dots, p_K \in \mathbb{M}[X_{i+1}, \dots, X_\ell]$*, and commitment polynomial* $q(X) \in \mathbb{C}[X]$ *such that*

$$\forall j \in [K]\,,\ q(r_i^{(j)}) = \sum_{\omega_{i+1}, \dots, \omega_\ell \in H} f_{\mathsf{CM}}\Big(p_j(\omega_{i+1}, \dots, \omega_\ell), p_{\mathsf{ck}}(r_1, \dots, r_{i-1}, r_i^{(j)}, \omega_{i+1}, \dots, \omega_\ell)\Big) \ . \tag{1}$$

Then $\mathcal{I}$ *outputs an opening polynomial* $p \in \mathbb{M}[X_i, \dots, X_\ell]$ *such that*

$$\sum_{\omega_i \in H} q(\omega_i) = \sum_{\omega_i, \dots, \omega_\ell \in H} f_{\mathsf{CM}}\Big(p(\omega_i, \dots, \omega_\ell), p_{\mathsf{ck}}(r_1, \dots, r_{i-1}, \omega_i, \dots, \omega_\ell)\Big) \ . \tag{2}$$

The above definition omits technicalities such as the fact that the inputs to the inverter should be restricted to be efficiently generated by an adversary (given the commitment key ck) and the fact that input and output opening polynomials should be restricted to be "admissible" (partial evaluations of p_{m} for some m).

The extractor receives a K-ary tree of accepting transcripts for $(\mathsf{ck}, \mathsf{cm})$. In more detail, for every $i \in [\ell]$ and $(r_1, \dots, r_{i-1}) \in \mathbb{F}^{i-1}$, $q_i[r_1, \dots, r_{i-1}] \in \mathbb{C}[X]$ is the polynomial corresponding to the path $(r_1, \dots, r_{i-1})$ in the transcript tree (the prover's polynomial in the i-th round of the sumcheck protocol for these challenges); moreover, for every $(r_1, \dots, r_\ell) \in \mathbb{F}^\ell$, $w[r_1, \dots, r_\ell] \in \mathbb{C}$ is the opening corresponding to the path $(r_1, \dots, r_\ell)$ in the transcript tree (sent by the prover after the sumcheck protocol for these challenges). Every transcript is accepting, so we know that for every $(r_1, \dots, r_\ell) \in \mathbb{F}^\ell$ it holds that

$$\sum_{\omega_1 \in H} q_1(\omega_1) = \mathsf{cm} \quad , \quad \left\{ \sum_{\omega \in H} q_i[r_1, \dots, r_{i-1}](\omega) = q_{i-1}[r_1, \dots, r_{i-2}](r_{i-1}) \right\}_{i \in \{2, \dots, \ell\}} \quad ,$$

$$\text{and} \quad f_{\mathsf{CM}}(w[r_1, \dots, r_\ell], p_{\mathsf{ck}}(r_1, \dots, r_\ell), 1) = q_\ell(r_\ell) \quad .$$

The extractor iterates over the whole tree, proceeding with $i = \ell, \dots, 1$. In the iteration for a path $(r_1, \dots, r_{i-1}) \in \mathcal{C}^{i-1}$ with children $\{r_i^{(j)}\}_{j \in [K]}$, the extractor uses the inverter $\mathcal{I}$ to transform polynomials $\{p[r_1, \dots, r_{i-1}, r_i^{(j)}]\}_{j \in [K]}$ in $\mathbb{M}[X_{i+1}, \dots, X_\ell]$ that satisfy Equation (1) into a new polynomial $p[r_1, \dots, r_{i-1}]$ in $\mathbb{M}[X_i, \dots, X_\ell]$ that satisfies Equation (2). The initial polynomials $\{p[r_1, \dots, r_\ell]\}_{(r_1, \dots, r_\ell) \in \mathcal{C}^\ell}$ are the constant polynomials corresponding to the opening values $\{w[r_1, \dots, r_\ell]\}_{(r_1, \dots, r_\ell) \in \mathcal{C}^\ell}$. The fact that transcripts are accepting ensures that the initial polynomials satisfy the required condition, and that each produced polynomial satisfies the invertibility condition for the prior layer.

After all these iterations the extractor has found a polynomial p in $\mathbb{M}[X_1, \dots, X_\ell]$ such that $\sum_{\omega_1 \in H} q_1(\omega_1) = \sum_{\underline{\omega} \in H^\ell} f_{\mathsf{CM}}(p(\underline{\omega}), p_{\mathsf{ck}}(\underline{\omega}))$; again by the accepting condition we know that $\sum_{\omega_1 \in H} q_1(\omega_1) = \mathsf{cm}$ so we deduce that $\mathsf{cm} = \sum_{\underline{\omega} \in H^\ell} f_{\mathsf{CM}}(p(\underline{\omega}), p_{\mathsf{ck}}(\underline{\omega}))$, and the desired polynomial is p.

Whence invertibility? Invertibility is incomparable to the commitment's binding property. For example, the Pedersen commitment scheme is unconditionally invertible (see Section 2.2.1), whereas invertibility for the scalar-product commitment scheme in Protocol 2 relies on the hardness of the discrete logarithm problem. In Section 2.5 we elaborate on how to establish invertibility for different choices of commitment schemes.

Examples. Protocol 3 captures sumcheck arguments for several commitment schemes.

– The Pedersen commitment scheme (used in Protocol 1) is sumcheck-friendly because, for the function $f_{\mathsf{CM}}(a, \mathsf{G}) := 2^{-\ell} a \cdot \mathsf{G}$, for every commitment key $\underline{\mathsf{G}} \in \mathbb{G}^n$ and message $\underline{a} \in \mathbb{F}^n$ it holds that $\mathsf{CM.Commit}(\underline{\mathsf{G}}, \underline{a}) = \sum_{\underline{\omega} \in H^\ell} f_{\mathsf{CM}}(p_{\underline{a}}(\underline{\omega}), p_{\underline{\mathsf{G}}}(\underline{\omega}))$, where $H := \{-1, 1\}$, $\ell := \log n$, and $p_{\underline{a}}(\underline{X}), p_{\underline{\mathsf{G}}}(\underline{X})$ are the multilinear polynomials induced by $\underline{a}, \underline{\mathsf{G}}$ respectively. (See Definition 2.)

– The scalar-product commitment scheme (used in Protocol 2) is sumcheck-friendly because, for the function $f_{\mathsf{CM}}((a, b), (\mathsf{G}, \mathsf{H})) := 2^{-\ell}(a \cdot \mathsf{G}, b \cdot \mathsf{H}, a \cdot b)$, for every commitment key $(\underline{G}, \underline{H}) \in \mathbb{G}^n \times \mathbb{G}^n$ and message $(\underline{a}, \underline{b}) \in \mathbb{F}^n \times \mathbb{F}^n$ it holds that $\mathsf{CM.Commit}((\underline{G}, \underline{H}), (\underline{a}, \underline{b})) = \sum_{\underline{\omega} \in H^\ell} f_{\mathsf{CM}}((p_{\underline{a}}(\underline{\omega}), p_{\underline{b}}(\underline{\omega})), (p_{\underline{G}}(\underline{\omega}), p_{\underline{H}}(\underline{\omega})))$, where $H := \{-1, 1\}$, $\ell := \log n$, and $p_{\underline{a}}(\underline{X}), p_{\underline{b}}(\underline{X}), p_{\underline{G}}(\underline{X}), p_{\underline{\mathsf{H}}_0}(\underline{X})$ are the multilinear polynomials induced by $\underline{a}, \underline{b}, \underline{G}, \underline{H}$ respectively. (See Definition 2.)

More generally, all inner-product commitments in [BMMTV19] are sumcheck-friendly; this includes pairing-based commitment schemes appearing in works such as [LMR19]. Below we describe inner-product commitments via the notion of *sum-bilinear* commitments, which is easier to work with in our setting.

Definition 4. *A commitment scheme* CM *is* **sum-bilinear** *over a finite field* $\mathbb{F}$ *if the key, message, and commitment spaces are* $\mathbb{F}$*-linear spaces and the following properties hold for all commitment keys* $\mathsf{ckL}, \mathsf{ckR} \in \mathbb{K}^n$*, and messages* $\mathsf{mL}, \mathsf{mR} \in \mathbb{M}^n$*:*

$$\begin{aligned}\mathsf{CM.Commit}\,(\mathsf{ckL}+\mathsf{ckR}, \mathsf{mL}+\mathsf{mR}) &= \mathsf{CM.Commit}\,(\mathsf{ckL}, \mathsf{mL}) + \mathsf{CM.Commit}\,(\mathsf{ckR}, \mathsf{mL}) \\ &\quad + \mathsf{CM.Commit}\,(\mathsf{ckL}, \mathsf{mR}) + \mathsf{CM.Commit}\,(\mathsf{ckR}, \mathsf{mR}) \quad \textit{and} \\ \mathsf{CM.Commit}\,(\mathsf{ckL}\|\mathsf{ckR}, \mathsf{mL}\|\mathsf{mR}) &= \mathsf{CM.Commit}\,(\mathsf{ckL}, \mathsf{mL}) + \mathsf{CM.Commit}\,(\mathsf{ckR}, \mathsf{mR}) \quad .\end{aligned}$$

Claim (informal). If CM is sum-bilinear then CM is sumcheck-friendly.

Proof sketch. The first property allows us to "lift" the commitment function to a polynomial. For the function $f_{\mathsf{CM}}(\underline{a}, \underline{\mathsf{G}}) = 2^{-\ell}\mathsf{CM.Commit}(\underline{\mathsf{G}}; \underline{a})$, it holds that for every message $\underline{a} \in \mathbb{M}^n$ and commitment key $\underline{\mathsf{G}} \in \mathbb{K}^n$

$$\begin{aligned}f_{\mathsf{CM}}(p_{\underline{a}}(\underline{X}), p_{\underline{\mathsf{G}}}(\underline{X})) &= 2^{-\ell}\mathsf{CM.Commit}\left(p_{\underline{\mathsf{G}}}(\underline{X}), p_{\underline{a}}(\underline{X})\right) \\ &= 2^{-\ell} \sum_{\underline{i},\underline{j} \in \{0,1\}^\ell} \mathsf{CM.Commit}\left(\underline{\mathsf{G}}_{\underline{j}}, \underline{a}_{\underline{i}}\right) \cdot X_1^{i_1+j_1} \cdots X_\ell^{i_\ell+j_\ell}\end{aligned}$$

where $\ell := \log n$, and $p_{\underline{a}}(\underline{X}), p_{\underline{\mathsf{G}}}(\underline{X})$ are the multilinear polynomials induced by $\underline{a}, \underline{\mathsf{G}}$ via Definition 2. The second property implies that $\sum_{\underline{\omega} \in H^\ell} f_{\mathsf{CM}}(p_{\underline{a}}(\underline{\omega}), p_{\underline{\mathsf{G}}}(\underline{\omega})) = \mathsf{CM.Commit}\,(\mathsf{ck}, \mathsf{m})$ for $H = \{-1, 1\}$. □

2.4 Extending sumcheck arguments to modules

We have so far discussed sumcheck arguments for sumcheck-friendly commitment schemes involving a prime-order group and its scalar field. Yet sumcheck arguments can be formulated more generally to capture commitments in other settings, such as groups of unknown order [BFS20] and lattices [BLNS20]. We explain the changes for this generalization, and how they affect completeness and knowledge soundness.

Modules, norms, slackness. To motivate the considerations that arise, we find it helpful to first recall the Pedersen commitment scheme in other cryptographic settings (ignoring for now randomness for hiding).

– *Pedersen over groups of unknown order.* Let $\mathbb{G}$ be a group of unknown order and let $q, p > 2$ be primes that satisfy certain conditions (determined by the type of instantiation of $\mathbb{G}$). A Pedersen commitment is computed as $\mathsf{CM.Commit}(\underline{\mathsf{G}}, \underline{a}) = \langle \underline{a}, \underline{\mathsf{G}} \rangle \in \mathbb{G}$ where the commitment key $\underline{\mathsf{G}}$ equals $(1 \cdot \mathsf{G}, q \cdot \mathsf{G}, \ldots, q^{n-1} \cdot \mathsf{G})$ for a random group element $\mathsf{G} \in \mathbb{G}$ and the message $\underline{a}$ is a vector in $\left((-\frac{p-1}{2}, \frac{p-1}{2}) \cap \mathbb{Z}\right)^n$.
– *Pedersen over lattices.* Let R be a normed ring and let B_{SIS} be a norm bound of "short" ring elements; a popular choice is $R = \mathbb{Z}_q[X]/\langle X^d + 1 \rangle$ and short ring elements in $R(B_{\mathsf{SIS}})$, i.e. elements of R with norm at most B_{SIS}, for a suitable B_{SIS}. A Pedersen commitment is computed as $\mathsf{CM.Commit}(\underline{\mathsf{G}}, \underline{a}) = \langle \underline{a}, \underline{\mathsf{G}} \rangle$ where $\underline{\mathsf{G}}$ is a matrix of random ring elements and $\underline{a}$ is a vector of short ring elements.

These examples suggest that we need to consider algebraic structures that are not necessarily rings but whose scalars are over a ring, and so we rely on the notion of modules over a ring. Moreover, we need to take into account the norms of openings. Finally, we will only be able to extract a "relaxed" opening for a given commitment, which differs from a regular opening in two ways: (i) the opening might have larger norm than an honestly committed value; (ii) the opening might not satisfy the commitment equation but only a related equation parametrized by a *slackness* c, which we model via an opening algorithm $\mathsf{CM.Open}$ that additionally takes c as input. This is similar to what happens for Schnorr protocols in these settings, as we explain in Section 2.5.

Extending the sumcheck-friendly property. We extend the definition of a sumcheck-friendly commitment scheme (Definition 3) as follows: (i) the spaces $\mathbb{M}, \mathbb{K}, \mathbb{C}$ are modules over the same ring R; (ii) the summation domain is a subset H of R; (iii) a message polynomial $p_{\mathsf{m}}(\underline{X})$ is over the module $\mathbb{M}$; (iv) a key polynomial $p_{\mathsf{ck}}(\underline{X})$ is over the module $\mathbb{K}$; (v) the combiner function f_{CM} maps $\mathbb{M} \times \mathbb{K}$ (and a slackness factor) to the module $\mathbb{C}$; (vi) the summation condition now involves an efficient predicate ϕ_{sc} and is as follows:

$$\mathsf{CM.Commit}(\mathsf{ck}, \mathsf{m}) = \textstyle\sum_{\underline{\omega} \in H^{\ell}} f_{\mathsf{CM}}(p_{\mathsf{m}}(\underline{\omega}), p_{\mathsf{ck}}(\underline{\omega}), 1) \quad \text{and for every slackness } c$$
$$\mathsf{CM.Open}(\mathsf{ck}, \mathsf{m}, \mathsf{cm}, c) = 1 \;\Leftrightarrow\; \phi_{\mathrm{sc}}\big(\mathsf{cm}, \textstyle\sum_{\underline{\omega} \in H^{\ell}} f_{\mathsf{CM}}(p_{\mathsf{m}}(\underline{\omega}), p_{\mathsf{ck}}(\underline{\omega}), c), c\big) = 1 \enspace .$$

(Thus Definition 3 is the special case where $\mathbb{M}, \mathbb{K}, \mathbb{C}$ are $\mathbb{F}$-linear, $R = \mathbb{F}$, ϕ_{sc} checks equality of cm and the sum, and there are no slackness factors.)

Extending sumcheck arguments. In the sumcheck argument for a commitment scheme that is sumcheck-friendly according to the extended definition, we must additionally ensure that: (i) we use a challenge set $\mathcal{C} \subseteq R$ for the sumcheck protocol that satisfies certain properties (discussed further below) that facilitate proving knowledge soundness; (ii) we use norm bounds for commitment openings, so the underlying ring R and the module $\mathbb{M}$ must be equipped with a norm. With these in mind, we now rewrite Protocol 3 for the more general setting (differences in blue), which will allow us to capture the different cryptographic settings.

Protocol 4: sumcheck argument for sumcheck-friendly commitments (over modules)

For $n = 2^{\ell}$, the prover and verifier receive as input commitment key ck and commitment cm. The prover also receives as input an opening m such that $\|p_{\mathsf{m}}(\underline{X})\| \leq B_{\mathsf{C}}$ and $\mathsf{cm} = \mathsf{CM.Commit}(\mathsf{ck}, \mathsf{m})$.

The prover and verifier engage in a sumcheck protocol for the instance

$$\mathbb{x}_{\mathrm{SC}} := (R, M = \mathbb{C}, H, \ell, \tau = \mathsf{cm}, \mathcal{C})$$

where the prover uses the polynomial $p_{\mathsf{m},\mathsf{ck}}(\underline{X}) := f_{\mathsf{CM}}(p_{\mathsf{m}}(\underline{X}), p_{\mathsf{ck}}(\underline{X}), 1)$. At the end of the sumcheck protocol, the prover learns $\underline{r} \in R^{\ell}$ and the verifier learns $(\underline{r}, v) \in R^{\ell} \times \mathbb{C}$. Then the prover computes and sends $w := p_{\mathsf{m}}(\underline{r})$ to the verifier. Finally the verifier computes $p_{\mathsf{ck}}(\underline{r})$, checks that $\|w\| \leq B_{\mathsf{SA}}$ (for B_{SA} discussed in the completeness property below), and checks that $f_{\mathsf{CM}}(w, p_{\mathsf{ck}}(\underline{r}), 1) = v$.

Completeness. This follows similarly as in the special case considered in Section 2.3.2, with the main difference that the norm bounds must be set so that they hold for any valid execution of the protocol. We need that for any message m (in the message space of the given commitment key ck) such that $\|p_{\mathsf{m}}(\underline{X})\| \leq B_{\mathsf{C}}$ and challenge vector $\underline{r} \in \mathcal{C}^{\ell}$ it holds that $\|p_{\mathsf{m}}(\underline{r})\| \leq B_{\mathsf{SA}}$. An explicit expression for B_{SA} can be computed in a straightforward way from the maximum norm of a challenge in $\mathcal{C}$, the number of variables ℓ of $p_{\mathsf{m}}(\underline{X})$, the degree of $p_{\mathsf{m}}(\underline{X})$, and B_{C} (a bound on the maximum norm of a coefficient in $p_{\mathsf{m}}(\underline{X})$).

Knowledge soundness. We wish to prove that Protocol 4 is an opening protocol for CM: given a tree of accepting transcripts for the commitment key ck and commitment cm, we can extract a corresponding (relaxed) opening m. Similarly to Section 2.3.2, we argue knowledge soundness based on an invertibility property that generalizes the prior one (Definition 2); the challenge set $\mathcal{C}$ is now part of the property.

Definition 3 (informal). *CM is (K, N, ξ)-**invertible** if there exists an efficient algorithm $\mathcal{I}$ satisfying the following. Suppose that $\mathcal{I}$ receives $i \in [\ell]$, challenge vector $(r_1, \dots, r_{i-1}) \in \mathcal{C}^{i-1}$, distinct challenges $r_i^{(1)}, \dots, r_i^{(K)} \in \mathcal{C}$, opening polynomials $p_1, \dots, p_K \in \mathbb{M}[X_{i+1}, \dots, X_\ell]$, commitment polynomial $q(X) \in \mathbb{C}[X]$, and slackness c such that $\forall\, j \in [K]$,*

$$\phi_{\mathrm{sc}}\left(q(r_i^{(j)}),\ \sum_{\omega_{i+1},\dots,\omega_\ell \in H} f_{\mathsf{CM}}\Big(p_j(\omega_{i+1}, \dots, \omega_\ell), p_{\mathsf{ck}}(r_1, \dots, r_{i-1}, r_i^{(j)}, \omega_{i+1}, \dots, \omega_\ell), c\Big)\right) = 1 \ .$$

Then $\mathcal{I}$ outputs an opening polynomial $p \in \mathbb{M}[X_i, \dots, X_\ell]$ of norm at most $N \cdot \max_{j \in [K]} \|p_j\|$ such that

$$\phi_{\mathrm{sc}}\left(\sum_{\omega_i \in H} q(\omega_i),\ \sum_{\omega_i,\dots,\omega_\ell \in H} f_{\mathsf{CM}}\Big(p(\omega_i, \dots, \omega_\ell), p_{\mathsf{ck}}(r_1, \dots, r_{i-1}, \omega_i, \dots, \omega_\ell), \xi \cdot c\Big)\right) = 1 \ .$$

Theorem 4. *If the sumcheck-friendly commitment scheme CM is (K, N, ξ)-invertible then Protocol 4 is an opening protocol for CM: there exists an extractor that given a commitment key ck, commitment cm for a message with norm bound B_{C}, and a K-ary tree of accepting transcripts for $(\mathsf{ck}, \mathsf{cm})$, finds an opening m with norm $\|p_{\mathsf{m}}(\underline{X})\| \leq N^{\ell} B_{\mathsf{SA}}$ such that $\mathsf{CM.Open}\left(\mathsf{ck}, \mathsf{cm}, \mathsf{m}, \xi^{\ell}\right) = 1$.*

Note that since the extractor works over a tree of depth ℓ, the final loss in norm and slackness involves ℓ factors of N and ξ respectively. Technical details for our sumcheck argument are given in the full version of the paper. The final definition of invertibility that we use (related to Definition 2) has an extra parameter B_{INV}, which is an absolute upper bound on the norm of a relaxed opening for which invertibility can hold.

The slackness loss ξ depends on the cryptographic setting, and in the settings that we consider, $\xi \neq 1$ in the lattice and in the GUO setting.

2.5 Instantiations of sumcheck-friendly commitments

Our main theorem on sumcheck arguments (Theorem 1) applies to any sumcheck-friendly commitment that is invertible. Below, we summarize how to construct such commitment schemes; details are provided in the full version of the paper.

– In Section 2.5.1 we introduce secure bilinear modules.
– In Section 2.5.2 we explain how to construct a (generalized) Pedersen commitment scheme from a secure bilinear module, and give intuition for why it is sumcheck-friendly and invertible. In the technical sections, we also discuss other commitment schemes, which capture linear functions and scalar products.
– In the full version of the paper,we outline how to instantiate secure bilinear modules in different cryptographic settings: (i) prime-order groups; (ii) bilinear groups; (iii) unknown-order groups; and (iv) lattices.

2.5.1 Secure bilinear modules

A *bilinear module* $\mathcal{M} = (R, M_{\mathrm{L}}, M_{\mathrm{R}}, M_{\mathrm{T}}, e)$ consists of a ring R, three modules $M_{\mathrm{L}}, M_{\mathrm{R}}, M_{\mathrm{T}}$ over R, and a non-degenerate bilinear map $e \colon M_{\mathrm{L}} \times M_{\mathrm{R}} \to M_{\mathrm{T}}$; moreover, R and M_{L} are equipped with norms. For notational simplicity we denote $e(\underline{a}, \underline{\mathsf{G}})$ as $\langle \underline{a}, \underline{\mathsf{G}} \rangle$ and define $M(B) := \{m \in M \text{ such that } \|m\| \leq B\}$.

A *bilinear-module generator* is a tuple $\mathsf{BM} = (\mathsf{Setup}, \mathsf{KeyGen})$ where: $\mathsf{BM.Setup}$ (given a security parameter and length parameter n) samples a bilinear module $\mathcal{M}$, integer $h \in \mathbb{N}$, and auxiliary string aux; and $\mathsf{BM.KeyGen}$ (given $\mathsf{BM.Setup}$'s output) samples a vector $\underline{\mathsf{G}} = (\underline{\mathsf{G}}_0, \underline{\mathsf{G}}_1)$ in M_{R}^{n+h}.

A bilinear-module generator BM is *secure* if it satisfies the following.

– It satisfies the *bilinear relation assumption*: for a norm bound B_{BRA} specified in aux and given $\underline{\mathsf{G}} \leftarrow \mathsf{BM.KeyGen}$, it is hard to find a non-zero $\underline{a} \in M_{\mathrm{L}}^{n+h}(B_{\mathsf{BRA}})$ such that $\langle \underline{a}, \underline{\mathsf{G}} \rangle = 0$. (This is a natural generalization of the discrete logarithm assumption, the SIS assumption, and others.)
– The integer h is hiding: there is a distribution $\mathcal{U}_{M_{\mathrm{L}}}$ on M_{L}^{h} such that, for every $\underline{a} \in M_{\mathrm{L}}^{n}$, the following two random variables are statistically close:

$$\left\{ (\underline{\mathsf{G}}, \langle \underline{a}, \underline{\mathsf{G}}_0 \rangle + \langle \underline{r}, \underline{\mathsf{G}}_1 \rangle) \middle| \begin{array}{r} \underline{\mathsf{G}} \leftarrow \mathsf{BM.KeyGen} \\ \underline{r} \leftarrow \mathcal{U}_{M_{\mathrm{L}}} \end{array} \right\} \text{ and } \left\{ (\underline{\mathsf{G}}, \langle \underline{r}, \underline{\mathsf{G}}_1 \rangle) \middle| \begin{array}{r} \underline{\mathsf{G}} \leftarrow \mathsf{BM.KeyGen} \\ \underline{r} \leftarrow \mathcal{U}_{M_{\mathrm{L}}} \end{array} \right\} .$$

– The string aux specifies a norm bound B_{C} such that $B_{\mathsf{C}} \leq B_{\mathsf{BRA}}$.
– The string aux specifies pseudoinverse parameters $(\mathcal{C}, \xi, N)$ for (R, M_{T}): for every $m, m^* \in M_{\mathrm{T}}$, $a \in R$, and distinct $c_1, c_2 \in \mathcal{C}$, if $(c_1 - c_2)m = am^*$ then there exists (and one can efficiently find) $r \in R$ such that $\xi m = rm^*$ and $\|r\| \leq N\|a\|$.

2.5.2 Sumcheck-friendly commitments over bilinear modules

We use secure bilinear-module generators to construct several sumcheck-friendly commitment schemes that are invertible: a generalized Pedersen commitment scheme , as well as commitment schemes that capture linear functions and scalar products . Below we restrict our technical overview to the Pedersen commitment scheme. Details of the other commitment schemes can be found in the full version of the paper.

Definition 4 (informal). *Let* $\mathsf{BM} = (\mathsf{Setup}, \mathsf{KeyGen})$ *be a secure bilinear-module generator and consider an output* $(\mathcal{M}, h, \mathsf{aux})$ *of* $\mathsf{BM}.\mathsf{Setup}$ *(for a message length* n*) and an output* $\underline{\mathsf{G}} = (\underline{\mathsf{G}}_0, \underline{\mathsf{G}}_1) \in M_{\mathrm{R}}^n \times M_{\mathrm{R}}^h$ *of* $\mathsf{BM}.\mathsf{KeyGen}$*. The* **(generalized) Pedersen commitment scheme** *for messages of length* n *has messages of the form* $\underline{a} \in M_{\mathrm{L}}^n(B_{\mathsf{C}})$*, and a commitment is computed as* $\mathsf{C} := \langle \underline{a}, \underline{\mathsf{G}}_0 \rangle + \langle \underline{\rho}, \underline{\mathsf{G}}_1 \rangle$*, where* ρ *is sampled appropriately from* $M_{\mathrm{L}}^h(B_{\mathsf{C}})$*. An opening with slackness* $c \in R$ *for a commitment* $\mathsf{C} \in M_{\mathrm{T}}$ *under the commitment key* $(\underline{\mathsf{G}}_0, \underline{\mathsf{G}}_1) \in M_{\mathrm{R}}^n \times M_{\mathrm{R}}^h$ *is a vector* $(\underline{a}, \underline{\rho}) \in M_{\mathrm{L}}^n(B_{\mathsf{BRA}}) \times M_{\mathrm{L}}^h(B_{\mathsf{BRA}})$ *such that* $c \cdot \mathsf{C} = \langle \underline{a}, \underline{\mathsf{G}}_0 \rangle + c \cdot \langle \underline{\rho}, \underline{\mathsf{G}}_1 \rangle$*.*

The Pedersen commitment scheme is binding under the bilinear relation assumption (which holds because BM is secure) and is hiding by the property of h (which also holds because BM is secure). Moreover, the Pedersen commitment scheme is (unconditionally) sumcheck-friendly; this can be argued in a similar way as for the usual Pedersen commitment scheme (over prime-order groups).

Establishing invertibility, however, is more challenging. Rather than specifically discussing invertibility of the Pedersen commitment, in this informal overview we describe how the fact that BM is secure enables us to (straightforwardly) obtain an extraction algorithm for the (suitably generalized) Schnorr protocol. This protocol is a simple zero-knowledge argument of knowledge for a commitment opening of a given Pedersen commitment, and the extractor is asked to produce a (possibly relaxed) opening for the commitment given two accepting transcripts sharing the same first message. The considerations that arise when establishing knowledge soundness of the (non-succinct) Schnorr protocol are loosely related to, though technically simpler than, those that arise when establishing invertibility for the Pedersen commitment scheme (which in turns leads to succinct arguments of knowledge via our sumcheck arguments).

Definition 5 (informal). *In the* **Schnorr protocol** *for the (generalized) Pedersen commitment scheme, the prover and verifier receive a key* $\underline{\mathsf{G}} = (\underline{\mathsf{G}}_0, \underline{\mathsf{G}}_1) \in M_{\mathrm{R}}^{n+h}$*, commitment* $\mathsf{C} \in M_{\mathrm{T}}$ *and norm bound* B_{C}*; and the prover additionally receives as witness a message* $\underline{a} \in M_{\mathrm{L}}^n(B_{\mathsf{C}})$ *and randomness* $\underline{\rho} \in M_{\mathrm{L}}^h(B_{\mathsf{C}})$ *such that* $\langle \underline{a}, \underline{\mathsf{G}}_0 \rangle + \langle \underline{\rho}, \underline{\mathsf{G}}_1 \rangle = \mathsf{C}$*. The prover and verifier interact as follows:*

- *the prover samples* $\underline{b} \in M_{\mathrm{L}}^{n+h}(\kappa \|\mathcal{C}\| B_{\mathsf{C}})$*, where* $\|\mathcal{C}\| := \max_{r \in \mathcal{C}} \|r\|$*, and sends* $t := \langle \underline{b}, \underline{\mathsf{G}} \rangle \in M_{\mathrm{T}}$*;*
- *the verifier sends a random challenge* $r \in \mathcal{C}$*;*
- *the prover sends the response* $\underline{s} := r \cdot (\underline{a}, \underline{\rho}) + \underline{b} \in M_{\mathrm{L}}^{n+h}$ *if* $\|\underline{s}\| \leq (\kappa - 1)\|\mathcal{C}\| B_{\mathsf{C}}$ *(otherwise aborts);*
- *the verifier accepts if* $\langle \underline{s}, \underline{\mathsf{G}} \rangle = r \cdot \mathsf{C} + t$ *and* $\|\underline{s}\| \leq (\kappa - 1)\|\mathcal{C}\| B_{\mathsf{C}}$*.*

The parameter κ is chosen such that $\underline{b}$ "masks" $(\underline{a}, \underline{\rho})$. We discuss how to choose κ in Section 2.6, where similar considerations arise in other protocols; here, instead, we focus on discussing knowledge extraction. The extractor recovers an opening of C from two accepting transcripts $(t, r_1, \underline{s}_1)$ and $(t, r_2, \underline{s}_2)$ sharing the same first message t but with distinct challenges r_1 and r_2. First, subtracting the verification equation for one transcript from that of the other transcript shows that $\langle \underline{s}_1 - \underline{s}_2, \underline{\mathsf{G}} \rangle = (r_1 - r_2) \cdot \mathsf{C}$. The fact that BM is secure implies that $(\mathcal{C}, \xi, N)$ are pseudoinverse parameters for (R, M_{T}), so we can compute an $r \in R$ such that $\xi \cdot \mathsf{C} = r \langle \underline{s}_1 - \underline{s}_2, \underline{\mathsf{G}} \rangle$ with $\|r\| \leq N$. Therefore, the extractor has found a relaxed opening $(\underline{a}', \underline{\rho}') := r(\underline{s}_1 - \underline{s}_2)$ such that

$\langle \underline{a}', \underline{\mathsf{G}}_0 \rangle + \langle \underline{\rho}', \underline{\mathsf{G}}_1 \rangle = \xi \cdot \mathsf{C}$ with $\|(\underline{a}', \underline{\rho}')\| \leq N2(\kappa - 1)\|\mathcal{C}\| B_{\mathsf{C}}$. (And we see that the norm B_{C} must satisfy $N2(\kappa - 1)\|\mathcal{C}\| B_{\mathsf{C}} \leq B_{\mathsf{BRA}}$.)

The norm computations above ignore *expansion factors* that appear when computing the norms of expressions that involve the multiplication of ring and module elements (see the full version of the paper for details).

2.6 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. This involves choosing a specific sumcheck-friendly 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 2.7 we explain how to build on this to prove Theorem 2.

We first introduce the notion of *protocol-friendly* bilinear-module generator. A bilinear-module generator BM is *protocol-friendly* if it satisfies the following.

– BM is secure (see Section 2.5.1).
– M_{L} is not merely an R-module but also a ring itself (so that scalar products over M_{L} are defined).[6]
– The string aux specifies $\kappa \in \mathbb{N}$ such that BM is *masking-friendly* (i.e., for every $B \in \mathbb{N}$ with $B_{\mathsf{C}} \leq B \leq B_{\mathsf{BRA}}/\kappa$ and $\underline{a} \in M_{\mathrm{L}}^n(B)$, $\{\underline{a} + \underline{b}\}_{\underline{b} \leftarrow M_{\mathrm{L}}^n(\kappa B)}$ is close to uniform).
– The string aux specifies an ideal I such that multiplication by ξ (which is part of the pseudoinverse parameters $(\mathcal{C}, \xi, N)$ also in aux) is invertible modulo I.

The instantiations of bilinear-module generators given in the full version of this paper are also protocol-friendly.

– *Prime-order groups:* $\mathsf{BM.Setup}$ additionally outputs $\kappa := \infty$ and $I := \{0\}$. This means that the argument supports scalar products over $M_{\mathrm{L}}/I = \mathbb{F}_q$, the scalar field of a prime-order group $\mathbb{G}$.
– *Bilinear groups:* $\mathsf{BM.Setup}$ additionally outputs $\kappa := \infty$ and $I := \{0\}$. This means that the argument supports scalar products over $M_{\mathrm{L}}/I = \mathbb{G}_1$ (alternatively, $\mathbb{G}_2$), a source group in the bilinear group.
– *GUO setting:* $\mathsf{BM.Setup}$ additionally outputs $\kappa := O(2^{\lambda})$ and $I := n\mathbb{Z}$ for $n \in \mathbb{Z}$ whose prime factors are greater than or equal to p. This means that the argument supports scalar products over $M_{\mathrm{L}}/I = \mathbb{Z}/n\mathbb{Z}$ for any n satisfying these conditions.
– *Lattice setting:* $\mathsf{BM.Setup}$ additionally outputs $\kappa := O(dn)$ and $I := n\mathbb{Z}$ for odd $n \neq -1, 1$. This means that the argument supports scalar products over $M_{\mathrm{L}}/I = \mathbb{Z}/n\mathbb{Z}$ for any n satisfying these conditions.

The commitment scheme that we consider has two-part messages and includes a commitment to their scalar-product; it is the extension of the scalar-product commitment from Section 2.3.1 to bilinear modules.

[6] In the pairing setting where M_{L} is not a ring, we define scalar-product commitments differently. See the full version for details.

Definition 6 (informal). *Let* $\mathsf{BM} = (\mathsf{Setup}, \mathsf{KeyGen})$ *be a protocol-friendly bilinear-module generator. The* **(generalized) scalar-product commitment scheme** *for messages of length* n *has messages of the form* $(\underline{a}, \underline{b}) \in M_{\mathrm{L}}^n \times M_{\mathrm{L}}^n$ *such that* $\|\underline{a}\|, \|\underline{b}\| \leq B_{\mathsf{C}}$*, and commitment keys of the form* $(\underline{\mathsf{G}}_0, \underline{\mathsf{G}}_1, \underline{\mathsf{H}}_0, \underline{\mathsf{H}}_1, \mathsf{U}_0, \underline{\mathsf{U}}_1) \in M_{\mathrm{R}}^{n+h} \times M_{\mathrm{R}}^{n+h} \times M_{\mathrm{R}}^{1+h}$*. A commitment is computed by sampling* $\underline{\rho}_a, \underline{\rho}_b, \underline{\rho}_t \in M_{\mathrm{L}}^h(B_{\mathsf{C}})$ *and computing*

$$\Big(\langle \underline{a}, \underline{\mathsf{G}}_0 \rangle + \langle \underline{\rho}_a, \underline{\mathsf{G}}_1 \rangle,\ \langle \underline{b}, \underline{\mathsf{H}}_0 \rangle + \langle \underline{\rho}_b, \underline{\mathsf{H}}_1 \rangle,\ \langle \underline{a}, \underline{b} \rangle \cdot \mathsf{U}_0 + \langle \underline{\rho}_t, \underline{\mathsf{U}}_1 \rangle\Big)$$

In other words, a commitment is the tuple consisting of three generalized Pedersen commitments: for the first part of the message $\underline{a}$*, for the second part of the message* $\underline{b}$*, and for their scalar product* $\langle \underline{a}, \underline{b} \rangle \in M_{\mathrm{L}}$*.*

A valid opening for a commitment $(\mathsf{C}_a, \mathsf{C}_b, \mathsf{C}_t) \in M_{\mathrm{T}}^3$ *with keys* $(\underline{\mathsf{G}}_0, \underline{\mathsf{G}}_1, \underline{\mathsf{H}}_0, \underline{\mathsf{H}}_1 \mathsf{U}_0, \underline{\mathsf{U}}_1) \in M_{\mathrm{R}}^{n+h} \times M_{\mathrm{R}}^{n+h} \times M_{\mathrm{R}}^{1+h}$ *and slackness* $c \in R$ *is a vector* $(\underline{a}, \underline{b}, \underline{\rho}_a, \underline{\rho}_b, \underline{\rho}_t) \in M_{\mathrm{L}}^n(B_{\mathsf{BRA}}) \times M_{\mathrm{L}}^n(B_{\mathsf{BRA}}) \times M_{\mathrm{L}}^{3h}(B_{\mathsf{BRA}})$ *such that*

$$c^2 \cdot \mathsf{C} = \big(c \cdot \langle \underline{a}, \underline{\mathsf{G}}_0 \rangle + c^2 \cdot \langle \underline{\rho}_a, \underline{\mathsf{G}}_1 \rangle,\ c \cdot \langle \underline{b}, \underline{\mathsf{H}}_0 \rangle + c^2 \cdot \langle \underline{\rho}_b, \underline{\mathsf{H}}_1 \rangle,\ \langle \underline{a}, \underline{b} \rangle \cdot \mathsf{U}_0 + c^2 \langle \underline{\rho}_t, \underline{\mathsf{U}}_1 \rangle\big) \quad .$$

The generalized scalar-product commitment scheme is binding under the bilinear relation assumption. Moreover, it is sumcheck-friendly (unconditionally). The proof of invertibility follows from algebraic manipulations analogous to the case of generalized Pedersen commitments discussed in Section 2.5; though note that establishing invertibility in this case requires computational assumptions (even in the discrete logarithm setting as discussed in Section 2.3.1).

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 $\mathcal{R}_{\mathsf{CMSP}}$.

Definition 7 (informal). *The committed scalar-product relation* $\mathcal{R}_{\mathsf{CMSP}}(c, B_{\mathsf{C}})$ *are the pairs* $(\mathbb{x}, \mathbb{w})$ *where:*

- *The instance* $\mathbb{x}$ *contains*
  - *a protocol-friendly bilinear-module generator* BM*;*
  - *commitment keys* $(\underline{\mathsf{G}}_0, \underline{\mathsf{G}}_1, \underline{\mathsf{H}}_0, \underline{\mathsf{H}}_1, \mathsf{U}_0, \underline{\mathsf{U}}_1) \in M_{\mathrm{R}}^n \times M_{\mathrm{R}}^h \times M_{\mathrm{R}}^n \times M_{\mathrm{R}}^h \times M_{\mathrm{R}} \times M_{\mathrm{R}}^h$*;*
  - *commitments* $\mathsf{C}_a, \mathsf{C}_b, \mathsf{C}_t \in M_{\mathrm{T}}$*.*
- *The witness* $\mathbb{w} = (\underline{a}, \underline{\rho}_a, \underline{b}, \underline{\rho}_b, t, \underline{\rho}_t) \in M_{\mathrm{L}}^{2n+1+3h}$ *is such that* $\|\underline{a}\|, \|\underline{\rho}_a\|, \|\underline{b}\|, \|\underline{\rho}_b\|, \|t\|, \|\underline{\rho}\| \leq B_{\mathsf{C}}$ *and*
  - $(\underline{a}, \underline{\rho}_a)$ *is a valid opening of the Pedersen commitments* C_a *with slackness* c*;*
  - $(\underline{b}, \underline{\rho}_b)$ *is a valid opening of the Pedersen commitments* C_b *with slackness* c*;*
  - $(t, \underline{\rho}_t)$ *is a valid opening of the Pedersen commitment* C_t *with slackness* c^2 *and* $t = \langle \underline{a}, \underline{b} \rangle \bmod I$*.*

The relation reasons about scalar-product relations over the quotient ring $R_\bullet = M_{\mathrm{L}}/I$ (M_{L} modulo I) for the ideal $I \subseteq M_{\mathrm{L}}$ specified in aux. In certain settings, such as the lattice and GUO setting, we only extract openings to commitments with slackness $c \neq 1$, we 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 $R_\bullet$. We now summarize the scalar-product argument; details can be found in the full version.

The prover begins by computing a commitment $\mathsf{C} \in M_{\mathrm{T}}$ to $\langle \underline{a}, \underline{b} \rangle \in M_{\mathrm{L}}$, which may not be equal to $t \in M_{\mathrm{L}}$. Then the prover and verifier engage in the these sub-protocols: (i) an interactive reduction masking the three Pedersen commitments to $\underline{a}, \underline{b}, t$, converting them into a single scalar-product commitment; (ii) a sumcheck argument to prove knowledge of an opening to the scalar-product commitment; and (iii) a consistency check that the committed values $\langle \underline{a}, \underline{b} \rangle$ and t equal modulo I.

Reduction to a sumcheck argument. The prover samples masking values $\underline{y}_a$ and $\underline{y}_b$ to rerandomize the commitments to $\underline{a}, \underline{b}, \langle \underline{a}, \underline{b} \rangle$: the prover sends commitments to $\underline{y}_a$ and $\underline{y}_b$, and also to $v_1 := \langle \underline{a}, \underline{y}_b \rangle + \langle \underline{b}, \underline{y}_a \rangle$ and $v_0 := \langle \underline{y}_a, \underline{y}_b \rangle$ (which depend only on $\underline{a}, \underline{b}, \underline{y}_a, \underline{y}_b$). Then the verifier sends to the prover a random challenge $\alpha \in \mathcal{C}$. Then the prover computes $\underline{e}_a := \alpha \underline{a} + \underline{y}_a$, $\underline{e}_b := \alpha \underline{b} + \underline{y}_b$, and $\langle \underline{e}_a, \underline{e}_b \rangle = \alpha^2 \langle \underline{a}, \underline{b} \rangle + \alpha v_1 + v_0$. The openings of the rerandomized commitments do not leak any information about $\underline{a}$ or $\underline{b}$, and so the prover can safely send the corresponding commitment randomness to the verifier. Finally, the prover and verifier engage in a sumcheck argument on the scalar-product commitment consisting of the commitments to $\underline{e}_a$, $\underline{e}_b$, and $\langle \underline{e}_a, \underline{e}_b \rangle$. Since the sumcheck argument is invoked on inputs that have been masked, zero knowledge is ensured (i.e., no information about the witness $\mathbb{w} = (\underline{a}, \underline{\rho}_a, \underline{b}, \underline{\rho}_b, t, \underline{\rho}_t)$ is revealed) even though the sumcheck argument itself is not zero knowledge.

Checking consistency modulo I. The sumcheck argument merely convinces the verifier that the prover knows a witness for the scalar-product commitment $(\mathsf{C}_a, \mathsf{C}_b, \mathsf{C})$, while the verifier additionally wants to know that the openings of C and C_t are equal modulo I. For this, we rely on a protocol on the commitments to $\langle \underline{a}, \underline{b} \rangle$ and t to check that they are equivalent modulo I. First, before receiving the verifier's challenge α, the prover samples a masking value ζ, and sends to the verifier its Pedersen commitment C_ζ and its reduction $\zeta \bmod I$ (in the clear); after receiving α the prover sends to the verifier the value $\bar{v} := \alpha \cdot (\langle \underline{a}, \underline{b} \rangle - t) + \zeta$. The verifier then checks that $\bar{v} = \zeta \bmod I$, and that $\bar{v}$ is a valid opening for the commitment to $\alpha \cdot (\langle \underline{a}, \underline{b} \rangle - t) + \zeta$ (for appropriate commitment randomness). Intuitively, if $\bar{v} = \alpha \cdot (\langle \underline{a}, \underline{b} \rangle - t) + \zeta$ for two distinct values of α, then one can solve linear equations to deduce that $\xi \cdot (\langle \underline{a}, \underline{b} \rangle - t) = 0 \bmod I$. Then, since multiplication by the constant ξ from the pseudoinverse parameters $(\mathcal{C}, \xi, N)$ is invertible modulo I (this is required by the protocol-friendly property), we conclude that $\langle \underline{a}, \underline{b} \rangle = t \bmod I$.

2.7 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 $R_\bullet$ asks: given coefficient matrices $A, B, C \in R_\bullet^{n \times n}$ and an instance vector $\underline{x}$ over $R_\bullet$, is there a witness vector $\underline{w}$ over $R_\bullet$ such that $\underline{z} := (\underline{x}, \underline{w}) \in R_\bullet^n$ satisfies $A\underline{z} \circ B\underline{z} = C\underline{z}$? To a first order, Theorem 2 is proved by reducing the R1CS problem over $R_\bullet$ to several scalar-product sub-problems over $R_\bullet$, and then relying on the zero-knowledge succinct argument for scalar products in Section 2.6. This implies that we support R1CS over the rings supported in that section: $R_\bullet = M_{\mathrm{L}}/I$, where M_{L} is the left module of a protocol-friendly bilinear module, and $I \subseteq M_{\mathrm{L}}$ is an ideal. As with our scalar-product arguments, I is used to cancel out slackness factors and prove exact relations. Below we summarize the reduction from R1CS to scalar products.

The prover $\mathbf{P}$ sends commitments to the full assignment $\underline{z} \in R_\bullet^n$ and to its linear combinations $\underline{z}_A, \underline{z}_B \in R_\bullet^n$. Then $\mathbf{P}$ is left to convince the verifier $\mathbf{V}$ that the committed information satisfies these conditions:

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

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

Next, the prover $\mathbf{P}$ sends a commitment to $\underline{z}'_A := \underline{r} \circ \underline{z}_A$, and also commitments to $\alpha := \langle \underline{r}_A, \underline{z} \rangle$, $\beta := \langle \underline{r}_B, \underline{z} \rangle$, and $\gamma := \langle \underline{r}_C, \underline{z} \rangle$. Then the prover and verifier engage in scalar-product sub-protocols (described in Section 2.6) to verify these 7 scalar products (recall each party can compute $\langle \underline{s}, \underline{x} \rangle$):

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

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

All commitments in the protocol are hiding, and hence do not leak any information about the witness vector $\underline{w}$. 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.

Technical details can be found in the full version of the paper.

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, and part while employed by IBM Research – Zurich, supported by the SNSF ERC Transfer Grant CRETP2-166734 – FELICITY.

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.

[ACK21] Thomas Attema, Ronald Cramer, and Lisa Kohl. *A Compressed Σ-Protocol Theory for Lattices*. Cryptology ePrint Archive, Report 2021/307. 2021.

[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.

[AL21] Martin R. Albrecht and Russell W. F. Lai. *Subtractive Sets over Cyclotomic Rings: Limits of Schnorr-like Arguments over Lattices*. Cryptology ePrint Archive, Report 2021/202. 2021.

[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.

[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.

[BCGRS17] Eli Ben-Sasson, Alessandro Chiesa, Ariel Gabizon, Michael Riabzev, and Nicholas Spooner. "Interactive Oracle Proofs with Constant Rate and Query Complexity". In: *Proceedings of the 44th International Colloquium on Automata, Languages and Programming*. ICALP '17. 2017, 40:1–40:15.

[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.

[BHRRS21] Alexander R. Block, Justin Holmgren, Alon Rosen, Ron D. Rothblum, and Pratik Soni. "Time- and Space-Efficient Arguments from Groups of Unknown Order". In: *Proceedings of the 41st Annual International Cryptology Conference*. CRYPTO '21. 2021, ???–???

[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.

[CFIK03] Ronald Cramer, Serge Fehr, Yuval Ishai, and Eyal Kushilevitz. "Efficient Multi-party Computation over Rings". In: *Proceedings of the 22nd Annual International Conference on Theory and Application of Cryptographic Techniques*. EUROCRYPT '03. 2003, pp. 596–613.

[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.

[CMSZ21] Alessandro Chiesa, Fermi Ma, Nicholas Spooner, and Mark Zhandry. *Post-Quantum Succinct Arguments*. Cryptology ePrint Archive, Report 2021/334. 2021.

[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.

[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.

[GNS21] Chaya Ganesh, Anca Nitulescu, and Eduardo Soria-Vazquez. *Rinocchio: SNARKs for Ring Arithmetic*. Cryptology ePrint Archive, Report 2021/322. 2021.

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

[JKKZ20] Ruta Jawale, Yael Tauman Kalai, Dakshita Khurana, and Rachel Zhang. *SNARGs for Bounded Depth Computations and PPAD Hardness from Sub-Exponential LWE*. IACR Cryptology ePrint Archive, Report 2020/980. 2020.

[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.

[ZXZS20] 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. 2020, pp. 859–876.

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