Pairing-Based SNARGs with Two Group Elements

Gal Arnon
galarnon42@gmail.com
Bocconi University

Jesko Dujmovic
mail@ind-jesko.net
Northeastern University

Eylon Yogev
eylon.yogev@biu.ac.il
Bar-Ilan University

November 27, 2025

Abstract

SNARGs are cryptographic primitives that allow a prover to demonstrate membership in an NP language while sending a proof that is much smaller than the witness. In this work, we focus on the succinctness of publicly-verifiable group-based SNARGs, analyzed in a model that combines both a generic bilinear group ($\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$) and a random oracle (the GGM + ROM).

We construct the first publicly-verifiable SNARG in the GGM + ROM where the proof consists of exactly 2 elements of $\mathbb{G}_1$ and no additional bits, achieving the smallest proof size among all known publicly verifiable group-based SNARGs. Our security analysis is tight, ensuring that the construction incurs no hidden security losses. Concretely, when instantiated with the BLS12-381 curve for 128-bit security, our scheme yields a proof size of 768 bits, nearly a $2\times$ improvement over the best known pairing-based SNARG. While our scheme is not yet concretely efficient, it demonstrates the feasibility of ultra-short proofs and opens the door to future practical instantiations.

Complementing this construction, we establish a new lower bound for group-based SNARGs. We prove that under mild and natural restrictions on the verifier (which are satisfied by all known schemes) no SNARG exists in the Maurer GGM + ROM with a proof that consists of a single group element (assuming one-way functions). This substantially strengthens the lower bound of Groth, which was more restrictive and did not extend to settings with a random oracle.

Keywords: succinct arguments; interactive proofs; generic group model; pairing-based cryptography

Contents

1 Introduction **1**

1.1 Our results 2

1.2 Related work 4

1.3 Open problems 4

2 Techniques **6**

2.1 SNARGs with two group elements 6

2.2 Lower bound for SNARGs with one group element 11

3 Preliminaries **19**

3.1 Idealized models 19

3.2 Succinct non-interactive arguments 20

4 Linear map commitments **21**

5 SNARGs with two group elements **26**

5.1 Linear PCPs 26

5.2 SNARGs from PCPs with small accepting set 29

5.3 Proof of Theorems 5.1 and 5.2 35

6 Lower bound for SNARGs with one group element **37**

6.1 Removing the ROM from one group element SNARGs 38

6.2 Ruling out one group element SNARGs in the pure GGM 47

Acknowledgments **53**

References **53**

1 Introduction

Succinct non-interactive arguments (of knowledge), known as SNARGs (or SNARKs) are cryptographic primitives that enable a prover to demonstrate membership in an NP language while sending only a small proof, much smaller than the witness itself. Their succinctness and efficiency have made SNARGs a central tool in applications such as verifiable delegation of computation, zero-knowledge proofs for privacy-preserving protocols, and scalability solutions in blockchain systems. Recent years have witnessed substantial progress in the construction of SNARGs, leading to proof systems with concretely short arguments and improved efficiency of both prover and verifier, which are widely used and deployed in practice.

The focus of this work is the succinctness of publicly verifiable group-based SNARGs. This class also captures pairing-based constructions and schemes that additionally rely on the random oracle model. Group-based SNARGs are among the most succinct and practically efficient proof systems known, and they are the ones most widely deployed in real-world applications. Our work is motivated by the following fundamental question:

What is the smallest possible (publicly verifiable) group-based SNARG proof size?

The quest for minimal proofs. A long sequence of works, beginning with Groth's construction [Gro10], has aimed to minimize the concrete proof size of SNARGs for NP. Most of these works are built over pairing-friendly bilinear groups ($\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$), with soundness typically analyzed in the generic group model [Sho97; Mau05; Nec94]. Groth's initial construction achieved proofs consisting of 42 bilinear group elements; this was later reduced to 39 elements by Lipmaa [Lip12]. Subsequently, Gennaro et al. [GGPR13] constructed SNARGs with proofs of just 7 group elements.

Bitansky et al. [BCIOP13] introduced the notion of a *linear* PCP, where a verifier issues a small number of inner-product queries to a proof vector, and described a general compiler from linear PCPs to SNARGs using the notion of linear-only encryption. A similar compiler was implicit in [GGPR13], and both works followed the high-level blueprint introduced in [IKO05; Gro10]. Later, Danezis et al. [DFGK14] constructed succinct arguments with proofs consisting of just 4 bilinear group elements.

Finally, Groth [Gro16] presented a construction with only 3 bilinear group elements (amounting to just over 1000 bits in practice [SCI14]) and with highly efficient verification. The succinctness and efficiency of this last scheme made it particularly appealing in practice, and it has since been widely implemented and deployed. For a long time, it was not clear if one can construct bilinear-based (publicly verifiable) SNARGs with fewer than 3 group elements.

Smaller proofs in the GGM + ROM. Recently, two improvements have been made over the construction of Groth [Gro16] by additionally leveraging a random oracle beyond the generic group (which we will refer to as the GGM + ROM). Lipmaa leveraged this model [Lip24] to reduce the proof size by designing a SNARG consisting of only three $\mathbb{G}_1$ group elements (the typically smaller group in a bilinear map) and one field element. This optimization roughly halves the proof length compared to [Gro16] (for some parameter settings).

Subsequently, Dellepere, et al. [DMS24] introduced the "PARI" scheme, a SNARG that breaks the three-group-element barrier (also in the GGM+ROM). PARI has a proof size of just two $\mathbb{G}_1$ elements and two field elements, representing the smallest proof size achieved by any known publicly verifiable SNARG.

Groth [Gro16] showed that there are no group-based SNARGs for NP with proofs consisting of a single group element, as long as the verifier admits a certain structure. However, the lower bound does not extend to the ROM model, a gap explicitly highlighted by Lipmaa [Lip24]. At present, no lower bounds are known in this combined model for group-based SNARGs. It is also worth noting that allowing a single round of interaction enables proofs of just one group element [BIOW20].

Designated verifiers. One approach to reducing proof size is to consider the relaxed *designated-verifier* SNARG (dv-SNARG) model. In this setting, the verifier privately holds a verification key generated during setup, and soundness is lost if this key is ever leaked. Such schemes are considerably less applicable in practice compared to their publicly verifiable counterparts.

Barta et al. [BIOW20] constructed a dv-SNARG with proofs of just two group elements in a pairing-free setting. However, their construction achieves only inverse-polynomial (rather than negligible) soundness error. More recently, Arnon et al. [ADI25] presented a dv-SNARG consisting of a single group element plus $O(\lambda)$ bits for λ-bit security, achieving negligible soundness error. However, their scheme is non-reusable: it can be used only once, after which the setup must be repeated.

Non-group-based SNARGs. There are also several constructions of SNARGs that do not rely on bilinear groups (or on groups at all). Perhaps the most prominent examples are hash-based SNARGs, whose security is proven in the random oracle model without additional assumptions. These schemes enjoy many desirable practical properties (and are post-quantum secure), but their proofs are comparatively large. Moreover, there is strong evidence that such schemes cannot achieve proofs significantly smaller than $\Theta(\lambda^2)$ for λ-bit security [HNY22].

If one is willing to rely on stronger cryptographic assumptions, such as general-purpose obfuscation, then it is possible to construct SNARGs with optimal proof length [SW14; WW24; WZ24; WW25]. However, today these schemes remain far from practical and, moreover, they satisfy the stronger notion of knowledge soundness only in elaborate ideal models (e.g., [JLLW23]). Finally, lattice-based proof systems, e.g., [BISW17; BS23; SSEKYZ24; AFLN24], are not currently competitive with group-based systems in terms of concrete succinctness.

1.1 Our results

We make significant progress towards pinpointing the minimal proof size in group-based SNARGs in terms of both new constructions and improved lower bounds.

The 2-element upper bound. We construct a pairing-based, publicly verifiable SNARG with proofs consisting of exactly *two* $\mathbb{G}_1$ *elements*, achieving the smallest proof size among all known publicly verifiable SNARGs and answering an open question raised in [DMS24]. Our analysis provides tight security guarantees in the combined generic group and random oracle models, ensuring that the construction incurs no hidden security losses. We emphasize that our focus is not on optimizing prover efficiency, and the scheme is therefore not intended to compete with concretely efficient constructions. Nevertheless, our results demonstrate the feasibility of achieving publicly verifiable SNARGs with proofs of just two group elements, opening the door to future practical instantiations.

Theorem 1.1 ($2\mathbb{G}_1$ SNARG; informal)**.** *Let λ be a security parameter, and assume that $\lambda \leq \log^c n$ for some constant c, where n is the instance size. Let $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a generic bilinear group (of prime order $p \geq 2^\lambda$). Every language in NP has a (publicly verifiable) SNARG with the following properties:*

- *Proof size: 2 $\mathbb{G}_1$ elements;*

- *Soundness and knowledge error:* $\mathrm{poly}(n) \cdot s^2 \cdot 2^{-\lambda}$*, against* s*-sized adversaries (in the GGM + ROM);*
- *Completeness error:* $\mathrm{poly}(n, \lambda) \cdot 2^{-\lambda}$.

See Theorem 5.1 for a formal statement.

Instantiating the group with the widely used BLS12-381 curve (for 128-bit security) yields a proof size of $2 \cdot 384 = 768$ bits—nearly a $2\times$ improvement over the best known pairing-based SNARGs. This proof size also closely matches that of the best designated-verifier SNARG [ADI25]. We provide a detailed comparison of our proof size against existing publicly verifiable group-based SNARGs in Section 1.1.

Scheme	**Model**	**Proof Structure**	**Size (Bits)**
Groth16 [Gro16]	GGM	$2\,\mathbb{G}_1 + 1\,\mathbb{G}_2$	1536
Lipmaa [Lip24]	GGM + ROM	$3\,\mathbb{G}_1 + 1\,\mathbb{F}$	≈ 1408
PARI [DMS24]	GGM + ROM	$2\,\mathbb{G}_1 + 2\,\mathbb{F}$	≈ 1280
This Work	**GGM + ROM**	$\mathbf{2\,\mathbb{G}_1}$	**768**

Table 1: Comparison of proof sizes for publicly verifiable group-based SNARGs. Bit sizes are estimated for the BLS12-381 curve ($\approx$ 381-bit field), assuming point compression ($|\mathbb{G}_1| \approx 384$ bits, $|\mathbb{G}_2| \approx 768$ bits).

The theorem above (Theorem 1.1) guarantees λ bits of security for the reasonable parameter regime where $\lambda \leq \log^c n$. For instance, in a typical setting with $\lambda = 100$ and $n = 2^{20}$, the assumption already holds with $c = 2$. As a more fundamental question, we also examine the general parameter regime.

We obtain a two-group-element SNARG for NP (similarly to Theorem 1.1) that has negligible soundness error also for the case of $\lambda = \omega(\mathrm{polylog}\, n)$, but the construction additionally relies on the *unique games conjecture* (UGC). The UGC, introduced by Khot [Kho02], posits that it is computationally hard to approximate certain constraint satisfaction problems known as *unique games*. This conjecture has become a central assumption in computational complexity and hardness-of-approximation results.

Theorem 1.2 ($2\mathbb{G}_1$ UGC based SNARG; informal)**.** *Let* λ *be a security parameter, and assume the unique games conjecture holds. Let* $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ *be a generic bilinear group (of prime order* $p \geq 2^{\lambda}$*). Every language in NP has a (publicly verifiable) SNARG with the following properties:*

- *Proof size: 2* $\mathbb{G}_1$ *elements;*
- *Soundness and knowledge error:* $\mathrm{poly}(n) \cdot s^2 \cdot 2^{-\lambda}$*, against* s*-sized adversaries (in the GGM+ROM);*
- *Completeness error:* $\mathrm{poly}(n, \lambda) \cdot 2^{-\lambda}$.

See Theorem 5.2 for a formal statement.

The 1-element lower bound. As we have mentioned, there are no known group-based SNARGs with a single group element. In [Gro16], Groth gave strong limitations on the feasibility of pairing-based SNARGs that consist of only a single group element. He ruled out a class of SNARGs, where the verifier is a conjunction of pairing equation checks.

Complementing our construction, we give strong lower bounds for group-based SNARGs with single-element proofs. First, we strengthen Groth's lower bound by extending it to all SNARGs in the Maurer [Mau05] generic group model (GGM). Next, we extend this bound to the setting where the prover and verifier have access to both the generic group and a random oracle (GGM+ROM). The domain and range of the oracle are flexible: they may consist of group elements, pure strings, or a mixture of both. Our lower bound applies under the restriction that the verifier's random-oracle queries are the output of predefined pairing functions of the CRS and the proof (while the verifier may still issue arbitrary GGM queries and is otherwise unrestricted). Notably, all known SNARG constructions satisfy this restriction.

Theorem 1.3 (No $1\mathbb{G}$ SNARG in the GGM+ROM; informal)**.** *If one-way functions exist (in the plain model), there are no SNARGs for* NP *in the (Maurer) Bilinear GGM with a random oracle, with the following properties:*

- *The proof consists of a single group element;*
- *The completeness and soundness errors are negligible;*
- *The verifier performs queries to the ROM that are non-adaptive*[1]*;*
- *The setup algorithm does not perform random oracle queries.*

See Theorem 6.1 for a formal statement.

1.2 Related work

Pairing-free. There has been a line of work [BCIOP13; BCCGP16; BBBPWM18; BIOW20; BHIRW24] which constructs SNARGs using generic pairing-free groups, namely in the standard GGM. The simpler structure gives hope for more conservative group instantiations with better concrete parameters. Unlike pairing-based SNARGs, the most succinct GGM-based SNARGs apply only in the designated-verifier setting.

Pure random oracle model. There is a long line of work constructing SNARGs that rely solely on a random oracle (a.k.a., hash-based SNARGs). The best trait of these constructions is a fast prover, since the prover performs only "symmetric key cryptography" operations (oracle calls, or field operations), whereas group-based operations are typically heavier. At a technical level, these SNARGs combine the blueprint of Kilian [Kil92] and Micali [Mic00] with an interactive variant of classical PCPs known as an IOP [BCS16]. See [CY24] for further details. However, even the most succinct hash-based SNARGs [AHIV17; BBHR19; STW23; GLSTW23; ZCF24; BBHR18; ACY23; ACFY24; ACFY25] have proofs with tens to hundreds of kilobytes in size. Moreover, there is strong evidence that such schemes cannot achieve proofs significantly smaller than $\Theta(\lambda^2)$ for λ-bit security [HNY22].

1.3 Open problems

While our work establishes tight bounds for publicly verifiable SNARGs in the combined GGM+ROM, several intriguing problems remain open.

[1]More precisely, the verifier can be split into two parts; The first part does not do any pairing equation checks or ROM calls but it produces the call to the ROM, the second part receives the responses to the ROM call but can not do any further ROM calls.

- **Removing the Random Oracle.** Our construction relies on the random oracle to compress the proof down to two elements. In the standard Generic Group Model (without a random oracle), the state-of-the-art remains the 3-element construction of Groth [Gro16]. *Do publicly verifiable SNARGs with proofs of exactly two group elements exist in the standard GGM?*

- **Perfect Completeness.** Our scheme achieves negligible completeness error rather than perfect completeness. This limitation stems from our use of single-query linear PCPs, which inherently has imperfect completeness (e.g., [BHIRW24, Appendix A] and [BIOW20, Remark 4.9]). While acceptable in theory, perfect completeness is a desirable property in practice. *Can one construct a group-based SNARG in the GGM+ROM with two group elements and perfect completeness?*

- **Single-Element Designated Verifiers.** Our lower bound rules out single-element proofs for *publicly verifiable* schemes. However, in the designated-verifier setting, Arnon et al. [ADI25] recently constructed a SNARG consisting of a single group element plus a few auxiliary bits. *Do designated-verifier SNARGs with a proof consisting of exactly one group element exist in the GGM+ROM?*

- **Unrestricted Lower Bounds.** Our impossibility result assumes a natural structural restriction on the verifier (specifically, that ROM queries are non-adaptive). While this captures all known group-based SNARGs, a fully general lower bound would be theoretically satisfying. *Can we prove a lower bound for single-element SNARGs in the GGM+ROM without any structural assumptions on the verifier?*

2 Techniques

We give a high-level overview of our techniques:
- In Section 2.1, we give an overview of our constructions (namely, Theorem 1.1 and Theorem 1.2).
- In Section 2.2, we give an overview of our lower bound (namely, Theorem 1.3).

2.1 SNARGs with two group elements

In this section we sketch the proofs of Theorems 1.1 and 1.2, describing the construction of a SNARG in the bilinear generic group plus random oracle model with argument size of two group elements.

2.1.1 SNARGs from linear PCPs

The starting point of our construction is the work of Lai and Malovolta [LM19] who built a SNARG by combining a linear PCP with a linear map commitment. Before describing the construction, we first give informal definitions of these two building blocks:

- A (non-adaptive) linear PCP $(\mathbf{P}, \mathbf{V})$ over a finite field $\mathbb{F}$ generalizes the standard notion of a PCP by allowing the verifier to query linear functions of the proof, rather than individual coordinates as in a standard PCP. In more detail, the prover generates a proof $\pi \in \mathbb{F}^{\ell}$, and the verifier, given an instance $\mathbb{x}$ and randomness ρ, produces a matrix $M \in \mathbb{F}^{\ell \times q}$ (which we interpret as q linear queries). The verifier then receives as an answer the vector $\alpha = M\pi \in \mathbb{F}^q$, and makes its decision. Completeness and soundness are defined analogously to those in a standard PCP.
- A linear map commitment $(\mathsf{Setup}, \mathsf{Com}, \mathsf{Open}, \mathsf{Verify})$ is a commitment scheme with the following properties. After a setup phase Setup that generates a common reference string crs, the sender generates a commitment $\mathsf{c} = \mathsf{Com}(\mathsf{crs}, \pi)$ to a string $\pi \in \mathbb{F}^{\ell}$. Given a matrix $M \in \mathbb{F}^{\ell \times q}$, the sender can produce an answer and a decommitment proof $(\alpha, \mathsf{pf}) = \mathsf{Open}(\mathsf{crs}, \pi, M)$ attesting to the fact that the vector π committed to in c satisfies $\alpha = M\pi$, which is verified using $\mathsf{Verify}(\mathsf{crs}, \mathsf{c}, M, \alpha, \mathsf{pf})$.

Lai and Malavolta [LM19] combine linear PCPs with linear map commitments and the Fiat-Shamir transformation [FS86] to construct a SNARG $(\mathcal{S}, \mathcal{P}, \mathcal{V})$ in the random oracle model, where all parties are given access to a random function $\mathcal{O}$. For a security parameter λ, their scheme works as follows:

- $\mathcal{S}(1^n, 1^\lambda)$: Generate reference string crs using Setup.
- $\mathcal{P}(\mathsf{crs}, \mathbb{x}, \mathbb{w})$:
  1. Compute $\pi = \mathbf{P}(\mathbb{x}, \mathbb{w})$ and a commitment $\mathsf{c} = \mathsf{Com}(\mathsf{crs}, \pi)$.
  2. Query $\rho = \mathcal{O}(\mathbb{x}, \mathsf{c})$ and use $\mathbb{x}$ and ρ to compute the PCP verifier query matrix M.
  3. Compute $(\alpha, \mathsf{pf}) = \mathsf{Open}(\mathsf{crs}, \pi, M)$, a decommitment proof to the answer $\alpha = M\pi$.
  4. Output the proof $\Pi = (\mathsf{c}, \alpha, \mathsf{pf})$.
- $\mathcal{V}(\mathsf{crs}, \mathbb{x}, \Pi = (\mathsf{c}, \alpha, \mathsf{pf}))$:
  1. Query $\rho = \mathcal{O}(\mathbb{x}, \mathsf{c})$ and use $\mathbb{x}$ and ρ to compute the PCP verifier query matrix M.
  2. Accept if and only if the PCP verifier accepts given $\mathbb{x}$, randomness ρ, and answer α, and if $\mathsf{Verify}(\mathsf{crs}, \mathsf{c}, M, \alpha, \mathsf{pf}) = 1$.

The above scheme is a SNARG[2] with argument size $|\mathsf{c}| + |\mathsf{pf}| + |\alpha|$, with completeness error α and soundness error $\beta' = \mathrm{poly}(\lambda) \cdot \beta + \mathsf{negl}(\lambda)$ against $\mathrm{poly}(\lambda)$-size adversaries, where α and β are the completeness and soundness errors of the PCP respectively (as we will discuss later on in this section, [LM19] do not do a tight security analysis for their scheme). Thus, in order to construct SNARGs with short proof length using this paradigm, we need a linear map commitment with small commitment and opening sizes, and a PCP with small answer size (which, since $\alpha \in \mathbb{F}^q$, means working over a small field with small query complexity q).

Lai and Malavolta construct a linear map commitment in the bilinear generic group model ($\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$), where the groups have prime order $p \geq 2^\lambda$. Their commitment scheme satisfies $|\mathsf{c}| = |\mathsf{pf}| = 1\mathbb{G}_1$, and the underlying field is $\mathbb{F} = \mathbb{Z}_p$. Consequently, the resulting SNARG has size $2\mathbb{G}_1 + q \cdot \lambda$ bits and achieves at most $\lambda/2$ bits of security, since discrete logarithm attacks run in time proportional to the square root of the group size.

It seems that we have reached an impasse. Despite extensive research on improving the parameters of linear PCPs (e.g., [ALMSS98; IKO07; BCIOP13; BIOW20; BHIRW24]), even under a dream scenario with a 1-query PCP defined over a field of size 2^λ (so as to achieve λ bits of security), the resulting proof would still consist of two group elements (one for the commitment and one for the opening proof) plus λ additional bits (for the answer α). Thus, our central goal of obtaining SNARGs with exactly two group elements remains out of reach.

2.1.2 Reducing to two group elements

We realize the goal of only two group elements as follows. We have the prover *not send the query answer* α; instead, the verifier computes α locally by itself! At first glance, this seems paradoxical: how can a PCP be verified without receiving the query result? If the verifier can compute α by itself, maybe it does not need the prover at all, rendering the language to be easily decidable. We observe, however, that this is not a contradiction. Even if the verifier can predict α in advance, it still "learns new information" by making the specified query; namely, that the prover has committed to a string π such that $M\pi = \alpha$.

Another way to describe the property we require from the PCP is via its *accepting set*, which we want to be minimal–specifically, of size 1. Formally, for an instance $\mathbb{x}$ and verifier randomness ρ, the accepting set $S_{\mathbb{x},\rho}$ consists of all answers that would make the verifier accept. The accepting set size s is defined as the maximum size of $S_{\mathbb{x},\rho}$ over all instances and randomness.[3] This parameter provides a way to minimize communication: if the PCP has accepting set size s, only $\log s$ bits are needed to encode the message sent to the verifier.[4] Pushing this idea to the extreme, if $s = 1$, then *no bits* need to be communicated.

The above discussion motivates optimizing linear PCPs for accepting set size, rather than for query complexity or field size–once the length of α no longer affects communication, these traditionally important parameters become far less critical. This naturally leads to the following question:

Do linear PCPs with accepting set size 1 *exist?*

Linear PCPs with accepting set size 1. We answer this question in the affirmative. In particular, the linear PCP for the NP-complete language $\mathsf{GapMWSP}$ described in [BIOW20]—previously

[2]In fact, the transformation results in a SNARK if the PCP satisfies knowledge soundness.

[3]The value $\log s$ is sometimes called the "free bit complexity" of the PCP.

[4]This assumes an efficient mapping exists between the accepting set and the bit encoding.

used to construct 2-message laconic arguments—satisfies this property. We now sketch their PCP and explain why its accepting set has size 1:

Lemma 1. *There is a constant $c > 1$ such that there exists a linear PCP for* $\mathsf{GapMWSP}$ *with completeness error $1/\log n$, soundness error $n^{-\log^c n} + 1/|\mathbb{F}|$ and accepting set size $s = 1$.*

Proof sketch. We first (informally) describe the $\mathsf{GapMWSP}$ language: given $(A, b, d) \in \mathbb{F}^{\ell \times n} \times \mathbb{F}^{\ell} \times \mathbb{N}$ decide whether (in the Yes case) there exists x such that $Ax = b$ and has Hamming weight at most d, or (in the No case) every x with $Ax = b$ has Hamming weight at least $d \cdot \mathrm{polylog}(n)$.

We now sketch the linear PCP $(\mathbf{P}, \mathbf{V})$ for this language, due to [BIOW20], and explain why its accepting set size is 1. See [BIOW20] for an analysis of completeness and soundness.

- $\mathbf{P}((A, b, d), x)$: let $\pi = x$ be the proof string.
- $\mathbf{V}(A, b, d)$: Choose a random vector r and a sparse vector e, and make the query $M = (r^T A + e^T)$ (in this case the matrix M is a vector). Given an answer $\alpha = Mx$, accept if and only if $\alpha = r^T b$.

Observe that the verifier accepts if and only if the answer α that it receives is equal to $r^T b$, where this target value $r^T b$ can be computed by the verifier given the input and its randomness. Thus, the PCP has accepting set $\{r^T b\}$ of size 1. $\square$

Our main scheme (Theorem 1.1) is obtained by instantiating the previously described construction with the PCP from Lemma 1. The resulting SNARG has soundness error $\mathrm{poly}(\lambda) \cdot (n^{-\mathrm{polylog}\, n} + 1/|\mathbb{F}|) + \mathsf{negl}(\lambda)$ for $|\mathbb{F}| \approx 2^{\lambda}$, and thus it satisfies soundness $\mathsf{negl}(\lambda)$ when $\lambda \leq \mathrm{polylog}\, n$. This suffices for most realistic settings.

The Samorodnitsky and Trevisan PCP. Outside of this parameter regime, this PCP does not suffice. Note that standard repetition of the PCP does help in this case: since the completeness error of the PCP is nonzero, the amplified verifier's decision will need to be equal to the majority decision of each execution, which increases the accepting set size.

To fulfill Theorem 1.2, we turn to a second PCP. Samorodnitsky and Trevisan [ST06] describe a PCP that, assuming the *unique games conjecture* (UGC),[5] for a parameter k satisfies: completeness error 2^{-k}, soundness error $O(k/2^k)$, query complexity $O(k)$, and accepting set size k. To reduce the accepting set size from k to 1, we apply the following transformation: the verifier guesses a random element in the accepting set and accepts only if the answer is equal to its guess. This converts a PCP with accepting set size k and completeness error α into one with accepting set size 1 and completeness error $\alpha + (1 - 1/k)$, while leaving soundness unchanged.

By instantiating this PCP with $k = \lambda$, we obtain (assuming the UGC) a 2-group-element SNARG with negligible soundness error in any parameter regime, thereby proving Theorem 1.2. An interesting open problem is to construct linear PCPs with inverse exponential soundness error and accepting set size 1, without relying on additional assumptions (we stress that the unique games conjecture is used here for completeness rather than soundness, as it is a conjecture on NP reductions rather than a security assumption).

Amplifying completeness. The SNARG resultant from the construction (roughly) preserves the completeness and soundness errors of the PCP. While we have been careful to ensure that the soundness errors of our PCPs are negligible, the completeness errors are highly inadequate: the

[5]See Section 1.1 for a brief description of the unique games conjecture.

[BIOW20] PCP has error $1/\log n$ and the PCP derived from [ST06] has error $1 - \Theta(1/\lambda)$. As mentioned previously, standard repetition of the PCP cannot be used to decrease the completeness error, as it increases the accepting set size.

We instead get around this issue by altering the SNARG construction rather than the PCP: we have the honest prover "resample" the query matrix M until the verifier accepts. This must be done without affecting the argument size. We describe the new prover for a parameter t, assuming already that the PCP has accepting set size 1, and so the query answer is not sent:

```
• $\mathcal{P}(\mathsf{crs}, \mathbb{x}, \mathbb{w})$:
  1. Compute $\pi = \mathbf{P}(\mathbb{x}, \mathbb{w})$.
  2. For $i = 1$ to $t$:
     (a) Compute $\mathsf{c}_i$ to the string $\pi_i = (\pi || i)$.[6]
     (b) Query $\rho_i = \mathcal{O}(\mathbb{x}, \mathsf{c}_i)$ and use $\mathbb{x}$ and $\rho_i$ to compute the PCP verifier query matrix $M_i$.
     (c) Check whether the PCP verifier accepts given answer $\alpha_i = M_i \pi$.
     (d) If the above check passes then compute $\mathsf{pf}$, a decommitment proof to the answer $\alpha_i = M_i \pi$
         and output SNARG proof $\Pi = (\mathsf{c}_i, \mathsf{pf})$.[7]
     (e) Otherwise, go to the next iteration.
  3. Output $\perp$.
```

The verifier is identical to the verifier described in the beginning of this section, except that it pads the matrix M with zeroes so that it has the same number of rows as π_i.

The key idea in the above construction is that, since each c_i commits to a distinct vector, the security of the commitment scheme guarantees (up to negligible probability) that all c_i are unique (as otherwise, one could break binding). The Fiat–Shamir hash (the random oracle $\mathcal{O}$) applied to unique commitments, yields a fresh uniformly random PCP randomness. This effectively allows the prover to resample randomness for the PCP verifier, providing multiple independent attempts to make the verifier accept. If the underlying PCP has completeness error α, then the completeness error of the resulting SNARG is approximately α^t. As the verifier remains essentially unchanged, this transformation boosts completeness without affecting the proof size or the soundness error relative to the original scheme.

2.1.3 Tight analysis via multi-extraction

Our goal of precisely quantifying the security bits of our SNARG requires a tight analysis that minimizes error losses. We describe how the analysis of [LM19] works, identify the sources of security loss at each step, and explain how we modified the analysis to achieve tightness.

1. *Prove function binding of the linear map commitment.* Roughly, function binding means that for a commitment c no efficient sender can supply $(\alpha_1, M_1, \mathsf{pf}_1)$ and $(\alpha_2, M_2, \mathsf{pf}_2)$ so that verification passes on both using commitment c, but there exists no vector π such that simultaneously $M_1\pi = \alpha_1$ and $M_2\pi = \alpha_2$. In [LM19], this analysis is done loosely, describing a $\mathsf{negl}(\lambda)$ error for polynomial-time adversaries.

[6]For simplicity of this exposition we treat i an element in the field $\mathbb{F}$ of the linear PCP.

[7]Formally, since the commitment is to π_i, the decommitment attests to $\alpha_i = M_i' \pi_i$ where M_i' is M_i padded with zeroes.

We give a tight analysis in the GGM, showing that the linear map commitment has function binding error $\varepsilon = s^2/p$ (up to small dependencies on the committed vector length) against s-size adversaries where p is the order of the GGM group.

2. *Straightline extraction.* The [LM19] scheme is the application of the Fiat–Shamir [FS86] transformation on the interactive protocol where the prover sends c, the verifier chooses a uniformly random challenge ρ, and the prover replies with (α, pf). By function-binding of the linear map commitment, there must be a single string π that is consistent with all valid answers. Lai and Malovolta show that this interactive protocol is sound by running the prover up until it sends c, and repeatedly generating matrix-answer pairs until an entire PCP proof π can be derived. This reduces soundness to that of the PCP. Here, extracting π by repeatedly running the prover is very costly.

   We reduce this error by showing that the linear map commitment is *straightline extractable.* Roughly speaking, the linear-map commitment is straightline extractable if the committed to string can be extracted given only a commitment and the query–answer trace made by the adversary to the GGM oracle. This bypasses the need to rewind the interactive protocol, as π can be extracted given a single commitment. Using straightline extraction, it can be shown that the interactive protocol derived from the scheme prior to Fiat–Shamir has tight soundness error $\beta' = \beta + \varepsilon$ where β is the soundness error of the PCP, and ε is the same binding error as described above.

3. *Fiat–Shamir via multi-extraction.* Lai and Malovolta apply the Fiat–Shamir transformation generically to the interactive protocol above. Against s-size adversaries, incurs error (roughly) $s \cdot \beta'$. Intuitively, this is because the prover can try s different first messages (i.e., c), each of which will generate verifier randomness, thus giving it another attempt to attack the PCP. Thus, even if we use the tight analysis described above with straightline extraction, we would get an error of $s \cdot \beta' \approx s \cdot \beta + s^3/p$. While the $s \cdot \beta$ term is to be expected and tight (since the prover can try multiple PCPs), the term s^3/p is not. Since the linear map commitment is secure in the GGM, the random oracle calls do not grant the prover any power, and so we would expect to see a factor of s^2/p rather than s^3/p.

   To resolve this issue and get a tight bound, we strengthen the security of the linear map commitment, showing that it is *multi-extractable.* At a high-level, multi-extraction considers the prover's success probability at breaking the commitment in a setting where it has many attempts at different commitments (in our case, due to multiple Fiat–Shamir queries). An analog of this definition for Merkle trees was given in [CY24] in the context of standard PCPs.

   We provide an informal yet detailed definition:

   Definition 2.1 (Multi-extraction, informal)**.** *A linear map commitment scheme satisfies multi-extraction with error ε if there exists an efficient extractor such that the following holds for any efficient multi-step adversary $\mathcal{A} = (\mathcal{A}_1, \dots, \mathcal{A}_n, \mathcal{A}')$:*

$$\Pr\left[\begin{array}{l|l} \left(\begin{array}{l} \mathsf{Com.Verify}(\mathsf{crs}, \mathsf{c}, M, \alpha, \mathsf{pf}) = 1 \\ \wedge\ M x_{i^*} \neq \alpha \end{array}\right) & \mathsf{crs} \leftarrow \mathsf{Com.Setup}(1^\lambda) \\ \vee & \mathsf{aux}_0 := \bot \\ \left(\begin{array}{l} \exists i_1, i_2 \in [n] : \\ \mathsf{c}_{i_1} = \mathsf{c}_{i_2}\ \wedge\ x_{i_1} \neq x_{i_2} \end{array}\right) & \forall i \in [n] : \left(\begin{array}{r} (\mathsf{c}_i, \mathsf{aux}_i) \leftarrow \mathcal{A}_i(\mathsf{crs}, \mathsf{aux}_{i-1}) \\ x_i \leftarrow \mathsf{Ext}(\mathcal{A}_1, \dots, \mathcal{A}_i, \mathsf{crs}) \end{array}\right) \\ & (i^*, M, \alpha) \leftarrow \mathcal{A}'(\mathsf{crs}, \mathsf{aux}_n) \end{array}\right] \leq \varepsilon\,.$$

We show that the [LM19] linear map commitment has (straightline) multi-extraction error roughly $\varepsilon = s^2/p$ against s-size adversaries, where p is the order of the GGM.

We show tight bounds for our SNARG construction assuming a linear map commitment with multiextraction:

Lemma 2 (informal)**.** *Given a linear PCP with completeness error α and soundness error β, and a linear-map commitment with multiextraction error ε against s-size adversaries, the SNARG described in this section has completeness error $\alpha + \varepsilon$ and soundness error $s \cdot \beta + \varepsilon$ against s-size adversaries.*

See Lemma 5.11 for a formal statement.

Given all of the above, we prove that if the PCP has soundness error β and the GGM has order $p > 2^\lambda$, then the SNARG has error (roughly) $O(s \cdot \beta + s^2/2^\lambda)$, thus deriving the tight error bounds of Theorem 1.1 and Theorem 1.2.

2.2 Lower bound for SNARGs with one group element

In this section, we outline the proof of Theorem 1.3, showing a lower-bound for pairing-based one group element SNARGs in the GGM ($\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$) + ROM (denoted by $\mathcal{O}$) where (1) the setup algorithm does not use the random oracle, and (2) the verifier performs only non-adaptive queries to the random oracle prior to any GGM queries (i.e., no query depends on the answer of a different query or, equivalently, all of the verifier's queries are made at the same time, and this is prior to any GGM equation checks). In this overview, we consider a simplified case compared to Theorem 1.3:

- the SNARG proof is an element in $\mathbb{G}_1$,
- the random oracle receives a group element and outputs a bit (i.e., $\mathcal{O} \colon \mathbb{G}_1 \cup \mathbb{G}_2 \cup \mathbb{G}_T \to \{0,1\}$),
- the common reference string includes the generators of $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ (this is without loss of generality), and
- the verifier is deterministic.[8]

We stress that the full proof supports SNARGs where the (single) group element in the proof an element of either of the groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ and where the random oracle receives any number of group elements or bits and outputs bits and group elements (e.g., $\mathcal{O} \colon (\{0,1\} \cup \mathbb{G}_1 \cup \mathbb{G}_2 \cup \mathbb{G}_T)^* \to \{0,1\} \times \mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{G}_T$). Before giving a high-level description of our proof, we identify two different types of ROM queries (in fact, these are types of group elements that the verifier has access to).

Types of ROM queries. Consider a SNARG in the GGM + ROM with the simplifications described above. We observe that in the Maurer GGM, since the verifier cannot depend on the description of group elements that it receives, the group element that the verifier uses to query the random oracle can be described in terms of linear functions of the group elements in the common reference string and the element sent as the SNARG proof Π.

To explain in more detail, we first need some notation. Here, we use additive notation for the group. Specifically, a $\mathbb{G}_1$ element with discrete logarithm q will be denoted by $[q]_1$. Similarly, $[q]_2$ and $[q]_T$ are group elements in $\mathbb{G}_2$ and $\mathbb{G}_T$, respectively. If it is not relevant which of the groups an element is in we write $[q]_*$. In this notation the following operations are efficient:

[8]SNARG verifiers are sometimes considered to be deterministic. However, we can rule out SNARGs with a randomized verifier. In fact, the transformations that we will see later on in this section will have the verifier end up being randomized.

- For $z \in \{1, 2, T\}$ and $x, y \in \mathbb{Z}_p$ two $\mathbb{G}_z$ elements $[a]_z, [b]_z$ can be linearly combined

$$x[a]_z + y[b]_z = [xa + yb]_z$$

- A $\mathbb{G}_1$ element $[a]_1$ and a $\mathbb{G}_2$ element $[b]_2$ can be multiplied $[a]_1 \cdot [b]_2 = [ab]_T$.

We call the $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ elements in the common reference string $[\mathsf{crs}_1]_1$, $[\mathsf{crs}_2]_2$, and $[\mathsf{crs}_T]_T$, respectively. Observe that the common reference string includes the generators of $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$, and so these vectors all also have the value $1 \in \mathbb{Z}_p$.

Consider a query $[q]_z$ for $z \in \{1, 2, T\}$ that the verifier makes to the random oracle. As we are in the Maurer GGM, the only things that the verifier can do to generate group elements are linear operations on existing group elements and pair $\mathbb{G}_1$ with $\mathbb{G}_2$ elements. We have three cases depending on whether the query is a $\mathbb{G}_1$, $\mathbb{G}_2$, or $\mathbb{G}_T$ element:

- $\mathbb{G}_1$: $[q]_1 = a \cdot [\Pi]_1 + \langle \mathbf{b}, [\mathsf{crs}_1]_1 \rangle$
- $\mathbb{G}_2$: $[q]_2 = \langle \mathbf{b}, [\mathsf{crs}_2]_2 \rangle$
- $\mathbb{G}_T$: $[q]_T = [\Pi]_1 \cdot \langle \mathbf{a}, [\mathsf{crs}_2]_2 \rangle + \langle \mathbf{b}, [\mathsf{crs}_1]_1 \otimes [\mathsf{crs}_2]_2 \rangle + \langle \mathbf{c}, [\mathsf{crs}_T]_T \rangle$

Above, $a \in \mathbb{Z}_p$ and $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$ are vectors of $\mathbb{Z}_p$ elements.

We say that a query is *proof-dependent* if (1) it is a $\mathbb{G}_1$ element and $a \neq 0$, or (2) it is a $\mathbb{G}_T$ element and $\langle \mathbf{a}, [\mathsf{crs}_2]_2 \rangle \neq [0]_2$. Any query that is not proof-dependent (which includes all $\mathbb{G}_2$ element queries) is considered *proof-independent*.

Proof overview. The main idea of our proof is to first remove the random oracle, thereby transforming the SNARG in the GGM + ROM to one simply in the GGM, and then showing that such SNARGs do not exist. The next sections are as follows:

1. In Section 2.2.1 we show how to remove the ROM queries in the case where all of them are proof-dependent.
2. In Section 2.2.2 we explain how to generalize this to the case where the verifier's ROM queries may be arbitrarily proof-dependent or independent.
3. In Section 2.2.3 we show that there do not exist one group element SNARGs in the GGM.

Put together, they conclude the proof of Theorem 1.3. See Section 6 for a full formal theorem statement and proof.

2.2.1 Proof-dependent ROM query

We show how to transform a SNARG in the GGM + ROM, where all of the verifer's queries are proof-dependent, into one in the GGM without a random oracle. Our construction removes the random oracle by having each party separately (and with no coordination) lazily sample a random oracle. We first describe the transformation and then discuss further. Let $(\mathcal{S}^{\mathcal{G}}, \mathcal{P}^{\mathcal{G},\mathcal{O}}, \mathcal{V}^{\mathcal{G},\mathcal{O}})$ be a SNARG in the GGM + ROM where all of the verifier's ROM queries are proof-dependent. The new SNARG is as follows:

- $\mathcal{S}'^{\mathcal{G}}(1^n, 1^\lambda)$: Compute $\mathsf{crs} \leftarrow \mathcal{S}^{\mathcal{G}}(1^n, 1^\lambda)$. Output crs.

- $\mathcal{P}'^{\mathcal{G}}(\mathsf{crs}, \mathbb{x}, \mathbb{w})$: Compute $[\Pi]_1 \leftarrow \mathcal{P}^{\mathcal{G},\mathcal{O}_\mathcal{P}}(\mathsf{crs}, \mathbb{x}, \mathbb{w})$, where $\mathcal{O}_\mathcal{P}$ is a lazily sampled random oracle. Output Π.

- $\mathcal{V}'^{\mathcal{G}}(\mathsf{crs}, \mathbb{x}, \mathbb{w})$: Run $\mathcal{V}^{\mathcal{G},\mathcal{O}_{\mathcal{V}}}(\mathsf{crs}, \mathbb{x})$, where $\mathcal{O}_{\mathcal{V}}$ is a lazily sampled random oracle. Accept if and only if $\mathcal{V}$ accepts.

By construction, the new SNARG does not rely on a random oracle. Furthermore, it is immediate that soundness still holds as the original was sound even when the malicious prover had query access to the random oracle seen by the verifier. However, it is not clear at all that the new protocol satisfies *completeness.*

To thoroughly understand why the protocol remains complete, we first consider the case where the verifier makes a single query to the random oracle (the prover may still make many queries). Let p be the probability that the prover and verifier both make the same query to the random oracle. We are in one of two cases:

1. *The probability p is small:* Since the probability that $\mathcal{P}$ and $\mathcal{V}$ make the same query is small, their view (in the original SNARG) of the random oracle with significant probability is indistinguishable from two independent random oracles. Thus, in this case, our transformation changes little, and completeness does not change when prover an verifier queries are distinct.

2. *The probability p is large:* In this case the prover makes the query $[q]_*$ with high probability. We henceforth condition on the prover making the same query as the verifier. Let $f(\mathsf{crs}, [\Pi]_1)$ be the function computing the verifier's query $[q]_*$ from the reference string and the proof. Then, informally writing the verifier as receiving the query answer to make its decision, the verifier's acceptance probability can be written as:

$$\Pr_{\mathsf{crs},\mathcal{O}}\left[\mathcal{V}^{\mathcal{G}}(\mathsf{crs},\mathbb{x},\alpha)=1 \;\middle|\; \begin{array}{l}[\Pi]_1 \leftarrow \mathcal{P}^{\mathcal{G},\mathcal{O}}(\mathsf{crs},\mathbb{x},\mathbb{w})\\ [q]_* = f([\Pi]_1,\mathsf{crs})\\ \alpha = \mathcal{O}([q]_*)\end{array}\right].$$

For every i, let $\mathcal{P}_i^{\mathcal{G},\mathcal{O}}(\mathsf{crs}, \mathbb{x}, \mathbb{w})$ be the prover algorithm run up until it makes the i-th ROM query, and let this query be its output. Let i^* be the random variable representing the index of the prover ROM query that is equal to the verifier's query, so that $[q]_* = \mathcal{P}_{i^*}^{\mathcal{G},\mathcal{O}}(\mathsf{crs}, \mathbb{x}, \mathbb{w})$. We will show below that f is a *permutation* with respect to $[\Pi]_1$, so that there exists f^{-1} where $f^{-1}([q]_*, \mathsf{crs}) = [\Pi]_1$. Thus, we can rewrite the above probability as:

$$= \Pr_{\mathsf{crs},\mathcal{O}}\left[\mathcal{V}^{\mathcal{G}}(\mathsf{crs},\mathbb{x},\alpha)=1 \;\middle|\; \begin{array}{l}[q]_* = \mathcal{P}_{i^*}^{\mathcal{G},\mathcal{O}}(\mathsf{crs},\mathbb{x},\mathbb{w})\\ [\Pi]_1 = f^{-1}([q]_*,\mathsf{crs})\\ \alpha = \mathcal{O}([q]_*)\end{array}\right].$$

Observe that the verifier is oblivious to all of the ROM queries made by the prover up to index i^*. Moreover, the prover's algorithm does not depend on the answer to the query $[q]_*$. Thus, we can separate the single random oracle into two independent random oracles:

$$= \Pr_{\mathsf{crs},\mathcal{O}_{\mathcal{P}},\mathcal{O}_{\mathcal{V}}}\left[\mathcal{V}^{\mathcal{G}}(\mathsf{crs},\mathbb{x},\alpha)=1 \;\middle|\; \begin{array}{l}[q]_* = \mathcal{P}_{i^*}^{\mathcal{G},\mathcal{O}_{\mathcal{P}}}(\mathsf{crs},\mathbb{x},\mathbb{w})\\ [\Pi]_1 = f^{-1}([q]_*,\mathsf{crs})\\ \alpha = \mathcal{O}_{\mathcal{V}}([q]_*)\end{array}\right].$$

Now we can reverse the process we did before to bring back the prover running in full, but now where the two parties run with independent oracles:

$$= \Pr_{\mathsf{crs},\mathcal{O}_{\mathcal{P}},\mathcal{O}_{\mathcal{V}}}\left[\mathcal{V}^{\mathcal{G}}(\mathsf{crs},\mathbb{x},\alpha)=1 \;\middle|\; \begin{array}{l}[\Pi]_1 \leftarrow \mathcal{P}^{\mathcal{G},\mathcal{O}_{\mathcal{P}}}(\mathsf{crs},\mathbb{x},\mathbb{w})\\ [q]_* = f([\Pi]_1)\\ \alpha = \mathcal{O}_{\mathcal{V}}([q]_*)\end{array}\right],$$

which is precisely the SNARG following the transformation.

Showing that f is a permutation. The only thing we are missing is to show that f is a permutation. To see this, let us consider how q is computed. We describe the case of the query being a $\mathbb{G}_1$ element, where the $\mathbb{G}_T$ case will be analogous, when replacing a with $\langle \mathbf{a}, \mathsf{crs}_2 \rangle$ and $\langle \mathbf{b}, \mathsf{crs}_1 \rangle$ with $\langle \mathbf{b}, \mathsf{crs}_1 \otimes \mathsf{crs}_2 \rangle + \langle \mathbf{c}, \mathsf{crs}_T \rangle$.

$$q = a \cdot \Pi + \langle \mathbf{b}, \mathsf{crs}_1 \rangle \ .$$

Recall that we are under the assumption that $[q]_*$ is proof-dependent, meaning that $a \neq 0$. Thus, we can rewrite the above as

$$\Pi = \frac{q - \langle \mathbf{b}, \mathsf{crs}_1 \rangle}{a} \ .$$

In other words, the proof string Π is uniquely defined by $[q]_*$ and by the common reference string, and so f is a permutation. Observe that it is in the final step, where we claim that Π is uniquely defined by $[q]_*$, that we crucially use the fact that the proof contains a single group element, as otherwise the proof could not be uniquely defined by one dependent query.

Handling multiple queries. Extending the analysis from a single ROM query to multiple queries follows a similar high-level structure, but requires more technical work for controlling the intersection between the prover's and verifier's queries. We show that, despite this complication, the core ideas carry over and suffice to establish the complete proof for the general case.

Furthermore, extending to a ROM that accepts multiple group elements as inputs adds complexity to the proof, but the same general ideas still apply, and we give a formal proof for this general case as well.

2.2.2 Handling any type of ROM query

In the previous section, we showed how to transform one group element SNARGs in the GGM + ROM with only proof-dependent queries into SNARGs in the pure GGM. However, this transformation fails when the verifier makes proof-independent queries. In this section, we describe how to remove all proof-independent queries without affecting the proof-dependent ones. Thus, the full transformation is:

- Transform $(\mathcal{S}^{\mathcal{G}}, \mathcal{P}^{\mathcal{G},\mathcal{O}}, \mathcal{V}^{\mathcal{G},\mathcal{O}})$ into a new SNARG scheme $(\mathcal{S}'^{\mathcal{G}}, \mathcal{P}'^{\mathcal{G},\mathcal{O}}, \mathcal{V}'^{\mathcal{G},\mathcal{O}})$ in the GGM + ROM where, for any common reference string and for any proof in the image of the prover, the verifier only makes proof-dependent queries to the random oracle.
- Transform $(\mathcal{S}'^{\mathcal{G}}, \mathcal{P}'^{\mathcal{G},\mathcal{O}}, \mathcal{V}'^{\mathcal{G},\mathcal{O}})$ into a new SNARG scheme $(\mathcal{S}''^{\mathcal{G}}, \mathcal{P}''^{\mathcal{G}}, \mathcal{V}''^{\mathcal{G}})$ in the GGM (without the random oracle).

The high-level idea of our transformation to get rid of proof-independent queries is that, since a ROM query is independent of the proof, it is essentially fixed once the common reference string and the instance are fixed. Thus, the answer is also fixed and outside the prover's control. Thus, there is no difference between it being chosen as the output of a random oracle and it being written down as part of the common reference string.

We begin by describing the transformation in the case that all queries are proof-independent, and then explain how the construction must change if, as in the general case, the queries can be either proof-dependent or proof-independent.

All queries are proof-independent. Consider a SNARG $(\mathcal{S}^{\mathcal{G}}, \mathcal{P}^{\mathcal{G},\mathcal{O}}, \mathcal{V}^{\mathcal{G},\mathcal{O}})$ where the verifier makes t ROM queries, all of which are proof-independent. We construct a new SNARG where all of these queries are removed (in fact, since these are all of the verifier's ROM queries, the random oracle is completely removed):

- $\mathcal{S}'^{\mathcal{G}}(1^n, 1^\lambda)$:
  1. Let $\mathsf{crs} \leftarrow \mathcal{S}^{\mathcal{G}}(1^n, 1^\lambda)$.
  2. For $i \in [t]$ sample $r_i \leftarrow \{0,1\}$ uniformly at random.
  3. Output $\mathsf{crs}' = (\mathsf{crs}, r_1, \ldots, r_t)$.
- $\mathcal{P}'^{\mathcal{G}}(\mathsf{crs}' = (\mathsf{crs}, r_1, \ldots, r_t), \mathbb{x}, \mathbb{w})$:
  1. Let $[\Pi']_1$ be an arbitrary $\mathbb{G}_1$ element (for concreteness we can pick the generator of $\mathbb{G}_1$.)
  2. Run $\mathcal{V}(\mathsf{crs}, \mathbb{x}, [\Pi']_1)$ where, on its i-th ROM query $[q_i]_*$, answer with r_i.
  3. Run $[\Pi]_1 = \mathcal{P}^{\mathcal{G},\mathcal{O}'}(\mathsf{crs}, \mathbb{x}, \mathbb{w})$, where $\mathcal{O}'$ returns r_i on query $[q_i]_*$ and is a lazily sampled random oracle at all other points.
  4. Output $[\Pi]_1$.
- $\mathcal{V}'^{\mathcal{G}}(\mathsf{crs}' = (\mathsf{crs}, r_1, \ldots, r_t), \mathbb{x}, [\Pi]_1)$:
  1. Simulate $\mathcal{V}(\mathsf{crs}, \mathbb{x}, [\Pi]_1)$, answering the i-th random oracle query $[q_i]_*$ with answer r_i.
  2. Accept if and only if $\mathcal{V}$ accepts.

Completeness and soundness of this new scheme follow by construction since all of the verifier's queries are proof-independent, putting the query answers into the reference string grants all parties an identical view of the verifier's query/answer pairs as in the ROM. Moreover, since these queries are independent of the proof, the prover gets the same queries when running the verifier with $[\Pi']_1$ as are made with the real proof $[\Pi]_1$.

Handling mixed query types. Recall our assumption that the verifier is non-adaptive with respect to the random oracle. This means that, without loss of generality, it can be thought of as making all queries at once (i.e., both proof-dependent and proof-independent ones). However, as the argument above fails for proof-dependent queries, we can only replace the proof-independent ones with randomness from the common reference string. However, which query is proof-dependent and which is proof-independent may depend on crs and $\mathbb{x}$, and so we cannot tell "at design time" which query is which. This initially seems difficult, as proof dependence is a property that depends on the (possibly hidden) exponent of the group elements that comprise the queries.

However, upon closer inspection, both the verifier (and therefore also the prover) can recognize whether a query is proof-dependent or not. Recall that all $\mathbb{G}_2$ queries are proof-independent. Any $\mathbb{G}_1$ query is of the form

$$[q]_1 = a \cdot [\Pi]_1 + \langle \mathbf{b}, [\mathsf{crs}_1]_1 \rangle \ ,$$

where proof dependence means that $a \neq 0$. It is easy to check whether $a \neq 0$.

Finally, in the $\mathbb{G}_T$ case, the query has the form

$$[q]_T = [\Pi]_1 \cdot \langle \mathbf{a}, [\mathsf{crs}_2]_2 \rangle + \langle \mathbf{b}, [\mathsf{crs}_1]_1 \otimes [\mathsf{crs}_2]_2 \rangle + \langle \mathbf{c}, [\mathsf{crs}_T]_T \rangle \ ,$$

and proof-dependence means that $\langle \mathbf{a}, \mathsf{crs}_2 \rangle \neq 0$. This test is equivalent to checking $\langle \mathbf{a}, [\mathsf{crs}_2]_2 \rangle \neq [0]_2$, which can be done without having to compute any discrete logarithms.

2.2.3 One group element SNARGs do not exist

At this point, following all of the previous sections, we have completely removed the random oracle from the SNARG, and we are left with one group element SNARG in the GGM. We show Theorem 1.3: assuming one-way functions in the standard model, there are no one group element SNARGs in the Maurer GGM. More specifically, letting G be a PRG, we show that a one group element SNARG for the relation $\mathcal{R} = \{(\mathsf{G}(s), s)\}$ can be used to distinguish between the distributions $\mathcal{D}_U$, which is a uniformly random string, and $\mathcal{D}_{\mathsf{G}}$, which is the output of the PRG with a uniformly random seed. We note that in order to give a more intuitive explanation, the high-level overview described in this section differs slightly from that of the formal proof in the main body of the paper (see proof of Lemma 6.11).

This proof can be viewed as an extension to Groth's lower bound [Gro16], who proves a similar theorem under the restriction that the verifier is a *conjunction* of pairing-equality checks.[9] We do not rely on the verifier being a conjunction. However, as we show below, we prove that any verifier must have a weaker structure that is sufficient for ruling out such SNARGs.

On the structure of the verifier. In the Maurer GGM, the verifier's only way of obtaining useful information is through GGM zero-test queries (equivalently, equality tests). Concretely, given a group element $[q]_*$, the verifier may query an oracle to determine whether q is zero. To analyze the structure of the verifier's algorithm, we once again use the notion of proof-dependent queries, previously applied to characterize queries to the ROM, and here adapted to capture zero-test oracle queries.

Recall that a proof-dependent query has one of the following two structures, depending on which group it belongs to:

- $\mathbb{G}_1$: $[q]_1 = a \cdot [\Pi]_1 + \langle \mathbf{b}, [\mathsf{crs}_1]_1 \rangle$ where $a \neq 0$
- $\mathbb{G}_T$: $[q]_T = [\Pi]_1 \cdot \langle \mathbf{a}, [\mathsf{crs}_2]_2 \rangle + \langle \mathbf{b}, [\mathsf{crs}_1]_1 \otimes [\mathsf{crs}_2]_2 \rangle + \langle \mathbf{c}, [\mathsf{crs}_T]_T \rangle$ where $\langle \mathbf{a}, \mathsf{crs}_2 \rangle \neq 0$.

Observe that in both cases there is a single and unique proof Π so that $q = 0$ and the zero-test returns 1 (e.g., in the $\mathbb{G}_1$ case $[\Pi]_1 = \frac{\langle \mathbf{b}, [\mathsf{crs}_1]_1 \rangle}{a}$).

Using these observations, we show that in order for the verifier to accept an instance sampled from $\mathcal{D}_{\mathsf{G}}$, there must be a proof-dependent zero-check that passes. As we will see later, this is enough structure to distinguish $\mathcal{D}_U$ from $\mathcal{D}_{\mathsf{G}}$.

Claim 2.2 (informal). *For $\mathbb{x} \leftarrow \mathcal{D}_{\mathsf{G}}$ and crs sampled as in the SNARG, if there exists $[\Pi]_1$ so that $\mathcal{V}(\mathsf{crs}, \mathbb{x}, [\Pi]_1) = 1$ then (except with negligible probability) there exists a proof-dependent zero-test query for which the answer is* 1.

Proof sketch. We show that for almost every $\mathbb{x}$ in the support of $\mathcal{D}_U$ (i.e., instances $\mathbb{x} \notin L(\mathcal{R})$) the verifier rejects if all proof-dependent queries evaluate to 0 with overwhelming probability. The claim follows as it is hard to distinguish $\mathcal{D}_U$ and $\mathcal{D}_{\mathsf{G}}$.[10]

Fix $\mathbb{x} \notin L(\mathcal{R})$ and some common reference string crs consider what it means for the verifier to accept when all of the proof-dependent queries evaluate to 0. Every proof-dependent query $[q_i]_*$

[9] Actually, Groth shows that there do not exist SNARGs *of any length* with such verifiers in the case where the entire proof is comprised of elements of the same group (e.g., the proof is only $\mathbb{G}_1$ elements). This, in particular, rules out the one group element case.

[10] Observe that the view of the verifier of $[\Pi]_1$ is only in the proof-dependent queries, so we can efficiently simulate this behaviour without knowing $[\Pi_1]_1$. Thus, showing that the PRG fools this efficient algorithm implies the same behavior for any $[\Pi]_1$ that induces the same verifier view.

returns 1 for exactly one group element $[\Pi_i]_1$. Since the verifier accepts if all these queries return 0, a malicious prover will cause the verifier to accept if it sends $[\Pi]_1$ that is different from all of the elements $[\Pi_i]_1$. Since the verifier makes at most $\mathrm{poly}(\lambda)$ queries, a prover that chooses a random group element for $[\Pi]_1$ causes the verifier to accept with probability $= 1 - \mathsf{negl}(\lambda)$.

Since the SNARG is sound, we conclude that the verifier accepts when all proof-dependent queries evaluate to 0 with negligible probability.

For $\mathbb{x}' \leftarrow \mathcal{D}_\mathsf{G}$, too, the verifier accepts when all proof-dependent queries evaluate to 0 with negligible probability. That is because $\mathbb{x} \leftarrow \mathcal{D}_U$ and $\mathbb{x}' \leftarrow \mathcal{D}_\mathsf{G}$ are computationally indistinguishable. If the verifier acted significantly different on $\mathbb{x}'$ then the PRG distinguisher can simulate the SNARG and detect the difference. □

Ruling out one group element SNARGs. Having established some structural understanding of the SNARG verifier, we now leverage this insight to construct a distinguisher against the PRG. To this end, we introduce a polynomial-time subroutine $\mathbf{A}_\mathcal{V}$, which takes as input a common reference string crs, an instance $\mathbb{x}$, and a function f as a polynomial with one of the following forms (matching to the relevant group):

- $\mathbb{G}_1$: $f([\Pi]_1, [\mathsf{crs}_1]_1) = a \cdot [\Pi]_1 + \langle \mathbf{b}, [\mathsf{crs}_1]_1 \rangle$ where $a \neq 0$
- $\mathbb{G}_T$: $f([\Pi]_1, [\mathsf{crs}_1]_1, [\mathsf{crs}_2]_2, [\mathsf{crs}_T]_T) = [\Pi]_1 \cdot \langle \mathbf{a}, [\mathsf{crs}_2]_2 \rangle + \langle \mathbf{b}, [\mathsf{crs}_1]_1 \otimes [\mathsf{crs}_2]_2 \rangle + \langle \mathbf{c}, [\mathsf{crs}_T]_T \rangle$ where $\langle \mathbf{a}, [\mathsf{crs}_2]_2 \rangle \neq [0]_2$.

The algorithm $\mathbf{A}_\mathcal{V}(\mathsf{crs}, \mathbb{x}, f)$ outputs a group element $[\Pi]_1$ that passes the zero-test if plugged in to a query of the form of f, i.e., in the $\mathbb{G}_1$ case, such that $f([\Pi]_1, [\mathsf{crs}_1]_1) = 0$.

In the case where f is a query made by the verifier and $\mathbb{x}$ is sampled from $\mathcal{D}_\mathsf{G}$, such an algorithm can be derived using ideas similar to those used in Groth's lower-bound [Gro16]. The algorithm essentially uses the fact that the verifier's queries form a linear subspace to progressively learn which answers will cause this query to pass. In the full proof (see Lemma 6.11) we give full details, but we note that they don't have the same high-level of abstraction as in this overview.

We now have all the tools we need in order to break the PRG:

Lemma 2.3. *If there exists a one group element SNARG in the Maurer GGM, then there exists an efficient distinguisher between $\mathcal{D}_U$ and $\mathcal{D}_\mathsf{G}$.*

Proof sketch. We describe the distinguisher $\mathbf{D}$. Suppose that the verifier makes at most t queries. Given an instance $\mathbb{x}$ the distinguisher $\mathbf{D}$ runs as follows:

1. Sample crs as in the SNARG.
2. For every $i = 1$ to t:
   (a) Run the verifier up to the i-th query (recall that we have established in prior sections that proof-dependent queries can be recognized), answering all previous proof-dependent queries with 0.
   (b) Compute $[\Pi_i]_1 = \mathbf{A}_\mathcal{V}(\mathsf{crs}, \mathbb{x}, f_i)$ where f_i is the structure of the i-th query (which can be efficiently derived by tracking all of the verifier's GGM queries).
3. If there exists i such that $\mathcal{V}(\mathsf{crs}, \mathbb{x}, [\Pi_i]_1) = 1$ then output 1. Otherwise, output 0.

We show that $\mathbf{D}$ distinguishes between the distributions by showing that, for instances sampled from $\mathcal{D}_b$, the distinguisher outputs the bit b with overwhelming probability.

Consider $\mathbb{x} \leftarrow \mathcal{D}_{\mathsf{G}}$. By completeness, there must be $[\Pi]_1$ that causes the verifier to accept. As shown in Claim 2.2, in order for this to be the case, it must be that $[\Pi]_1$ satisfies one of the proof-dependent zero-test queries. Let i^* be the index of the first proof-dependent zero test that passes. Since this element is unique, for $i = i^*$ the distinguisher will get from $\mathbf{A}_{\mathcal{V}}$ the output $[\Pi_{i^*}]_1 = [\Pi]_1$, and so $\mathcal{V}$ will accept on $[\Pi_{i^*}]_1$. Thus, in this case $\mathbf{D}$ will output 1.

Now consider $\mathbb{x} \leftarrow \mathcal{D}_U$. Since $\mathbf{D}$ is efficient and, after generating the common reference string, makes no further use of its randomness, it can be simulated by a malicious SNARG prover. By soundness of the SNARG, the verifier must therefore reject all candidate proofs $[\Pi_i]_1$ produced by $\mathbf{D}$. We conclude that the distinguisher outputs 0 for such instances. $\square$

3 Preliminaries

3.1 Idealized models

The SNARGs in this paper are proved secure in the generic group model with an additional random oracle. We describe these oracles:

Definition 3.1 (Maurer's generic group). *Maurer's generic group model introduces a new unit of information, a group element. In the setting of a group with asymmetric pairings, we have that a group element can be from one of three prime-p order groups $\mathbb{G}_1$, $\mathbb{G}_2$, or $\mathbb{G}_T$. We define a generator for each group g_1, g_2, and g_T, which every party implicitly receives. In this model, every algorithm can only interact with these group elements by using one of these three oracles:*

1. *$\mathcal{G}_l$ takes as input two group elements $g = g_i^a, h = g_i^b$ from the same group $\mathbb{G}_i$ with $i \in \{1, 2, T\}$ and $a, b \in \mathbb{Z}_p$ and outputs g_i^{a+b} in $\mathbb{G}_i$,*
2. *$\mathcal{G}_e$ takes as input two group elements $g = g_1^a \in \mathbb{G}_1$, $h = g_2^b \in \mathbb{G}_2$ for $a, b \in \mathbb{Z}_p$ and outputs $g_T^{a \cdot b}$ in $\mathbb{G}_T$,*
3. *$\mathcal{G}_c$ takes as input one group elements $g = g^a$ for some $a \in \mathbb{Z}_p$ from the group $\mathbb{G}_i$ with $i \in \{1, 2, T\}$ and outputs 1 if $a = 0$ and outputs 0 otherwise.*

We denote the combination of $\mathcal{G}_l$, $\mathcal{G}_e$, and $\mathcal{G}_c$ as $\mathcal{G}$.

Notice that the group elements do not have a bit representation. This way the only way an algorithm can learn *bits* from a group element is by querying $\mathcal{G}_c$.

Definition 3.2 (Random oracle). *We let RO be the set of all functions $\mathcal{O}\colon \{0,1\}^* \to \{0,1\}$. A construction in the "random oracle model" is a construction where all parties have oracle access to a uniformly random $\mathcal{O} \leftarrow \mathsf{RO}$.*

When simulating a random oracle, we use the notation "Lazily sample a random oracle $\mathcal{SO}$" and we then use $\mathcal{SO}$ in oracle algorithms. What that means in actuality is that first, we initiate an empty dictionary $\mathcal{SO}$. Then, whenever an oracle algorithm queries the oracle on some input X, check whether there is already an entry for X in $\mathcal{SO}$ .If not we sample a uniformly random Y and add $X \mapsto Y$ to the dictionary $\mathcal{SO}$.

We also define the generic group model and the random oracle model in the following way:

Definition 3.3 (GGM+ROM). *Maurer's generic group model, combined with the random oracle model, also has the unit of information, a group element. In the setting of a group with asymmetric pairings we have that a group element can be from one of three prime-p order groups $\mathbb{G}_1$, $\mathbb{G}_2$, or $\mathbb{G}_T$. We define a generator for each group g_1, g_2, and g_T which every party implicitly receives. We let RO be the set of all functions $\mathcal{O}\colon (\{0,1\} \cup \mathbb{G}_1 \cup \mathbb{G}_2 \cup \mathbb{G}_T)^* \to (\{0,1\} \times \mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{G}_T)$. In this model, we sample $\mathcal{O} \leftarrow \mathsf{RO}$ and every algorithm can only interact with these group elements by using one of these four oracles:*

1. *$\mathcal{G}_l$ takes as input two group elements $g = g_i^a, h = g_i^b$ from the same group $\mathbb{G}_i$ with $i \in \{1, 2, T\}$ and $a, b \in \mathbb{Z}_p$ and outputs g_i^{a+b} in $\mathbb{G}_i$,*
2. *$\mathcal{G}_e$ takes as input two group elements $g = g_1^a \in \mathbb{G}_1$, $h = g_2^b \in \mathbb{G}_2$ for $a, b \in \mathbb{Z}_p$ and outputs $g_T^{a \cdot b}$ in $\mathbb{G}_T$,*

3. *$\mathcal{G}_c$ takes as input one group elements $g = g^a$ for some $a \in \mathbb{Z}_p$ from the group $\mathbb{G}_i$ with $i \in \{1, 2, T\}$ and outputs 1 if $a = 0$ and outputs 0 otherwise,*

4. *$\mathcal{O}$ takes as input an arbitrary amount of group elements and bits and outputs an arbitrary one of those, too.*

We denote the combination of $\mathcal{G}_l$, $\mathcal{G}_e$, and $\mathcal{G}_c$ as $\mathcal{G}$.

Remark 3.4. We sometimes consider the random oracle as outputting many outputs. This can either be realized directly by a many-output ROM (matching a real-world hash function) or by adding further inputs and domain-separation as appropriate.

3.2 Succinct non-interactive arguments

Definition 3.5. *A succinct non-interactive argument (SNARG) for a relation $\mathcal{R} = \{(\mathbb{x}, \mathbb{w})\}$ is defined by a triple $(\mathcal{S}, \mathcal{P}, \mathcal{V})$ as follows:*

- Syntax.
  - *The setup algorithm $\mathcal{S}$ receives an input size parameter 1^n and a security parameter 1^λ. It outputs a common reference string* crs.
  - *The (honest) prover algorithm $\mathcal{P}$ receives as input a common reference string* crs*, instance $\mathbb{x} \in \{0,1\}^n$, and witness $\mathbb{w} \in \{0,1\}^m$. It outputs a proof* Π.
  - *The verifier algorithm $\mathcal{V}$ receives as an instance $\mathbb{x} \in \{0,1\}^n$, and a proof* Π*. It outputs a bit $b \in \{0,1\}$.*

- Completeness. *A SNARG has completeness error α if for all $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$ and $\lambda \in \mathbb{N}$:*

$$\Pr\left[\mathcal{V}(\mathbb{x}, \Pi) = 1 \;\middle|\; \begin{array}{c} \mathsf{crs} \leftarrow \mathcal{S}(1^{|\mathbb{x}|}, 1^\lambda) \\ \Pi \leftarrow \mathcal{P}(\mathsf{crs}, \mathbb{x}, \mathbb{w}) \end{array}\right] \geq 1 - \alpha(\lambda, |\mathbb{x}|)\,.$$

- Soundness. *A SNARG has (adaptive) soundness with error β if for every $\lambda \in \mathbb{N}$, $n \in \mathbb{N}$, and s-size prover $\mathcal{P}'$:*

$$\Pr\left[\begin{array}{c} |\mathbb{x}| \leq n \\ \mathbb{x} \notin L(\mathcal{R}) \\ \wedge\ \mathcal{V}(\mathsf{st}, \mathbb{x}, \Pi) = 1 \end{array} \;\middle|\; \begin{array}{c} \mathsf{crs} \leftarrow \mathcal{S}(1^n, 1^\lambda) \\ (\mathbb{x}, \Pi) \leftarrow \mathcal{P}'(\mathsf{crs}) \end{array}\right] \leq \beta(\lambda, n, s)\,.$$

- Succinctness. *For every large enough λ, $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$ and* crs *in the image of $\mathcal{S}(1^{|\mathbb{x}|}, 1^\lambda)$, we have $|\Pi| = o(\mathbb{w})$ for $\Pi = \mathcal{P}(\mathsf{crs}, \mathbb{x}, \mathbb{w})$.*

A SNARG with the following additional knowledge property is known as a SNARK:

- *Knowledge soundness.* A SNARG has (adaptive) knowledge soundness (in which case we refer to it as a SNARK) with knowledge κ if there exists an expected polynomial time PPT extractor $\mathcal{E}$ so that for every $\lambda \in \mathbb{N}$, $n \in \mathbb{N}$, and s-size prover $\mathbf{P}'$,

$$\Pr\left[\begin{array}{c} |\mathbb{x}| \leq n \\ (\mathbb{x}, \mathbb{w}) \notin \mathcal{R} \\ \wedge\ \mathcal{V}(\mathsf{st}, \mathbb{x}, \Pi) = 1 \end{array} \;\middle|\; \begin{array}{c} \mathsf{crs} \leftarrow \mathcal{S}(1^n, 1^\lambda) \\ (\mathbb{x}, \Pi) \leftarrow \mathcal{P}'(\mathsf{crs}) \\ \mathbb{w} \leftarrow \mathcal{E}(\mathcal{P}', \mathsf{crs}) \end{array}\right] \leq \kappa(\lambda, n, s)\,.$$

4 Linear map commitments

In this section we define linear map commitments, describe a construction in the generic group model due to Lai and Malovolta [LM19] and give a tight analysis of the errors in their linear map commitment. In fact, we prove a stronger security notion than the one proven i [LM19]; we prove multi-extraction.

We begin by defining linear map commitments:

Definition 4.1. *A linear map commitment over a finite field $\mathbb{F}$ is a tuple of algorithms*

$$(\mathsf{Com.Setup}, \mathsf{Com.Commit}, \mathsf{Com.Open}, \mathsf{Com.Verify})\,,$$

with the following properties:

- Correctness. *The linear map commitment has (perfect) correctness if for every $\lambda, \ell, q \in \mathbb{N}$, vector $x \in \mathbb{F}^{\ell}$ and matrix $M \in \mathbb{F}^{\ell \times q}$:*

$$\Pr\left[\mathsf{Com.Verify}(\mathsf{crs}, \mathsf{c}, M, \alpha, \mathsf{pf}) = 1 \;\middle|\; \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{Com.Setup}(1^{\lambda}, \ell, q) \\ (\mathsf{c}, \mathsf{aux}) \leftarrow \mathsf{Com.Commit}(\mathsf{crs}, x) \\ \mathsf{pf} \leftarrow \mathsf{Com.Open}(\mathsf{crs}, \mathsf{c}, \mathsf{aux}, M) \\ \alpha = Mx \end{array}\right] = 1\,.$$

- Multi-extraction. *The linear map commitment has multi-extraction error ε if there exists an extractor for every $\lambda, \ell, q, n, s \in \mathbb{N}$, there exists a polynomial-time extractor* Ext *such that for any s-size $n+1$-stage adversary $\mathcal{A} = (\mathcal{A}_1, \ldots, \mathcal{A}_n, \mathcal{A}')$:*

$$\Pr\left[\begin{array}{l} \left(\begin{array}{l} \mathsf{Com.Verify}(\mathsf{crs}, \mathsf{c}, M, \alpha, \mathsf{pf}) = 1 \\ \wedge\; Mx_i \neq \alpha \end{array}\right) \\ \vee \\ \left(\begin{array}{l} \exists i_1, i_2 \in [n] : \\ \mathsf{c}_{i_1} = \mathsf{c}_{i_2} \;\wedge\; x_{i_1} \neq x_{i_2} \end{array}\right) \end{array} \;\middle|\; \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{Com.Setup}(1^{\lambda}, \ell, q) \\ \mathsf{aux}_0 := \bot \\ \forall i \in [n] : \left(\begin{array}{r} (\mathsf{c}_i, \mathsf{aux}_i) \leftarrow \mathcal{A}_i(\mathsf{crs}, \mathsf{aux}_{i-1}) \\ x_i \leftarrow \mathsf{Ext}(\mathcal{A}_1, \ldots, \mathcal{A}_i, \mathsf{crs}) \end{array}\right) \\ (i, M, \alpha) \leftarrow \mathcal{A}'(\mathsf{crs}, \mathsf{aux}_n) \end{array}\right] \leq \varepsilon(\lambda, \ell, q, n, s)\,.$$

Remark 4.2. The linear map commitment that we use will be secure in the generic group model. In this case, we refer to errors as a function of the number of queries made by the adversary rather than its circuit size.

We now show that the [LM19] linear map commitment has the above properties and give a tight analysis of security:

Theorem 4.3. *There is a linear map commitment in the bilinear GGM with prime order p with $p \geq 2^{\lambda}$ over the field $\mathbb{F} = \mathbb{Z}_p$ with multi-extraction error $\frac{4\ell s^2}{2^{\lambda}}$ against s-query adversaries (as long as $s > 20\ell^2 q^2$). The commitment and opening sizes are both 1 element of $\mathbb{G}$. The committing time is ℓ source group operations, the opening time is $q\ell^2$ source group operations, and the verification entails 3 pairing operations and $\ell(q+1)$ additional group operations. The extraction time is $\mathsf{t}_{\mathsf{ext}}(s) = O(s)$.*

Proof. We begin by describing the construction:

Construction 4.4 ([LM19])**.** The linear map commitment is as follows. We assume all algorithms have access to a GGM oracle $\mathcal{G}$ with known prime order p, and all those that receive crs parse it as $\mathsf{crs} = \left(\{G_j\}_{j \in [\ell]}, \{H_{i,j}\}_{i \in [q], j \in [2\ell] \setminus \{\ell+1\}}\right)$.

- $\mathsf{Com.Setup}^{\mathcal{G}}(1^\lambda, \ell, q)$:
  1. $\beta, z_1, \ldots, z_q \leftarrow \mathbb{Z}_p$.
  2. For all $j \in [\ell]$ set $G_j = g_1^{\beta^j}$.
  3. For all $i \in [q]$ and $j \in [2\ell]$ set $H_{i,j} = g_2^{z_i \beta^j}$.
  4. Output $\mathsf{crs} = \left(\{G_j\}_{j\in[\ell]}, \{H_{i,j}\}_{i\in[q], j\in[2\ell]\setminus\{\ell+1\}}\right)$.

- $\mathsf{Com.Commit}^{\mathcal{G}}(\mathsf{crs}, x)$: Return $(\mathsf{c}, \mathsf{aux}) = \left(\prod_{j\in[\ell]} G_j^{x_j}, x\right)$.

- $\mathsf{Com.Open}^{\mathcal{G}}(\mathsf{crs}, \mathsf{c}, \mathsf{aux}, M)$: Parse $\mathsf{aux} = x$ and return pf defined as follows:

$$\mathsf{pf} = \prod_{i\in[q]} \prod_{j\in[\ell]} \prod_{j'\in[\ell]\setminus\{j\}} H_{i,\ell+1+j-j'}^{M_{i,j}\cdot x_{j'}} .$$

- $\mathsf{Com.Verify}^{\mathcal{G}}(\mathsf{crs}, \mathsf{c}, M, \alpha, \mathsf{pf})$: Accept if and only if:

$$e\left(\mathsf{c}, \prod_{i\in[q]}\prod_{j\in[\ell]} H_{i,\ell+1-j}^{M_{i,j}}\right) = e\left(G_1, \prod_{i\in[q]} H_{i,\ell}^{\alpha_i}\right) \cdot e\left(g_1, \mathsf{pf}\right) .$$

We show that the scheme above has extractable function binding. We describe an extractor Ext such that:

$$\Pr\left[\begin{array}{c|l} \left(\begin{array}{l} \mathsf{Com.Verify}(\mathsf{crs}, \mathsf{c}, M, \alpha, \mathsf{pf}) = 1 \\ \wedge\ Mx_i \neq \alpha \end{array}\right) & \mathcal{G} \leftarrow \mathsf{GGM}(1^\lambda) \\ & \mathsf{crs} \leftarrow \mathsf{Com.Setup}^{\mathcal{G}}(1^\lambda, \ell, q) \\ \vee & \mathsf{aux}_0 := \bot \\ \left(\begin{array}{l} \exists i_1, i_2 \in [n] : \\ \mathsf{c}_{i_1} = \mathsf{c}_{i_2}\ \wedge\ x_{i_1} \neq x_{i_2} \end{array}\right) & \forall i \in [n] : \left(\begin{array}{l} (\mathsf{c}_i, \mathsf{aux}_i) \leftarrow \mathcal{A}_i^{\mathcal{G}}(\mathsf{crs}, \mathsf{aux}_{i-1}) \\ x_i \leftarrow \mathsf{Ext}^{\mathcal{G}}(\mathcal{A}_1, \ldots, \mathcal{A}_i, \mathsf{crs}) \end{array}\right) \\ & (i, M, \alpha) \leftarrow \mathcal{A}'^{\mathcal{G}}(\mathsf{crs}) \end{array}\right] \leq \frac{4\ell s^2}{p} \leq \frac{4\ell s^2}{2^\lambda} .$$

We describe the extractor $\mathsf{Ext}^{\mathcal{G}}$ on inputs $\mathcal{A}_1, \ldots, \mathcal{A}_i$ and $\mathsf{crs} = \left(\{G_j\}_{j\in[\ell]}, \{H_{i,j}\}_{i\in[q], j\in[2\ell]\setminus\{\ell+1\}}\right)$:[11]

1. Initialize lists Λ_1, Λ_2, and Λ_T. These lists will hold entries of the form (g, f) where for $a \in \{1, 2, T\}$, $g \in \mathbb{G}_a$ and f is a polynomial over formal variables to be defined in the following item. Λ_1 is initiated as $\{(g_1, 1)\}$, Λ_2 is initiated as $\{(g_2, 1)\}$, and Λ_T is initially empty.

2. Let $X, Z_1, \ldots, Z_q$ be formal variables. For every $j \in [\ell]$ add (G_j, X^j) to Λ_1, and for every $i \in [q]$ and $j \in [2\ell]$ add $(H_{i,j}, Z_i \cdot X^j)$ to Λ_2.

3. Run $\mathcal{A}_1^{\mathcal{G}}(\mathsf{crs})$ up to $\mathcal{A}_i^{\mathcal{G}}(\mathsf{crs})$ in order, passing the auxiliary values between algorithms appropriately. Keep track of queries to the generic group as follows:

   - For a product query $h_1 \cdot h_2$ where $h_1, h_2 \in \mathbb{G}_a$ search Λ_a for entries (h_1, f_1) and (h_2, f_2). If no such entries exist, output $\bot$. If they exist and there is no entry of the form $(h', f_1 + f_2)$ then add $(h_1 \cdot h_2, f_1 + f_2)$ to Λ_a.

[11] Ext only needs the trace of oracle calls made by $\mathcal{A}_1, \ldots, \mathcal{A}_i$, and not additional access to the GGM oracle, but we write it in this way for simplicity.

- For a pairing query $e(h_1, h_2)$ where $(h_1, h_2) \in \mathbb{G}_1 \times \mathbb{G}_2$, search Λ_1 an entry (h_1, f_1) and Λ_2 for an entry (h_2, f_2). If no such entries exist, output $\perp$. If they exist and there is no entry in Λ_T of the form $(h', f_1 \cdot f_2)$ then add $(e(h_1, h_2), f_1 \cdot f_2)$ to Λ_T.

4. The adversary $\mathcal{A}_i$ eventually outputs c_i. If there is no entry (c_i, f) in Λ_1 then output $\perp$. Otherwise, due to how we are adding elements Λ_1, the polynomial f is over the variable X and has the form:
$$f(X) = \gamma_0 + \sum_{j \in [\ell]} \gamma_j \cdot X^j \,,$$
where $\gamma_0, \gamma_1, \ldots, \gamma_q \in \mathbb{Z}$.

5. Output $x_i = (\gamma'_1, \ldots, \gamma'_\ell) \in \mathbb{Z}_p^\ell$ where $\gamma'_j = \gamma_j \mod p$.

We additionally consider the following oracle $\mathcal{G}'$ which has an identical interface to the standard GGM oracle, except that it keeps lists of polynomials over formal variables in exactly the same way as done by Ext, and generates a fresh label whenever a polynomial appears which has not been seen before.

1. Initialize lists Λ_1, Λ_2, and Λ_T. These lists will hold entries of the form (g, f) where for $a \in \{1, 2, T\}$, $g \in \mathbb{G}_a$ and f is a polynomial over formal variables to be defined in the following item.

2. Choose a random label g_1 and add $(g_1, 1)$ to Λ_1. Similarly, add $(g_2, 1)$ to Λ_2 for a random label g_2.

3. Let $X, Z_1, \ldots, Z_q$ be formal variables For every $j \in [\ell]$ add (G_j, X^j) to Λ_1 where G_j is a random label not previously used, and for every $i \in [q]$ and $j \in [2\ell]$ add $(H_{i,j}, Z_i \cdot X^j)$ to Λ_2 where $H_{i,j}$ is a random label not previously used. When $\mathsf{Com.Setup}$ initiates crs, answer with the G_j and $H_{i,j}$ labels as appropriate.

4. For any query made by $\mathcal{A} = (\mathcal{A}_1, \ldots, \mathcal{A}_n, \mathcal{A}')$, $\mathsf{Com.Verify}$, or Ext, answer the query as follows:

   - For a product query $h_1 \cdot h_2$ where $h_1, h_2 \in \mathbb{G}_a$ search Λ_a for entries (h_1, f_1) and (h_2, f_2). If no such entries exist, output $\perp$. If they exist and there is no entry of the form $(h', f_1 + f_2)$ then choose a random unused label h' and add $(h', f_1 + f_2)$ to Λ_a.
   - For a pairing query $e(h_1, h_2)$ where $(h_1, h_2) \in \mathbb{G}_1 \times \mathbb{G}_2$, search Λ_1 an entry (h_1, f_1) and Λ_2 for an entry (h_2, f_2). If no such entries exist, output $\perp$. If they exist and there is no entry in Λ_T of the form $(h', f_1 \cdot f_2)$ then generate a random unused label h' and add $(h', f_1 \cdot f_2)$ to Λ_T.

We in show in Claim 4.5 that using $\mathcal{G}'$ in place of $\mathcal{G}$ yields the same success probability for the adversary up to a probability of $\frac{4\ell s^2}{p}$, and, in Claim 4.6, we show that when using $\mathcal{G}'$ the adversary will never succeed in the experiment. Together these proving multi-extraction of the scheme.

Claim 4.5.

$$\Pr\left[\begin{array}{l}\left(\begin{array}{l}\mathsf{Com.Verify}^{\mathcal{G}}(\mathsf{crs},\mathsf{c},M,\alpha,\mathsf{pf})=1\\ \wedge\ Mx_i\neq\alpha\end{array}\right)\\ \vee\\ \left(\begin{array}{l}\exists i_1,i_2\in[n]:\\ \mathsf{c}_{i_1}=\mathsf{c}_{i_2}\ \wedge\ x_{i_1}\neq x_{i_2}\end{array}\right)\end{array}\middle|\begin{array}{l}\mathcal{G}\leftarrow\mathsf{GGM}(1^\lambda)\\ \mathsf{crs}\leftarrow\mathsf{Com.Setup}^{\mathcal{G}}(1^\lambda,\ell,q)\\ \mathsf{aux}_0:=\bot\\ \forall i\in[n]:\left(\begin{array}{l}(\mathsf{c}_i,\mathsf{aux}_i)\leftarrow\mathcal{A}_i^{\mathcal{G}}(\mathsf{crs},\mathsf{aux}_{i-1})\\ x_i\leftarrow\mathsf{Ext}^{\mathcal{G}}(\mathcal{A}_1,\ldots,\mathcal{A}_i,\mathsf{crs})\end{array}\right)\\ (i,M,\alpha)\leftarrow\mathcal{A}'^{\mathcal{G}}(\mathsf{crs})\end{array}\right]$$

$$-\Pr\left[\begin{array}{l}\left(\begin{array}{l}\mathsf{Com.Verify}^{\mathcal{G}'}(\mathsf{crs},\mathsf{c},M,\alpha,\mathsf{pf})=1\\ \wedge\ Mx_i\neq\alpha\end{array}\right)\\ \vee\\ \left(\begin{array}{l}\exists i_1,i_2\in[n]:\\ \mathsf{c}_{i_1}=\mathsf{c}_{i_2}\ \wedge\ x_{i_1}\neq x_{i_2}\end{array}\right)\end{array}\middle|\begin{array}{l}\mathsf{crs}\leftarrow\mathsf{Com.Setup}^{\mathcal{G}'}(1^\lambda,\ell,q)\\ \mathsf{aux}_0:=\bot\\ \forall i\in[n]:\left(\begin{array}{l}(\mathsf{c}_i,\mathsf{aux}_i)\leftarrow\mathcal{A}_i^{\mathcal{G}'}(\mathsf{crs},\mathsf{aux}_{i-1})\\ x_i\leftarrow\mathsf{Ext}^{\mathcal{G}'}(\mathcal{A}_1,\ldots,\mathcal{A}_i,\mathsf{crs})\end{array}\right)\\ (i,M,\alpha)\leftarrow\mathcal{A}'^{\mathcal{G}'}(\mathsf{crs})\end{array}\right]$$

$$\leq\frac{4\ell s^2}{p}\,.$$

Proof. The view of all parties in the experiments only differs if there exist two polynomials f, f' added by $\mathcal{G}'$ to the list such that f and f' are not equivalent, and yet $\mathcal{G}$ gives them the same label.

Fix some f, f' in the lists of $\mathcal{G}'$. Since f, f' represent the values in $\mathcal{G}$ up to plugging in of the variables, this occurs only if

$$f(\beta, z_1, \ldots, z_q) = f'(\beta, z_1, \ldots, z_q)\,,$$

which, by the polynomial identity lemma, occurs with probability $\max\{\mathsf{deg}(f), \mathsf{deg}(f')\}/p$, where deg denotes the total degree of the polynomial.

Observe that polynomials in Λ_1 have degree at most ℓ, polynomials in Λ_2 have degree at most $2\ell+1$, and polynomials in Λ_T have degree at most $3\ell+1$. Thus, for f, f', this probability is at most $3\ell+1$.

Finally, the number of equality queries done to $\mathcal{G}'$ is bounded as follows: $\ell + q\cdot(2\ell-1)$ by the setup algorithm, s queries made by $\mathcal{A}$, none made by Ext (as it only makes the same queries as the stages in $\mathcal{A}$), and one the verification test. Together, the total number of queries is bound by $3q\ell + s$. Thus, the probability that there exists a pair of polynomials whose description differs in $\mathcal{G}'$ but is the same for $\mathcal{G}$ is bounded by

$$\begin{aligned}\binom{3q\ell+s}{2}\cdot\frac{3\ell+1}{p} &< \frac{27\ell^3q^2+9\ell^2q^2}{2p}+\frac{18\ell^2qs+6\ell qs}{2p}+\frac{4\ell s^2+s^2}{2p}\\ &< \frac{15\ell^3q^2}{p}+\frac{10\ell^2qs}{p}+\frac{3\ell s^2}{p}\\ &< \frac{20\ell^3q^2s}{p}+\frac{3\ell s^2}{p}\\ &< \frac{\ell s^2}{p}+\frac{3\ell s^2}{p}=\frac{4\ell s^2}{p}\,.\end{aligned}$$

The inequalities above hold since $20\ell^2q^2 < s$, and using the mild assumption that $\ell > 3$. □

Finally, we show that the event E' can never occur:

Claim 4.6.

$$\Pr\left[\begin{array}{l}\left(\begin{array}{l}\mathsf{Com.Verify}^{\mathcal{G}'}(\mathsf{crs},\mathsf{c},M,\alpha,\mathsf{pf})=1\\ \wedge\ Mx_i\neq\alpha\end{array}\right)\\ \vee\\ \left(\begin{array}{l}\exists i_1,i_2\in[n]:\\ \mathsf{c}_{i_1}=\mathsf{c}_{i_2}\ \wedge\ x_{i_1}\neq x_{i_2}\end{array}\right)\end{array}\;\middle|\;\begin{array}{l}\mathsf{crs}\leftarrow\mathsf{Com.Setup}^{\mathcal{G}'}(1^\lambda,\ell,q)\\ \mathsf{aux}_0:=\perp\\ \forall i\in[n]:\left(\begin{array}{l}(\mathsf{c}_i,\mathsf{aux}_i)\leftarrow\mathcal{A}_i^{\mathcal{G}'}(\mathsf{crs},\mathsf{aux}_{i-1})\\ x_i\leftarrow\mathsf{Ext}^{\mathcal{G}'}(\mathcal{A}_1,\ldots,\mathcal{A}_i,\mathsf{crs})\end{array}\right)\\ (i,M,\alpha)\leftarrow\mathcal{A}'^{\mathcal{G}'}(\mathsf{crs})\end{array}\right]=0\,.$$

Proof. We consider each part in the event separately and show that they both cannot hold when interacting with $\mathcal{G}'$:

- "$\mathsf{Com.Verify}^{\mathcal{G}'}(\mathsf{crs},\mathsf{c}_i,M,\alpha,\mathsf{pf})=1\wedge\ Mx_i\neq\alpha$": Consider $(i^*,M,\alpha,\mathsf{pf})$ chosen by $\mathcal{A}'$ and the matching commitment c_{i^*}. Let f be the polynomial matching the label c_{i^*} and h be the polynomial matching the label pf. Suppose that $\mathsf{Com.Verify}^{\mathcal{G}'}(\mathsf{crs},\mathsf{c},M,\alpha,\mathsf{pf})=1$. This implies that the following two polynomials F_1,F_2 get the same label:

  $$F_1(X,Z_1,\ldots,Z_q):=f(X,Z_1,\ldots,Z_q)\cdot\sum_{i\in[q]}\sum_{j\in[\ell]}M_{i,j}\cdot Z_iX^{\ell+1-j}$$
  $$F_2(X,Z_1,\ldots,Z_q):=\sum_{i\in[q]}\alpha_i\cdot Z_iX^{\ell+1}+h(X,Z_1,\ldots,Z_q)\,.$$

  By construction, the only way that F_1 and F_2 can share a label is if $F_1\equiv F_2$, and so we assume this to be the case.

  Observe that by construction, any polynomial in Λ_1 (which includes c_{i^*}) is of degree ℓ in X, and any polynomial in Λ_2 has monomials only of the form Z_iX^j for $i\in[q]$ and $j\in[2\ell]\setminus\{\ell+1\}$. As a result, f and h have the following forms:

  $$f(X)=\gamma_0+\sum_{j\in[\ell]}^{\ell}\gamma_jX^j\qquad\textit{and}\qquad h(X,Z_1,\ldots,Z_q)=\sigma+\sum_{i\in[q]}\sum_{j\in[2\ell]\setminus\{\ell+1\}}\sigma_{i,j}Z_iX^j\,,$$

  for some integers γ_j, σ and, $\sigma_{i,j}$.

  Thus, for every $i\in[q]$, the coefficient of $Z_iX^{\ell+1}$ is $\sum_{j\in\ell}M_{i,j}\gamma_j$ for F_1 and α_i for F_2. Since $F_1\equiv F_2$, we get that $\sum_{j\in\ell}M_{i,j}\gamma_j=\alpha_i \mod p$, or, in other words, the vector $x_{i^*}=(\gamma'_1,\ldots,\gamma'_\ell)$ where $\gamma'_i=\gamma i \mod p$ has $Mx_{i^*}=\alpha$. Finally, we observe that Ext and $\mathcal{G}'$ both have exactly the same definitions and view of $\Lambda_1,\Lambda_2,\Lambda_T$, and so Ext will derive the same vector x_{i^*}.

- "$\exists i_1,i_2\in[n]:\mathsf{c}_{i_1}=\mathsf{c}_{i_2}\ \wedge\ x_{i_1}\neq x_{i_2}$": by construction, there is a single polynomial f for every c such that $(\mathsf{c},f)\in\Lambda_1$ (in both the extractor and $\mathcal{G}'$). Thus, for any i_1,i_2 with $\mathsf{c}_{i_1}=\mathsf{c}_{i_2}$, the extractor will choose the same polynomial f from which to derive x_{i_1} and x_{i_2}. Thus, $x_{i_1}=x_{i_2}$.

□

□

5 SNARGs with two group elements

In this section we show that there exist SNARGs for NP in the GGM + ROM with only two group elements by combining linear PCPs with a linear map commitment scheme. We present two different SNARGs based on two different PCPs, where the main difference between the SNARGs is the soundness error.

Our first SNARG has error roughly $O(s/n^{\mathrm{polylog}\,n}+\mathrm{poly}(n,s)/2^\lambda)$, and so is negligible in λ when $\lambda \leq \mathrm{polylog}\, n$:

Theorem 5.1. *Let $c > 0$ be a constant. The NP-complete problem $\mathsf{GapMWSP}_m$ for $m = \log^{c+2} n$ has a SNARK in the bilinear generic group model with size 2^λ with the following parameters for inputs $(A, b, d) \in \mathbb{F}^{\ell\times n} \times \mathbb{F}^\ell \times \mathbb{N}$:*

- *Argument size: $2\ \mathbb{G}_1$.*
- *Completeness error: $\frac{24n^5\lambda^2}{2^\lambda\,\mathrm{loglog}(n)^2}$.*
- *Soundness/knowledge errors: $\frac{s+1}{n^{\log^c n}}+\frac{5ns^2}{2^\lambda}$ with extraction time $O(n+s)$ against s-size adversaries.*
- *Reference string length: $n\ \mathbb{G}_1 + 2n - 1\ \mathbb{G}_2$.*
- *Running times: proving $O(\ell n + n^2 + \lambda\,\mathrm{loglog}(n)^2)$, verification $O(\ell n)$.*

The above error bounds hold for $n > 170$ and $s > 20n^2$.

The following theorem improves upon the soundness error of Theorem 5.1 but requires the additional assumption of the unique games conjecture:

Theorem 5.2. *Assume the unique games conjecture. There is a SNARG for NP with in the bilinear generic group model with size 2^λ with the following parameters for inputs of size n:*

- *Argument size: $2\ \mathbb{G}_1$.*
- *Completeness error: $\frac{\lambda^4\mathrm{poly}(n)}{2^\lambda}$.*
- *Soundness error: $\frac{s^2\mathrm{poly}(n)}{2^\lambda}$.*
- *Reference string length:* $\mathrm{poly}(n)$ *$\mathbb{G}_1$ and $\mathbb{G}_2$ elements.*
- *Running times: proving* $\mathrm{poly}(n,\lambda)$*, verification* $\mathrm{poly}(n)$*.*

The above error bounds hold for $n > 671$, $s > 20n^2$ and, $ns > 8(25 + \lambda)$.

This section is organized as follows: (1) in Section 5.1 we define linear PCPs and describe the PCPs used in this paper, as well as a generic way to transform PCPs with a large accepting set into one with a small accepting set, (2) in Section 5.2 we show how to combine a linear PCP with a linear map commitment to construct a SNARK, and (3) in Section 5.3 we put these tools together to prove Theorems 5.1 and 5.2.

5.1 Linear PCPs

In this section we define linear PCPs. We define them in a slightly non-standard way as we will specifically refer to the accepting set of the PCP verifier.

Definition 5.3. *A (non-adaptive) linear PCP over a finite field $\mathbb{F}$ for a relation $\mathcal{R}$ is a tuple of algorithms $(\mathbf{P}, \mathbf{V})$ where the prover outputs a string $\pi \in \mathbb{F}^\ell$, and the verifier outputs a matrix $M \in \mathbb{F}^{\ell\times q}$ and a set $S \subseteq \mathbb{F}^q$ (for which membership is efficiently decidable) such that the following properties hold:*

- Completeness. *The PCP has completeness error α if for every $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$,*

$$\Pr\left[(M\pi) \in S \,\middle|\, \begin{array}{r} \pi = \mathbf{P}(\mathbb{x}, \mathbb{w}) \\ \rho \leftarrow \{0,1\}^{\mathsf{r}} \\ (M, S) \leftarrow \mathbf{V}(\mathbb{x}, \rho) \end{array}\right] \geq 1 - \alpha\,.$$

- Soundness. *The PCP has soundness error β if for every $\mathbb{x} \notin L(\mathcal{R})$ and $\pi \in \mathbb{F}^{\ell}$:*

$$\Pr\left[(M\pi) \in S \,\middle|\, \begin{array}{r} \rho \leftarrow \{0,1\}^{\mathsf{r}} \\ (M, S) \leftarrow \mathbf{V}(\mathbb{x}, \rho) \end{array}\right] \leq \beta\,.$$

- *Knowledge soundness. The PCP has knowledge soundness error κ with extractor running time* et *if there exists a* et*-time extractor* Ext *such that for every instance $\mathbb{x}$ and proof π, if*

$$\Pr\left[(M\pi) \in S \,\middle|\, \begin{array}{r} \rho \leftarrow \{0,1\}^{\mathsf{r}} \\ (M, S) \leftarrow \mathbf{V}(\mathbb{x}, \rho) \end{array}\right] \geq \kappa\,,$$

  then $(\mathbb{x}, \mathsf{Ext}(\mathbb{x}, \pi)) \in \mathcal{R}$.

The proving time pt is the time to run $\pi = \mathbf{P}(\mathbb{x}, \mathbb{w})$, and the verification time vt is the combined time to generate M, S and to test membership in S. We say that the accepting set size of the PCP is s if for every $\mathbb{x}$ and ρ: $|S| \leq s$. In this paper we consider PCPs with an injective map $\varphi_S \colon S \to [|S|]$. We let t_φ be the time to compute φ_S and φ_S^{-1}.

The following claim shows that any PCP can be transformed into one with small accepting set size at a high cost to the completeness error:

Claim 5.4. *Suppose $\mathcal{R}$ has a linear PCP with completeness error α, randomness complexity* r*, and accepting set size s. Then $\mathcal{R}$ has a linear PCP with completeness error $\alpha + \frac{s-1}{s}$, randomness complexity $\mathsf{r} + \log s$, and accepting set size 1. The soundness error, knowledge soundness error, and length of the PCP are preserved, and the running times are preserved up to an additive factor of $O(\log s)$.*

Proof. Let $(\mathbf{P}, \mathbf{V})$ be the PCP for $\mathcal{R}$. We construct a new PCP verifier $\mathbf{V}'$ that interacts with the same honest prover $\mathbf{P}$. The new verifier $\mathbf{V}'$, given $(\mathbb{x}, (\rho, z))$ where $z \in [s]$ computes $(M, S) \leftarrow \mathbf{V}(\mathbb{x}, \rho)$, and outputs M and accepting set $S' = \{\varphi_S^{-1}(z)\}$.

If $\mathbb{x}$ is in the relation, then the verifier rejects either when it rejects in the original PCP (with probability at most α) or if it chooses an incorrect answer from S, which happens with probability $\frac{s-1}{s}$. Thus, the completeness error is $\alpha + \frac{s-1}{s}$. Soundness does not degrade as M comes from the exact same distribution in $\mathbf{V}$ and $\mathbf{V}'$ and since $S' \subseteq S$. □

5.1.1 PCP and linear PCP constructions

We describe the PCPs used to derive Theorems 5.1 and 5.2. We begin by defining an NP-complete language:

Definition 5.5. *Fix a finite field $\mathbb{F}$. For an approximation factor m, and instance of* $\mathsf{GapMWSP}_m$ *over $\mathbb{F}$ is a triple (A, b, d) where $A \in \mathbb{F}^{\ell \times n}$, $b \in \mathbb{F}^{\ell}$, and $d \in \mathbb{N}$ such that:*
- *(A, b, d) is a YES instance if there exists $x \in \mathbb{F}^n$ such that $Ax = b$ and x has Hamming weight at most d.*

- *(A, b, d) is a NO instance if every solution $x \in \mathbb{F}^n$ where $Ax = b$ has Hamming weight at least $m \cdot d$.*

It is known that for certain values of m, the problem $\mathsf{GapMWSP}_m$ is NP-hard:

Lemma 5.6 ([BIOW20], Lemma 4.2)**.** *For any constant $c > 0$, $m = \log^c n$ and finite field $\mathbb{F}$ where $|\mathbb{F}| = \mathrm{poly}(n)$ there exists a Karp–Levin reduction*[12] *from SAT to $\mathsf{GapMWSP}_m$, where the reduction algorithm takes a target field $\mathbb{F}$ as an explicit input and outputs and instance (A, b, d) over $\mathbb{F}$ in time $\mathrm{poly}(n, \log|\mathbb{F}|)$.*

[BIOW20] further construct a PCP for $\mathsf{GapMWSP}_m$ with accepting size 1:

Theorem 5.7 ([BIOW20], Corollary 4.6 reformulated)**.** *Let $\alpha > 0$ be a completeness parameter and $m > 0$ be an approximation parameter. There is a 1-query linear PCP for the $\mathsf{GapMWSP}_m$ problem with accepting set size $s = 1$,*[13] *completeness error α and soundness error $1/|\mathbb{F}| + e^{-\alpha m}$. The prover time is $O(n)$ and the verifier time is $O(n\ell)$.*

In order to get better parameters, we also use a PCP that is proven under the unique games conjecture. A *unique game* is a tuple $(V, E, \Sigma, f_{(x,y)\in E})$ where $V \subseteq \Sigma$, $E \subseteq V \times V$, and for every $(x, y) \in E$, $f_{x,y} \colon \Sigma \to \Sigma$ is a permutation. The value of assignment $A \colon V \to \Sigma$ is the fraction of pairs $(x, y) \in E$ where $A(y) = f_{x,y}(A(x))$, and the value of the unique game is the value of the optimum assignment. The unique games conjecture is as follows:

Conjecture 5.8. *For every $\gamma > 0$ there is $\sigma = \sigma(\gamma)$ such that it is NP-hard to distinguish unique games of value $\geq 1 - \gamma$ from unique games of value γ, even when restricted to instances where $|\Sigma| \leq \sigma$ and where the constraint graph is bipartite.*

We can now state the PCP in question:

Theorem 5.9 ([ST06], Theorem 29)**.** *Assuming the unique games conjecture, for every $k \geq 2$ there exists a PCP for* NP *with completeness error 2^{-2^k}, soundness error $(2^k + 2)/2^{2^k}$, proof length $\mathrm{poly}(n)$, query complexity $2^k - 1$, and accepting set size 2^k.*

Proof. Theorem 29 of [ST06] states that if the unique games conjecture holds then for every $\delta > 0$ and $k \geq 2$ we have there exists a PCP for NP with completeness error δ, soundness error $(2^k + 1)/2^{2^k} + \delta$, proof length $\mathrm{poly}(n)$, and query complexity $2^k - 1$. We use $\delta = 2^{-2^k}$ and observe that the accepting set is of size 2^k.

All queries their verifier does are in the context of a δ-noisy hypergraph test (see Definition 26 of [ST06]). This hypergraph test receives $2^k - 1$ functions $(g^{(s)} : \{0,1\}^n \to \{0,1\})_{s\subset[k], s\neq\emptyset}$. The verifier samples uniform k bitstrings $x_1, \ldots, x_k$ and $2^k - 1$ sparse bitstrings $(\eta^{(s)})_{s\subset[k], s\neq\emptyset}$. The verifier queries $g^{(s)}(\eta^{(s)} + \sum_{i\in s} x_i)$. The verifier accepts if and only if for all $\emptyset \neq s \subset [k]$ we have

$$\sum_{i\in s} g^{(\{i\})}(x_i + \eta^{(\{i\})}) = g^{(s)}(\eta^{(s)} + \sum_{i\in s} x_i).$$

Notice, that on the right-hand side of the equation is just response to one term and the left-hand side of these equantions are only ever terms $g^{(\{i\})}(x_i + \eta^{(\{i\})})$ for $i \in [k]$. Therefore, the respone to these k uniquely identify one accepting response and the accepting set is size $\leq 2^k$. $\square$

[12] Here, by Karp–Levin reduction from $\mathcal{R}$ to $\mathcal{R}'$, we mean that there exist a pair of efficiently computable functions f, g so that if $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$ then $(f(\mathbb{x}), g(\mathbb{x}, \mathbb{w})) \in \mathcal{R}'$, and if $\mathbb{x} \notin L(\mathcal{R})$ then $f(\mathbb{x}) \notin L(\mathcal{R}')$.

[13] [BIOW20] do not explicitly say that their PCP has accepting set size 1, but this follows by their construction.

By applying Claim 5.4 to Theorem 5.9 and setting $\lambda = 2^k$ we get the following PCP with negligible soundness error and accepting set size 1:

Corollary 5.10. *Assuming the unique games conjecture, for every $\lambda > 8$, there exists a PCP for* NP *with completeness error $1-1/2\lambda$, soundness error $(\lambda+2)/2^\lambda$, proof length* $\text{poly}(n)$*, query complexity $\lambda - 1$, and accepting set size* 1.

5.2 SNARGs from PCPs with small accepting set

Given a linear PCP (ideally with small accepting set) and a linear map commitment, we construct a SNARG. This construction is similar to those in [LM19; CGKY25], except that we save in proof length by considering the accepting set size, and that we provide a technique for reducing completeness error at no runtime costs to the verifier.

Lemma 5.11. *Let $\mathbb{F}$ be a finite field and $t \in \mathbb{N}$ be a completeness error amplification parameter. Consider the following:*

- *A linear PCP for a relation $\mathcal{R}$ over $\mathbb{F}$ with completeness error α, soundness error β, knowledge soundness error κ, accepting set size s, mapping time t_φ, proving time pt and verifier time vt, and extraction time et.*
- *A linear map commitment over $\mathbb{F}$ with multi-extraction ε, committing time $\mathsf{t_{cc}}$, opening time $\mathsf{t_{co}}$, verification time $\mathsf{t_{cv}}$, and extraction time $\mathsf{t_{ext}}$.*

Then there is SNARK in the ROM for $\mathcal{R}$ with:
- *Argument size:* $s_\mathsf{c} + s_\mathsf{pf} + \log s$
- *Completeness error:* $\alpha^t + t^2 \cdot \varepsilon(\lambda, \ell', 1, 1, O(\mathsf{pt} + \mathsf{t_{cc}} + \mathsf{t_{co}})$
- *Soundness error:* $(s+1) \cdot \beta + \varepsilon(\lambda, \ell', q, s+1, s + \mathsf{vt} + \mathsf{t}_\varphi)$ *against s-size adversaries*
- *Knowledge error:* $(s+1) \cdot \kappa + \varepsilon(\lambda, \ell', q, s+1, s + \mathsf{vt} + \mathsf{t}_\varphi)$ *against s-size adversaries with extraction time* $O(\mathsf{pt} + \mathsf{et} + \mathsf{t_{ext}}(s))$
- *Random oracle queries:* 1
- *Proving time* $O(\mathsf{pt} + \mathsf{t}_\varphi + t \cdot (\mathsf{t_{cc}} + \mathsf{vt}) + \mathsf{t_{co}})$
- *Verification time:* $O(\mathsf{vt} + \mathsf{t}_\varphi + \mathsf{t_{cv}})$

Above, $\ell' = \ell + \lceil \log_{|\mathbb{F}|} t \rceil$.

Remark 5.12. When using a standard PCP (i.e., a linear PCP where the verifier makes only point queries), the linear map commitment in the above theorem can be relaxed to a subvector commitment scheme.

Remark 5.13. If, as is the case in this paper, the linear map commitment is secure in the generic group model, and the multi-extraction error is a function of GGM queries rather than circuit size (see Remark 4.2), then the errors in Theorem 5.1 can be improved as follows:
- Completeness error: $\alpha^t + t^2 \cdot \varepsilon(\lambda, \ell', 1, 1, 2 \cdot \mathsf{t_{cc}} + \mathsf{t_{co}})$
- Soundness error: $(s+1) \cdot \beta + \varepsilon(\lambda, \ell', q, s+1, s + \mathsf{t_{cv}})$
- Knowledge error: $(s+1) \cdot \kappa + \varepsilon(\lambda, \ell', q, s+1, s + \mathsf{t_{cv}})$

This follows from the fact that all of the reductions of adversaries made by our proofs do not make any GGM queries unless they explicitly call one of the algorithms of the commitment scheme.

Proof. Let $\ell' = \ell + \lceil \log_{|\mathbb{F}|} t \rceil$. For simplicity, we assume that the linear PCP verifier receiver λ bits of randomness. The scheme can be easily adapted if this is not the case by the random oracle outputting the appropriate number of bits. We begin by describing the construction:

Construction 5.14. The SNARG $(\mathcal{S}, \mathcal{P}, \mathcal{V})$ is as follows:

- $\mathcal{S}(1^n, 1^\lambda)$: Output $\mathsf{crs} \leftarrow \mathsf{Com.Setup}(1^\lambda, \ell', q)$.
- $\mathcal{P}^{\mathcal{O}}(\mathsf{crs}, \mathbb{x}, \mathbb{w})$:

```
1. $\pi \leftarrow \mathbf{P}(\mathbb{x}, \mathbb{w})$.
2. For $i = 1$ to $t$:
   (a) Set $\pi' = (\pi \| i)$ (where we treat $i$ as $\lceil \log_{|\mathbb{F}|} t \rceil$ field elements).
   (b) $(\mathsf{c}, \mathsf{aux}) \leftarrow \mathsf{Com.Commit}(\mathsf{crs}, \pi')$.
   (c) $\rho \leftarrow \mathcal{O}(\mathbb{x}, \mathsf{c})$.
   (d) $(M, S) \leftarrow \mathbf{V}(\mathbb{x}, \rho)$.
   (e) If $(M\pi) \in S$ then:
       i. Let $M' \in \mathbb{F}^{\ell' \times q}$ be defined as $M'[j,k] = \begin{cases} M[j,k] & j \in [\ell] \\ 0 & j > \ell \end{cases}$.
       ii. $(\alpha, \mathsf{pf}) \leftarrow \mathsf{Com.Open}(\mathsf{crs}, \mathsf{c}, \mathsf{aux}, M')$.
       iii. $z = \varphi_S(\alpha)$.
       iv. Output $\Pi = (\mathsf{c}, z, \mathsf{pf})$
3. If this point has been reached, output "FAILED".
```

- $\mathcal{V}^{\mathcal{O}}(\mathsf{crs}, \mathbb{x}, \Pi = (\mathsf{c}, z, \mathsf{pf}))$:

```
1. Compute $\rho \leftarrow \mathcal{O}(\mathbb{x}, \mathsf{c})$.
2. Let $(M, S) \leftarrow \mathbf{V}(\mathbb{x}, \rho)$, let $\alpha = \varphi_S^{-1}(z)$, and set $M' \in \mathbb{F}^{\ell' \times q}$ is defined as in the honest prover
   algorithm.
3. Accept if $\alpha \in S$ and $\mathsf{Com.Verify}(\mathsf{crs}, \mathsf{c}, M', \alpha, \mathsf{pf}) = 1$ and reject otherwise.
```

We first show completeness and then soundness and knowledge soundness.

Claim 5.15. *The SNARK has completeness error at most $\alpha^t + t^2 \cdot \varepsilon(\lambda, \ell', 1, 1, O(\mathsf{pt} + \mathsf{t_{cc}} + \mathsf{t_{co}}))$.*

Proof. Fix $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$ and $\pi = \mathbf{P}(\mathbb{x}, \mathbb{w})$. For iteration i, we denote $(\mathsf{c}_i, \mathsf{aux}_i) = \mathsf{Com.Commit}(\mathsf{crs}, (\pi||i))$, $\rho_i \leftarrow \mathcal{O}(\mathbb{x}, \mathsf{c}_i)$, and $(M_i, S_i) \leftarrow \mathbf{V}(\mathbb{x}, \rho_i)$. Observe that ρ_i and ρ_j are independently distributed uniformly random strings whenever $\mathsf{c}_i \neq \mathsf{c}_j$. Thus, by completeness of the linear PCP and correctness of the linear map commitment, if for every $i \in [t]$, c_i is unique, the verifier rejects with probability at most α^t. In other words:

$$\begin{aligned}&\Pr[\text{Verifier Rejects}] \\ &\quad \leq \Pr\left[\exists i, j \in [t],\ i \neq j \ \wedge\ \mathsf{c}_i = \mathsf{c}_j \;\middle|\; \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{Com.Setup}(1^\lambda, \ell', q) \\ \forall i \in [t],\ (\mathsf{c}_i, \mathsf{aux}_i) = \mathsf{Com.Commit}(\mathsf{crs}, (\pi||i)) \end{array}\right] + \alpha^t\end{aligned}$$

All that remains is to show that:

$$\Pr\left[\exists i, j \in [t],\ i \neq j \ \wedge\ \mathsf{c}_i = \mathsf{c}_j \;\middle|\; \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{Com.Setup}(1^\lambda, \ell', q) \\ \forall i \in [t],\ (\mathsf{c}_i, \mathsf{aux}_i) = \mathsf{Com.Commit}(\mathsf{crs}, (\pi||i)) \end{array}\right] < \varepsilon\,.$$

We show that this holds by the function binding property of the linear map commitment. Consider the following PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}')$ to function binding of the LMC. Let Ext be the extractor of the commitment scheme:

- $\mathcal{A}_1(\mathsf{crs}, \mathsf{aux} = \perp)$:

1. Compute $\pi = \mathbf{P}(\mathbb{x}, \mathbb{w})$.
2. Choose $(i,j) \leftarrow \binom{[t]}{2}$.
3. For $a \in \{i,j\}$, let $\pi_a = (\pi \| a)$ (where we treat a as $\lceil \log_{|\mathbb{F}|} t \rceil$ field elements) and compute $(\mathsf{c}_a, \mathsf{aux}_a) = \mathsf{Com.Commit}(\mathsf{crs}, \pi_a)$.
4. If $\mathsf{c}_i = \mathsf{c}_j$ then output $(\mathsf{c}_i, \mathsf{aux}' = (i, j, \mathsf{c}_i, \mathsf{aux}_i, \mathsf{aux}_j))$. Otherwise output $\perp$.

- $\mathcal{A}'(\mathsf{crs}, \mathsf{aux}' = (i, j, \mathsf{c}_i, \mathsf{aux}_i, \mathsf{aux}_j))$:

1. Choose $a \leftarrow \{i,j\}$ uniformly at random.
2. Let $k \in [\ell']$ be such that $\pi_i[k] \neq \pi_j[k]$. One such k exists since the last $\lceil \log_{|\mathbb{F}|} t \rceil$ symbols in π_i and π_j encode i and j respectively, and $i \neq j$.
3. Let $M \in \mathbb{F}^{\ell' \times 1}$ be the matrix that is equal to 0 everywhere, except the k-th row, which is equal to 1.
4. Compute $\mathsf{pf} = \mathsf{Com.Open}(\mathsf{crs}, \mathsf{c}_a, \mathsf{aux}_a, M)$.
5. Output $(M, \alpha, \mathsf{pf}) = (M, \pi_a[k], \mathsf{pf})$.

By function binding of the LMC,

$$\Pr\left[\begin{array}{l|l} \mathsf{Com.Verify}(\mathsf{crs}, \mathsf{c}, M, \alpha, \mathsf{pf}) = 1, & \mathsf{crs} \leftarrow \mathsf{Com.Setup}(1^\lambda, \ell', q) \\ (Mx) \neq \alpha & (\mathsf{c}, \mathsf{aux}) \leftarrow \mathcal{A}_1(\mathsf{crs}) \\ & x \leftarrow \mathsf{Ext}(\mathcal{A}_1, \mathsf{crs}) \\ & (M, \alpha, \mathsf{pf}) \leftarrow \mathcal{A}'(\mathsf{crs}, \mathsf{aux}) \end{array}\right] \leq \varepsilon \,.$$

Observe that, by construction, whenever there is $\mathsf{c}_i = \mathsf{c}_j$, and we guess these ndices correctly, it is always the case that: (1) by completeness of the linear map commitment, $\mathsf{Com.Verify}(\mathsf{crs}, \mathsf{c}, M, \alpha, \mathsf{pf}) = 1$, and (2) with probability $1/\binom{t}{2}$ for at least one choice of a it holds that $(Mx) \neq M\pi_a$ for $x \leftarrow \mathcal{A}'(\mathsf{crs}, \mathsf{aux})$. Therefore we conclude that:

$$\begin{aligned} &\Pr\left[\exists i,j \in [t],\ i \neq j \ \wedge\ \mathsf{c}_i = \mathsf{c}_j \,\middle|\, \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{Com.Setup}(1^\lambda, \ell', q) \\ \forall i \in [t],\ (\mathsf{c}_i, \mathsf{aux}_i) = \mathsf{Com.Commit}(\mathsf{crs}, (\pi || i)) \end{array}\right] \\ &\leq \binom{t}{2} \cdot \Pr\left[\begin{array}{l|l} \mathsf{Com.Verify}(\mathsf{crs}, \mathsf{c}, M, \alpha, \mathsf{pf}) = 1, & \mathsf{crs} \leftarrow \mathsf{Com.Setup}(1^\lambda, \ell', q) \\ (Mx) \neq \alpha & (\mathsf{c}, \mathsf{aux}) \leftarrow \mathcal{A}_1(\mathsf{crs}) \\ & x \leftarrow \mathsf{Ext}(\mathcal{A}_1, \mathsf{crs}) \\ & (M, \alpha, \mathsf{pf}) \leftarrow \mathcal{A}'(\mathsf{crs}, \mathsf{aux}) \end{array}\right] \\ &\leq t^2 \cdot \varepsilon(\lambda, \ell', 1, 1, O(\mathsf{pt} + \mathsf{t_{cc}} + \mathsf{t_{co}}))\,. \end{aligned}$$

□

Claim 5.16. *The SNARK has soundness error at most $(s+1) \cdot \beta + \varepsilon(\lambda, \ell', q, s+1, s + \mathsf{vt} + \mathsf{t}_\varphi)$ and knowledge error at most $(s+1) \cdot \kappa + \varepsilon(\lambda, \ell', q, s+1, s + \mathsf{vt} + \mathsf{t}_\varphi)$ against s-size adversaries. The extraction time is $O(\mathsf{pt} + \mathsf{et} + \mathsf{t_{ext}}(s))$.*

Proof. We argue knowledge soundness. Soundness follows by an identical argument while removing the need for extraction. For an m-query prover $\mathbf{P}$ (so that $m \leq s$), let q_i be its i-th to the random oracle query and aux_i be its state immediately prior to making the i-th query. We use the notation $\mathbf{P} = (\mathbf{P}_1, \ldots, \mathbf{P}_{m+1}, \mathbf{P}')$ as follows:

- For $i \in [m]$: $\mathbf{P}_i$ is the algorithm that, on input crs, aux_{i-1}, and a query answer ρ_{i-1} uses this information (by continuing to run $\mathbf{P}$) to output (q_i, aux_i).

- $\mathbf{P}_{m+1}$ is the algorithm that, on input $\mathsf{crs}, \mathsf{aux}_m, \rho_m$ runs keeps running $\mathbf{P}$ until it outputs $(\mathbb{x}, \Pi = (\mathsf{c}, z, \mathsf{pf}))$ and outputs $\mathsf{c}_{m+1} = \mathsf{c}$ and auxiliary state $\mathsf{aux}_{m+1} = (\mathbb{x}, z, \mathsf{pf})$.
- $\mathbf{P}'$ receives crs and $\mathsf{aux}_{m+1} = (\mathbb{x}, z, \mathsf{pf})$, computes $(M, S) \leftarrow \mathbf{V}(\mathbb{x}, \mathcal{O}(\mathbb{x}, \mathsf{c}))$ and outputs $(i, M, \varphi_S^{-1}(z), \mathsf{pf})$

We can now present the SNARK extractor $\mathcal{E}$:

Construction 5.17. Let Ext be the extractor of the linear map commitment. Then $\mathcal{E}^{\mathcal{O}}$ on inputs $\mathbf{P} = (\mathbf{P}_1, \ldots, \mathbf{P}_{m+1}, \mathbf{P}')$ and crs and runs as follows:

1. Run $(\mathbb{x}, \Pi = (\mathsf{c}, z, \mathsf{pf})) \leftarrow \mathbf{P}^{\mathcal{O}}(\mathsf{crs})$ and let i^* be the query index where $\mathbf{P}^{\mathcal{O}}$ queries $(\mathbb{x}, \mathsf{c})$, and $i^* = m + 1$ if the query was not made.
2. Compute $\pi' \leftarrow \mathsf{Ext}(\mathbf{P}_1^{\mathcal{O}}, \ldots, \mathbf{P}_{i^*}^{\mathcal{O}}, \mathsf{crs}) \in \mathbb{F}^{\ell'}$. Let π be the first ℓ symbols of π'.[14]
3. Derive $\mathbb{w} \leftarrow \mathbf{E}(\mathbb{x}, \pi)$.
4. Output $\mathbb{w}$.

Using our notation and the above extractor:

$$\Pr\left[\begin{array}{l|l} |\mathbb{x}| \leq n, & \mathcal{O} \leftarrow \mathsf{RO}(1^\lambda) \\ (\mathbb{x}, \mathbb{w}) \notin \mathcal{R}, & \mathsf{crs} \leftarrow \mathcal{S}^{\mathcal{O}}(1^n, 1^\lambda) \\ \mathcal{V}'(\mathbb{x}, \mathsf{c}, \rho, (z, \mathsf{pf})) = 1 & (\mathbb{x}, \Pi) \leftarrow \mathbf{P}^{\mathcal{O}}(\mathsf{crs}) \\ & \mathbb{w} \leftarrow \mathcal{E}^{\mathcal{O}}(\mathbf{P}, \mathsf{crs}) \end{array}\right]$$

$$= \Pr\left[\begin{array}{l|l} & \mathcal{O} \leftarrow \mathsf{RO}(1^\lambda) \\ & \mathsf{crs} \leftarrow \mathsf{Com.Setup}(1^\lambda, \ell', q) \\ & \mathsf{aux}_0 := \bot \\ |\mathbb{x}| \leq n, & \rho_0 := \bot \\ (\mathbb{x}, \mathbb{w}) \notin \mathcal{R}, & \forall i \in [m+1] : \begin{pmatrix} (q_i, \mathsf{aux}_i) \leftarrow \mathbf{P}_i(\mathsf{crs}, \mathsf{aux}_{i-1}) \\ \rho_i = \mathcal{O}(q_i) \end{pmatrix} \\ \mathsf{Com.Verify}(\mathsf{crs}, \mathsf{c}, M', \alpha, \mathsf{pf}) = 1, & (\alpha, \mathsf{pf}) \leftarrow \mathbf{P}'^{\mathcal{O}}(\mathsf{crs}, \mathsf{aux}_{m+1}) \\ \alpha \in S & (\mathsf{c}, \mathbb{x}) = q_{i^*} \\ & (M, S) \leftarrow \mathbf{V}(\mathbb{x}, \rho_{i^*}) \\ & \mathbb{w} \leftarrow \mathcal{E}(\mathbf{P}, \mathsf{crs}) \end{array}\right]$$

We add to the experiment an extractor for every query that the prover makes. So far we have only

[14]Formally, since Ext does not expect to receive an oracle circuit, by $\mathbf{P}_j^{\mathcal{O}}$ we mean the circuit that has the random oracle answers hard-coded. Observe that $\mathcal{E}$ can construct this circuit as it can collect all of the random oracle queries when it runs $\mathbf{P}^{\mathcal{O}}$. Furthermore, $\mathbf{P}_j^{\mathcal{O}}$ outputs a query q_j, which, in the case that we care about is equal to $(\mathbb{x}_j, \mathsf{c}_j)$ and not just a commitment as Ext expects. For simplicity of notation we ignore this discrepancy as the correct mapping is clear from context.

added it to the experiment, so the probability is equal:

$$= \Pr\left[\begin{array}{l|l} \begin{array}{l} |\mathbb{x}| \le n, \\ (\mathbb{x}, \mathbb{w}) \notin \mathcal{R}, \\ \mathsf{Com.Verify}(\mathsf{crs}, \mathsf{c}, M', \alpha, \mathsf{pf}) = 1, \\ \alpha \in S \end{array} & \begin{array}{l} \mathcal{O} \leftarrow \mathsf{RO}(1^\lambda) \\ \mathsf{crs} \leftarrow \mathsf{Com.Setup}(1^\lambda, \ell', q) \\ \mathsf{aux}_0 := \bot \\ \rho_0 := \bot \\ \forall i \in [m+1] : \begin{pmatrix} (q_i, \mathsf{aux}_i) \leftarrow \mathbf{P}_i(\mathsf{crs}, \mathsf{aux}_{i-1}) \\ \pi'_i \leftarrow \mathsf{Ext}(\mathbf{P}_1^{\mathcal{O}}, \ldots, \mathbf{P}_i^{\mathcal{O}}, \mathsf{crs}) \\ \rho_i = \mathcal{O}(q_i) \end{pmatrix} \\ (\alpha, \mathsf{pf}) \leftarrow \mathbf{P}'^{\mathcal{O}}(\mathsf{crs}, \mathsf{aux}_{m+1}) \\ (\mathsf{c}, \mathbb{x}) = q_{i^*} \\ (M, S) \leftarrow \mathbf{V}(\mathbb{x}, \rho_{i^*}) \\ \mathbb{w} \leftarrow \mathcal{E}(\mathbf{P}, \mathsf{crs}) \end{array} \end{array}\right]$$

Opening up the extractor $\mathcal{E}$ and noting that it computes π_{i^*} exactly as the way already computed in the experiment during the main loop, the above is equal to:

$$= \Pr\left[\begin{array}{l|l} \begin{array}{l} |\mathbb{x}| \le n, \\ (\mathbb{x}, \mathbb{w}) \notin \mathcal{R}, \\ \mathsf{Com.Verify}(\mathsf{crs}, \mathsf{c}, M', \alpha, \mathsf{pf}) = 1, \\ \alpha \in S \end{array} & \begin{array}{l} \mathcal{O} \leftarrow \mathsf{RO}(1^\lambda) \\ \mathsf{crs} \leftarrow \mathsf{Com.Setup}(1^\lambda, \ell', q) \\ \mathsf{aux}_0 := \bot \\ \rho_0 := \bot \\ \forall i \in [m+1] : \begin{pmatrix} (q_i, \mathsf{aux}_i) \leftarrow \mathbf{P}_i(\mathsf{crs}, \mathsf{aux}_{i-1}) \\ \pi'_i \leftarrow \mathsf{Ext}(\mathbf{P}_1^{\mathcal{O}}, \ldots, \mathbf{P}_i^{\mathcal{O}}, \mathsf{crs}) \\ \rho_i = \mathcal{O}(q_i) \end{pmatrix} \\ (\alpha, \mathsf{pf}) \leftarrow \mathbf{P}'^{\mathcal{O}}(\mathsf{crs}, \mathsf{aux}_{m+1}) \\ (\mathsf{c}, \mathbb{x}) = q_{i^*} \\ (M, S) \leftarrow \mathbf{V}(\mathbb{x}, \rho_{i^*}) \\ \mathbb{w} \leftarrow \mathbf{E}(\mathbb{x}, \pi_{i^*}) \end{array} \end{array}\right]$$

By multi-extraction of the linear map commitment scheme, α in the above check that $\alpha \in S$ can be replaced with $M\pi_{i^*}$ up to an added error of ε so that the above is bounded by:

$$\le \Pr\left[\begin{array}{l|l} \begin{array}{l} |\mathbb{x}| \le n, \\ (\mathbb{x}, \mathbb{w}) \notin \mathcal{R}, \\ \mathsf{Com.Verify}(\mathsf{crs}, \mathsf{c}, M', \alpha, \mathsf{pf}) = 1, \\ (M\pi_{i^*}) \in S \end{array} & \begin{array}{l} \mathcal{O} \leftarrow \mathsf{RO}(1^\lambda) \\ \mathsf{crs} \leftarrow \mathsf{Com.Setup}(1^\lambda, \ell', q) \\ \mathsf{aux}_0 := \bot \\ \rho_0 := \bot \\ \forall i \in [m+1] : \begin{pmatrix} (q_i, \mathsf{aux}_i) \leftarrow \mathbf{P}_i(\mathsf{crs}, \mathsf{aux}_{i-1}) \\ \pi'_i \leftarrow \mathsf{Ext}(\mathbf{P}_1^{\mathcal{O}}, \ldots, \mathbf{P}_i^{\mathcal{O}}, \mathsf{crs}) \\ \rho_i = \mathcal{O}(q_i) \end{pmatrix} \\ (\alpha, \mathsf{pf}) \leftarrow \mathbf{P}'^{\mathcal{O}}(\mathsf{crs}, \mathsf{aux}_{m+1}) \\ (\mathsf{c}, \mathbb{x}) = q_{i^*} \\ (M, S) \leftarrow \mathbf{V}(\mathbb{x}, \rho_{i^*}) \\ \mathbb{w} \leftarrow \mathbf{E}(\mathbb{x}, \pi_{i^*}) \end{array} \end{array}\right]$$

$$+ \varepsilon(\lambda, \ell', q, m+1, s + \mathsf{vt} + \mathsf{t}_\varphi)$$

We can now remove the verification of the commitment scheme. This, in turn, allows us to remove

$\mathbf{P}'$ from the experiment as its outputs are not used anymore:

$$\leq \Pr\left[\begin{array}{l|l} |\mathbb{x}| \leq n, & \mathcal{O} \leftarrow \mathsf{RO}(1^\lambda) \\ (\mathbb{x}, \mathbb{w}) \notin \mathcal{R}, & \mathsf{crs} \leftarrow \mathsf{Com.Setup}(1^\lambda, \ell', q) \\ (M\pi_{i^*}) \in S & \mathsf{aux}_0 := \bot \\ & \rho_0 := \bot \\ & \forall i \in [m+1] : \begin{pmatrix} (q_i, \mathsf{aux}_i) \leftarrow \mathbf{P}_i(\mathsf{crs}, \mathsf{aux}_{i-1}) \\ \pi'_i \leftarrow \mathsf{Ext}(\mathbf{P}_1^{\mathcal{O}}, \ldots, \mathbf{P}_i^{\mathcal{O}}, \mathsf{crs}) \\ \rho_i = \mathcal{O}(q_i) \end{pmatrix} \\ & (\mathsf{c}, \mathbb{x}) = q_{i^*} \\ & (M, S) \leftarrow \mathbf{V}(\mathbb{x}, \rho_{i^*}) \\ & \mathbb{w} \leftarrow \mathbf{E}(\mathbb{x}, \pi_{i^*}) \end{array}\right] + \varepsilon(\lambda, \ell', q, m+1, s + \mathsf{vt} + \mathsf{t}_\varphi)$$

Observe now that we only need to run $\mathbf{P}$ up to the i^*-th query (recall that i^* is not efficiently computable, and so we are not done yet):

$$= \Pr\left[\begin{array}{l|l} |\mathbb{x}| \leq n, & \mathcal{O} \leftarrow \mathsf{RO}(1^\lambda) \\ (\mathbb{x}, \mathbb{w}) \notin \mathcal{R}, & \mathsf{crs} \leftarrow \mathsf{Com.Setup}(1^\lambda, \ell', q) \\ (M\pi_{i^*}) \in S & \mathsf{aux}_0 := \bot \\ & \rho_0 := \bot \\ & \forall i \in [i^*-1] : \begin{pmatrix} (q_i, \mathsf{aux}_i) \leftarrow \mathbf{P}_i(\mathsf{crs}, \mathsf{aux}_{i-1}) \\ \pi'_i \leftarrow \mathsf{Ext}(\mathbf{P}_1^{\mathcal{O}}, \ldots, \mathbf{P}_i^{\mathcal{O}}, \mathsf{crs}) \\ \rho_i = \mathcal{O}(q_i) \end{pmatrix} \\ & (q_{i^*}, \mathsf{aux}_{i^*}) \leftarrow \mathbf{P}_{i^*}(\mathsf{crs}, \mathsf{aux}_{i^*-1}) \\ & \pi'_{i^*} \leftarrow \mathsf{Ext}(\mathbf{P}_1^{\mathcal{O}}, \ldots, \mathbf{P}_{i^*}^{\mathcal{O}}, \mathsf{crs}) \\ & (\mathsf{c}, \mathbb{x}) = q_{i^*} \\ & \rho_{i^*} = \mathcal{O}(q_{i^*}) \\ & (M, S) \leftarrow \mathbf{V}(\mathbb{x}, \rho_{i^*}) \\ & \mathbb{w} \leftarrow \mathbf{E}(\mathbb{x}, \pi_{i^*}) \end{array}\right] + \varepsilon(\lambda, \ell', q, m+1, s + \mathsf{vt} + \mathsf{t}_\varphi)$$

We now sample $j \leftarrow [m+1]$ instead of using i^*, getting $j = i^*$ with probability $1/(m+1)$. Thus, the above is bounded by:

$$\leq (m+1) \cdot \Pr\left[\begin{array}{l|l} |\mathbb{x}| \leq n, & \mathcal{O} \leftarrow \mathsf{RO}(1^\lambda) \\ (\mathbb{x}, \mathbb{w}) \notin \mathcal{R}, & j \leftarrow [m+1] \\ (M\pi_i) \in S & \mathsf{crs} \leftarrow \mathsf{Com.Setup}(1^\lambda, \ell', q) \\ & \mathsf{aux}_0 := \bot \\ & \rho_0 := \bot \\ & \forall i \in [j-1] : \begin{pmatrix} (q_i, \mathsf{aux}_i) \leftarrow \mathbf{P}_i(\mathsf{crs}, \mathsf{aux}_{i-1}) \\ \pi'_i \leftarrow \mathsf{Ext}(\mathbf{P}_1^{\mathcal{O}}, \ldots, \mathbf{P}_i^{\mathcal{O}}, \mathsf{crs}) \\ \rho_i = \mathcal{O}(q_i) \end{pmatrix} \\ & (q_j, \mathsf{aux}_j) \leftarrow \mathbf{P}_j(\mathsf{crs}, \mathsf{aux}_{j-1}) \\ & \pi'_j \leftarrow \mathsf{Ext}(\mathbf{P}_1^{\mathcal{O}}, \ldots, \mathbf{P}_j^{\mathcal{O}}, \mathsf{crs}) \\ & (\mathsf{c}, \mathbb{x}) = q_j \\ & \rho_j = \mathcal{O}(q_j) \\ & (M, S) \leftarrow \mathbf{V}(\mathbb{x}, \rho_j) \\ & \mathbb{w} \leftarrow \mathbf{E}(\mathbb{x}, \pi_j) \end{array}\right] + \varepsilon(\lambda, \ell', q, m+1, s+\mathsf{vt}+\mathsf{t}_\varphi)$$

We now notice that, as $\mathcal{O}(q_j)$ is queried only once (since $\mathbf{P}$ does not make the same query twice), we can sample it uniformly at random rather than from the random oracle:

$$\leq (m+1)\cdot \Pr\left[\begin{array}{l|l} & \mathcal{O} \leftarrow \mathsf{RO}(1^\lambda) \\ & j \leftarrow [m+1] \\ & \mathsf{crs} \leftarrow \mathsf{Com.Setup}(1^\lambda, \ell', q) \\ & \mathsf{aux}_0 := \bot \\ & \rho_0 := \bot \\ |\mathbb{x}| \leq n, & \forall i \in [j-1] : \begin{pmatrix} (q_i, \mathsf{aux}_i) \leftarrow \mathbf{P}_i(\mathsf{crs}, \mathsf{aux}_{i-1}) \\ \pi'_i \leftarrow \mathsf{Ext}(\mathbf{P}_1^{\mathcal{O}}, \dots, \mathbf{P}_i^{\mathcal{O}}, \mathsf{crs}) \\ \rho_i = \mathcal{O}(q_i) \end{pmatrix} \\ (\mathbb{x}, \mathbb{w}) \notin \mathcal{R}, & (q_j, \mathsf{aux}_j) \leftarrow \mathbf{P}_j(\mathsf{crs}, \mathsf{aux}_{j-1}) \\ (M\pi_i) \in S & \pi'_j \leftarrow \mathsf{Ext}(\mathbf{P}_1^{\mathcal{O}}, \dots, \mathbf{P}_j^{\mathcal{O}}, \mathsf{crs}) \\ & (\mathsf{c}, \mathbb{x}) = q_j \\ & \rho_j \leftarrow \{0,1\}^\lambda \\ & (M, S) \leftarrow \mathbf{V}(\mathbb{x}, \rho_j) \\ & \mathbb{w} \leftarrow \mathbf{E}(\mathbb{x}, \pi_j) \end{array}\right] + \varepsilon(\lambda, \ell', q, m+1, s+\mathsf{vt}+\mathsf{t}_\varphi)$$

Finally, by constructing a prover that runs the algorithms in in the experiment above up to the sampling of ρ_j (while emulating $\mathcal{O}$ with uniformly random answers) and outputting $\mathbb{x}$ and π'_i, and by knowledge soundness of the PCP, the above is bounded by $(m+1)\cdot\kappa+\varepsilon(\lambda, \ell', q, m+1, s+\mathsf{vt}+\mathsf{t}_\varphi)$ which is, in turn, bounded by $(s+1)\cdot\kappa + \varepsilon(\lambda, \ell', q, s+1, s+\mathsf{vt}+\mathsf{t}_\varphi)$. □

□

5.3 Proof of Theorems 5.1 and 5.2

In this section, we prove Theorem 5.1 and Theorem 5.2 by putting together a linear PCP and the linear map commitment described in Theorem 4.3 and deriving appropriate bounds from Lemma 5.11. Note that for both theorems, when analyzing the completeness error we will use the tighter multi-extraction bound of $\varepsilon(\lambda, \ell, q, n, s) = \frac{20\ell^3 q^2 s + 3\ell s^2}{2^\lambda}$ for the linear map commitment (see proof of Theorem 4.3).

- Theorem 5.1: we use the PCP for $\mathsf{GapMWSP}_m$ of Theorem 5.7. Letting $t = \lambda/\operatorname{loglog} n$ in Lemma 5.11, we have the following:
  - Completeness error:

$$\begin{aligned} &\alpha^t + t^2 \cdot \varepsilon(\lambda, n+1, 1, 1, n^2 + 2(n+1)) \\ &= \left(\frac{1}{\log n}\right)^{\frac{\lambda}{\operatorname{loglog} n}} + \left(\frac{\lambda}{\operatorname{loglog} n}\right)^2 \cdot \left(\frac{20(n+1)^3((n+1)^2 + 2(n+1)) + 3(n+1)((n+1)^2 + 2(n+1))^2}{2^\lambda}\right) \\ &< \frac{24 n^5 \lambda^2}{2^\lambda \operatorname{loglog}(n)^2}, \end{aligned}$$

    where the final inequality holds since $n > 170$.

– Soundness/knowledge error:

$$\begin{aligned}
&(s+1)\cdot\beta+\varepsilon(\lambda,n+1,1,s+1,s+3(n+1))\\
&=\frac{s+1}{e^{\log^{c+1}n}}+\frac{s+1}{2^\lambda}+\frac{4(n+1)\cdot(s+3(n+1))^2}{2^\lambda}\\
&=\frac{s+1}{e^{\log^{c+1}n}}+\frac{4ns^2+36n^3+24n^2s+108n^2+48ns+108n+4s^2+25s+37}{2^\lambda}\\
&\le\frac{s+1}{n^{\log^c n}}+\frac{5ns^2}{2^\lambda}\,,
\end{aligned}$$

where the final inequality follows from $s>20n^2$ and $n>170$.

- Theorem 5.2: we ue the PCP of Corollary 5.10. Letting $t=2\lambda^2$ in Lemma 5.11, we have the following error bounds:

– Completeness error:

$$\begin{aligned}
&\alpha^t+t^2\cdot\varepsilon(\lambda,n+1,1,1,n^2+2(n+1))\\
&=\left(1-\frac{1}{2\lambda}\right)^{2\lambda^2}+4\lambda^4\cdot\left(\frac{20(n+1)^3((n+1)^2+2(n+1))+3(n+1)((n+1)^2+2(n+1))^2}{2^\lambda}\right)\\
&\le\frac{1}{2^\lambda}+4\lambda^4\cdot\left(\frac{20(n+1)^3((n+1)^2+2(n+1))+3(n+1)((n+1)^2+2(n+1))^2}{2^\lambda}\right)\\
&<\frac{93n^5\lambda^2}{2^\lambda}\,,
\end{aligned}$$

where the final inequality holds since $n>671$.

– Soundness error:

$$\begin{aligned}
&(s+1)\cdot\beta+\varepsilon(\lambda,n+1,1,s+1,s+3(n+1))\\
&=\frac{(s+1)(\lambda+2)}{2^\lambda}+\frac{4(n+1)\cdot(s+3(n+1))^2}{2^\lambda}\\
&=\frac{4ns^2+36n^3+24n^2s+108n^2+48ns+108n+4s^2+(26+\lambda)s+\lambda+38}{2^\lambda}\\
&\le\frac{5ns^2}{2^\lambda}
\end{aligned}$$

Where the final inequality follows from $s>20n^2$, $ns>8(25+\lambda)$ and $n>671$.

6 Lower bound for SNARGs with one group element

In previous sections, we showed the existence of two group element SNARGs in the generic group model with random oracles. In this section, we established a lower bound matching our constructions. With some caveats, we prove that in Maurer's generic group model [Mau05] for asymmetric pairing groups with a random oracle, there does not exist a SNARG.

These caveats are that the setup algorithm does not have access to the random oracle and that the verifier queries are non-adaptive[15] All group-based publicly verifiable SNARGs we are aware of fall into the model covered by our lower bound.

Theorem 6.1. *If one-way functions exist (in the plain model) then there does not exist a SNARG for* NP *in the Maurer bilinear GGM ($\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$) + ROM with the following properties:*

- *The proof consists of a single group element (of either $\mathbb{G}_1$,$\mathbb{G}_2$, or $\mathbb{G}_T$);*
- *The completeness and soundness errors are negligible in λ;*
- *The verifier makes only non-adaptive ROM queries of the form $(\mathbb{G}_1 \cup \mathbb{G}_2 \cup \mathbb{G}_T \cup \{0,1\})^* \to \{0,1\} \times \mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{G}_T \times \{0,1\}$ where ROM queries are done prior to any GGM equality checks (or, equivalently, do not depend on them);*
- *The setup algorithm does not make random oracle queries.*

Proof. Follows from the upcoming Lemmas 6.4, 6.7, 6.9 and 6.11. □

Remark 6.2 (Comparisson to [Gro16])**.** Groth [Gro16], proved a limitation on the structure of SNARGs: in the asymmetric pairing setting there are no SNARGs where

1. The proof only consists of (possibly multiple) $\mathbb{G}_1$ group elements; and
2. The output of the verifier is a conjunction of predefined pairing equation checks.

This result is incomparable to Theorem 6.1 as it also rules out SNARGs with multiple $\mathbb{G}_1$ elements, while we rule out SNARGs with much more expressive verifiers. Indeed, a variety of modern SNARGs (e.g., [GWC19; Lip24; DMS24]) have verifiers that follow a structure not covered by the lower bound in [Gro16] because they utilise a random oracle. As shown in these works, by using a random oracle (and thus bypassing the structure required by Groth) it is possible where the proof consists only of $\mathbb{G}_1$ elements.

Remark 6.3. The proof in this section focuses on the prover's message Π being a $\mathbb{G}_1$ element. By symmetry the proof is identical if it is a $\mathbb{G}_2$ element. If Π is a $\mathbb{G}_T$ element then small changes are required due to the difference in ways that the prover can generate a $\mathbb{G}_T$ element compared to a $\mathbb{G}_1$ element, but the proof remains essentially the same.

Matching of upper bound. The SNARG construction in [SW14], together with our results characterize the tightness of our lower bound. We rule out SNARGs with ≤ 1 group elements plus $O(\log \lambda)$ bits[16]. Feasibility of a construction requires either relaxing the overhead to > 1 group elements (our constructions) or allowing $\omega(\log \lambda)$ extra bits (as in [SW14]).

[15]With non-adaptive, we mean that the verifier circuit $\mathcal{V}(\mathsf{crs}, \mathbb{x}, \Pi)$ can be split into two parts $\mathcal{V}_1$ and $\mathcal{V}_2$. $\mathcal{V}_1(\mathsf{crs}, \mathbb{x}, \Pi)$ does not do any equality check GGM calls or ROM calls outputs a state State and queries $(q_i)_{i\in[\ell]}$. $\mathcal{V}_2$ receives State and ROM response to these queries $(\mathcal{O}(q_i))_{i\in[\ell]}$ and accepts or rejects. $\mathcal{V}_2$ is allowed equality check GGM calls but no further ROM calls.

[16]We can handle these extra bits by having the verifier check all $2^{O(\log \lambda)}$ possibilities, which only reduces soundness polynomially.

Roadmap. To prove Theorem 6.1 we first prove in Section 6.1 that the random oracle does not help to build one group element SNARGs. We do this by compiling every one group element SNARG in Maurer's model with non-adaptive random oracle calls to a one group element SNARG in Maurer's model *without the random oracle*. Then, in Section 6.2, we prove that a one group element SNARG in Maurer's generic group model does not exist if one-way functions exist (in the plain model).

Notation 1. In this section we use additive notation for the cryptographic group. For example $[\mathbf{u}]_1$ are a vector of group elements in $\mathbb{G}_1$ and $\mathbf{u}$ is the vector of the discrete logarithms of $[\mathbf{u}]_1$. Similarly, $[\mathbf{v}]_2$ and $[\mathbf{w}]_T$ are vectors of group elements in $\mathbb{G}_2$ and $\mathbb{G}_T$, respectively. If the specific group of an element with discrete logarithm u is not relevant we will denote it by $[u]_*$.

6.1 Removing the ROM from one group element SNARGs

In this section, we show how to remove the random oracle from any one group element SNARG with non-adaptive random oracle calls. We achieve this in two steps. In the first step, we remove all the verifier calls to the random oracle which do not depend on the SNARG proof Π. Then, in a second step, we remove all the other calls to the random oracle. We discuss what it means for a group element to depend on the proof later in this section.

6.1.1 Making the verifier deterministic

To remove random oracle queries, we will need it to be the case that the verifier is deterministic (except for the responses of the random oracle). Here we argue that this holds without loss of generality. We amplify the completeness and soundness over the randomness of the verifier and then we fix a randomness that preserves completeness and soundness using an averaging argument.

Lemma 6.4. *Let $(\mathcal{S}', \mathcal{P}', \mathcal{V}')$ be a SNARG for relation $\mathcal{R}$ with completeness error $\alpha(\lambda)$ where $\alpha \leq 1 - 1/\mathrm{poly}(\lambda)$ and (non-adaptive) soundness error $\beta(\lambda)$. Then there exists a SNARG for $\mathcal{R}$ with a deterministic verifier with completeness error $\alpha + 2^{-\lambda}$ and (non-adaptive) soundness error $4\beta/(1-\alpha) + 2^{-\lambda}$. The argument size is preserved and all other parameters are polynomially related.*

Remark 6.5. While described for SNARGs in the standard model, the proof of Lemma 6.4 relativizes with respect to oracles, and so it also holds for SNARG constructions relative to oracles.

Proof. Let $(\mathcal{S}', \mathcal{P}', \mathcal{V}')$ be the SNARG with randomized verifier and denote by $|\mathsf{crs}'|$ its common reference string size and $|\Pi|$ its argument string size. Let $t = \frac{12\log(e)(\lambda+|\mathsf{crs}'|+|\Pi|)}{1-\alpha}$ be a repetition parameter, and observe that t is polynomial in λ. We construct the following SNARG $(\mathcal{S}, \mathcal{P}, \mathcal{V})$ with deterministic verifier:

- $\mathcal{S}(1^\lambda)$:
  1. Let $\mathsf{crs}' \leftarrow \mathcal{S}'(1^\lambda)$.
  2. For $i \in t$ sample verifier randomness r_i uniformly at random.
  3. Output $\mathsf{crs} := (\mathsf{crs}', (\mathsf{r}_i)_{i\in[t]})$
- $\mathcal{P}(\mathsf{crs}, \mathbb{x}, \mathbb{w})$: Output $\Pi \leftarrow \mathcal{P}'(\mathsf{crs}, \mathbb{x}, \mathbb{w})$.
- $\mathcal{V}(\mathsf{crs}, \mathbb{x}, \Pi)$:

1. Let $\mathsf{counter} \leftarrow 0$.
2. For $i \in [t]$: Let $\mathsf{counter} \leftarrow \mathsf{counter} + \mathcal{V}'(\mathsf{crs}, \mathbb{x}, \Pi; \mathsf{r}_i)$.
3. If $\mathsf{counter} \geq \frac{(1-\alpha)\cdot t}{2}$ output 1 otherwise output 0.

Completeness. Fix an $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$. For a common reference string crs', let

$$p_{\mathsf{crs}'} = \Pr_{\mathsf{r}_{\mathcal{V}'}}[\mathcal{V}'_{\mathsf{crs}'}(\mathbb{x}, \mathcal{P}'_{\mathsf{crs}'}(\mathbb{x}, \mathbb{w}); \mathsf{r}_{\mathcal{V}'}) = 1].$$

Fix a common reference string crs' with $p_{\mathsf{crs}'} \geq 1-\alpha$. Letting X_i be the 0/1 random variable where

$$X_i = 1 \iff \mathcal{V}'_{\mathsf{crs}'}(\mathbb{x}, \mathcal{P}'_{\mathsf{crs}'}(\mathbb{x}, \mathbb{w}); \mathsf{r}_i) = 1,$$

so that $X_{\mathsf{crs}'} = \sum_{i=1}^{t} X_i$ has $\mathbb{E}[\mathsf{counter}] = (1-\alpha)\cdot t$, and applying the multiplicative Chernoff bound, we have that (when crs' is chosen),

$$\Pr\left[X_{\mathsf{crs}'} \leq \frac{(1-\alpha)\cdot t}{2}\right] \leq 2^{-(1-\alpha)\cdot t \cdot \log(e)/8} \leq 2^{-\lambda}.$$

Observe that, by completeness of the original SNARG, $\Pr_{\mathsf{crs}'}[p_{\mathsf{crs}'} \geq (1-\alpha)] \geq 1-\alpha$. Since $\mathcal{V}$ accepts if and only if $\mathsf{counter} \geq (1-\alpha)t/2$, which is equivalent to requiring that crs' is chosen such that $X_{\mathsf{crs}'} \geq (1-\alpha)t/2$, we have that $\mathcal{V}$ rejects with probability at most

$$\Pr[p_{\mathsf{crs}'} < 1-\alpha] + \Pr\left[X_{\mathsf{crs}'} \leq \frac{(1-\alpha)\cdot t}{2} \;\middle|\; p_{\mathsf{crs}'} \geq 1-\alpha\right] \leq \alpha + 2^{-\lambda},$$

where above the probabilities are over the randomness of crs (which is the only randomness in the new SNARG).

Soundness. Fix an instance $\mathbb{x} \notin L$ and a malicious prover $\tilde{\mathcal{P}}$. For a common reference string crs' and proof string Π let

$$p_{\mathsf{crs}',\Pi} = \Pr_{\mathsf{r}_{\mathcal{V}}}[\mathcal{V}'(\mathsf{crs}, \mathbb{x}, \Pi; \mathsf{r}_{\mathcal{V}}) = 1].$$

Fix crs' and Π with $p_{\mathsf{crs}',\Pi} < (1-\alpha)/4$ and let X_i be the 0/1 random variable where

$$X_i = 1 \iff \mathcal{V}'_{\mathsf{crs}'}(\mathbb{x}, \mathcal{P}'_{\mathsf{crs}'}(\mathbb{x}, \mathbb{w}); \mathsf{r}_i) = 1,$$

Then $X_{\mathsf{crs}',\Pi} = \sum_{i=1}^{t} X_i$ has expectation $\mathbb{E}[X_{\mathsf{crs}',\Pi}] < (1-\alpha)\cdot t/4$. By the multiplicative Chernoff bound,

$$\Pr_{\mathsf{r}_1,\ldots,\mathsf{r}_t}\left[X_{\mathsf{crs}',\Pi} \geq \frac{(1-\alpha)\cdot t}{2}\right] \leq 2^{-4(1-\alpha)\cdot\log(e)\cdot t/3} \leq 2^{-(\lambda+|\mathsf{crs}'|+|\Pi|)}.$$

By taking a union bound over all crs' and Π, we have that

$$\Pr_{\mathsf{r}_1,\ldots,\mathsf{r}_t}\left[\exists \mathsf{crs}', \Pi : p_{\mathsf{crs}',\Pi} \leq (1-\alpha)\cdot t/4 \wedge X_{\mathsf{crs}',\Pi} \geq \frac{(1-\alpha)\cdot t}{2}\right] \leq 2^{-\lambda}$$

Finally, observe that by soundness of the SNARG:

$$\begin{aligned}\beta \geq \Pr_{\mathsf{r}_{\mathcal{V}}}[\mathcal{V}'(\mathsf{crs}', \mathbb{x}, \tilde{\mathcal{P}}(\mathsf{crs}'); \mathsf{r}_{\mathcal{V}}) = 1] &= \sum_{\delta} \delta \cdot \Pr_{\mathsf{crs}'}[p_{\mathsf{crs}',\tilde{\mathcal{P}}(\mathsf{crs}')} = \delta] \\ &\geq (1-\alpha)/4 \cdot \Pr_{\mathsf{crs}'}[p_{\mathsf{crs}',\tilde{\mathcal{P}}(\mathsf{crs}')} \geq (1-\alpha)/4],\end{aligned}$$

and so $\Pr_{\mathsf{crs}'}[p_{\mathsf{crs}',\tilde{\mathcal{P}}(\mathsf{crs}')} \geq (1-\alpha)/4] \leq 4\beta/(1-\alpha)$.

Since $\mathcal{V}$ accepts if and only if $\mathsf{counter} \geq (1-\alpha)t/2$, which is equivalent to requiring that crs' and $\Pi = \tilde{\mathcal{P}}(\mathsf{crs}')$ is chosen such that $X_{\mathsf{crs}',\tilde{\mathcal{P}}(\mathsf{crs}')} \geq (1-\alpha)t/2$, we have that $\mathcal{V}$ accepts with probability at most

$$\Pr_{\mathsf{crs}'}[p_{\mathsf{crs}',\tilde{\mathcal{P}}(\mathsf{crs}')} \geq (1-\alpha)/4] + \Pr_{\mathsf{crs}',\mathsf{r}_1,\ldots,\mathsf{r}_t}\left[X_{\mathsf{crs}',\tilde{\mathcal{P}}(\mathsf{crs}')} \geq \frac{(1-\alpha)\cdot t}{2} \mid p_{\mathsf{crs}',\tilde{\mathcal{P}}(\mathsf{crs}')} \leq (1-\alpha)\cdot t/4\right],$$

which is bounded by $4\beta/(1-\alpha) + 2^{-\lambda}$. $\square$

6.1.2 Defining proof-dependent and independent queries

In the generic group model for asymmetric pairing groups, any group element that an algorithm produces can be thought of as a polynomial in the algorithm's inputs. In our setting, for example, any group element the verifier receives can be thought of as a polynomial f in the formal variables $\hat{\Pi}$, $\hat{\mathsf{crs}}_1$, $\hat{\mathsf{crs}}_2$, and $\hat{\mathsf{crs}}_T$. These formal variables represent the proof Π, and the vector of $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ elements in the common reference string crs.

We consider the case where the proof is an element in $\mathbb{G}_1$. By symmetry between $\mathbb{G}_1$ and $\mathbb{G}_2$ this suffices also for the case that the proof is in $\mathbb{G}_2$. The proof for $\mathbb{G}_T$ is similar while accounting to the different structure of the possible ways the prover can generate a $\mathbb{G}_T$ proof. As the proof is otherwise identical, we focus on $\mathbb{G}_1$.

To indicate that the proof is in $\mathbb{G}_1$, we write $[\Pi]_1$ from here on.

Then in the verifier all elements in $\mathbb{G}_1$ can be expressed by a polynomial $a\hat{\Pi} + \langle \mathbf{b}, \hat{\mathsf{crs}}_1\rangle$ where $a \in \mathbb{Z}_p$, and $\mathbf{b}$ is a vector over $\mathbb{Z}_p$ because $[\Pi]_1$ and the elements in $[\mathsf{crs}_1]_1$ are in $\mathbb{G}_1$. Similarly, all elements in $\mathbb{G}_2$ can be expressed by a polynomial $\langle \mathbf{b}, \hat{\mathsf{crs}}_2\rangle$ because $[\Pi]_1$ can not be moved to $\mathbb{G}_2$ and all $\mathbb{G}_T$ elements can be represented by $\left\langle \mathbf{a}, \hat{\Pi}\cdot\hat{\mathsf{crs}}_2\right\rangle + \langle \mathbf{b}, \hat{\mathsf{crs}}_1 \otimes \hat{\mathsf{crs}}_2\rangle + \langle \mathbf{c}, \hat{\mathsf{crs}}_T\rangle$.

These properties allow us to define a function $\mathsf{IsIndependent}$ which determines whether the output of such a polynomial is independent of the proof $[\Pi]_1$. Essentially, it is independent if all coefficients in monomials containing $\hat{\Pi}$ are 0. More precisely,

$\mathsf{IsIndependent}_{\mathsf{crs}}(f(\hat{\Pi}, \hat{\mathsf{crs}}_1, \hat{\mathsf{crs}}_2, \hat{\mathsf{crs}}_T))$: where $\hat{\Pi}$, $\hat{\mathsf{crs}}_1$, $\hat{\mathsf{crs}}_2$, and $\hat{\mathsf{crs}}_T$ are formal variables respresenting the proof and the $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ elements in the common reference string, respectively. We abuse notation and say that f outputs a $\mathbb{G}_1, \mathbb{G}_2$ or $\mathbb{G}_T$ element, meaning that the output of f represents the discrete logarithm of an element of that group. Whenever $\mathsf{IsIndependent}$ is called, it will be clear which group this is referring to, and so we do not specify this as an input, but leave it implicit.

1. If f outputs a $\mathbb{G}_1$ element:
   (a) f has the form
   $$a\hat{\Pi} + \langle \mathbf{b}, \hat{\mathsf{crs}}_1\rangle$$
   where $a \in \mathbb{Z}_p$, and $\mathbf{b}$ is a vector over $\mathbb{Z}_p$.
   (b) If $a = 0$ then output 1.
   (c) If $a \neq 0$ then output 0.
2. If f outputs a $\mathbb{G}_2$ element then output 1.
3. If f outputs a $\mathbb{G}_T$ element:

(a) f has the form
$$\hat{\Pi} \cdot \langle \mathbf{a}, \hat{\mathsf{crs}}_2 \rangle + \langle \mathbf{b}, \hat{\mathsf{crs}}_1 \otimes \hat{\mathsf{crs}}_2 \rangle + \langle \mathbf{c}, \hat{\mathsf{crs}}_T \rangle$$
where $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ are vectors over $\mathbb{Z}_p$.

(b) If $\langle \mathbf{a}, [\mathsf{crs}_2]_2 \rangle = [0]_2$ then output 1.

(c) If $\langle \mathbf{a}, [\mathsf{crs}_2]_2 \rangle \neq [0]_2$ then output 0.

Crucial observation: Our proof relies on the fact that any algorithm that has access to crs and the structure of f can compute $\mathsf{IsIndependent}_{\mathsf{crs}}$ on their own. For $\mathbb{G}_2$ elements this is trivial. For G_1 elements this requires only knowing a, which is fixed in f. For $\mathbb{G}_T$ elements, since $\mathbf{a}$ is known, it is possible to compute the group element whose exponent is $\langle \mathbf{a}, [\mathsf{crs}_2]_2 \rangle$, and then check whether this exponent is 0 or not via a GGM equality check.

Claim 6.6. *Let f a function such that $\mathsf{IsIndependent}(f) = 0$. Then for every crs there exists a function f^{-1}_{crs} such that $f^{-1}_{\mathsf{crs}}(f(\hat{\Pi}, \mathsf{crs})) = \hat{\Pi}$.*

Proof. The function f^{-1} will be different depending on to which group the output of the query that f matches. For a common reference string crs, let $[\mathsf{crs}_1]_1$, $[\mathsf{crs}_2]_2$, and $[\mathsf{crs}_T]_T$ be the elements in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ respectively.

f outputs a $\mathbb{G}_1$ element: then f has the form $a\hat{\Pi} + \langle \mathbf{b}, \hat{\mathsf{crs}}_1 \rangle$ where $a \in \mathbb{Z}_p$ is non-zero, and $\mathbf{b}$ is a vector over $\mathbb{Z}_p$. Then
$$f^{-1}_{\mathsf{crs}}(\hat{Y}) = \frac{\hat{Y} - \langle \mathbf{b}, \mathsf{crs}_1 \rangle}{a}.$$

f outputs a $\mathbb{G}_2$ element: this cannot hold as $\mathsf{IsIndependent}(f) = 1$ whenever f outputs a $\mathbb{G}_2$ element.

f outputs a $\mathbb{G}_T$ element: then f has the form
$$\left\langle \mathbf{a}, \hat{\Pi} \cdot \hat{\mathsf{crs}}_2 \right\rangle + \langle \mathbf{b}, \hat{\mathsf{crs}}_1 \otimes \hat{\mathsf{crs}}_2 \rangle + \langle \mathbf{c}, \hat{\mathsf{crs}}_T \rangle$$
where $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ are vectors over $\mathbb{Z}_p$, and $\langle \mathbf{a}, [\mathsf{crs}_2]_2 \rangle \neq 0$. Then
$$f^{-1}_{\mathsf{crs}}(\hat{Y}) = \frac{\hat{Y} - \langle \mathbf{b}, \mathsf{crs}_1 \otimes \mathsf{crs}_2 \rangle - \langle \mathbf{c}, \mathsf{crs}_T \rangle}{\langle \mathbf{a}, \mathsf{crs}_2 \rangle}.$$

□

6.1.3 Removing proof-independent queries

Lemma 6.7. *Let $(\mathcal{S}'^{\mathcal{G}}, \mathcal{P}'^{\mathcal{G},\mathcal{O}}, \mathcal{V}'^{\mathcal{G},\mathcal{O}})$ be a single element SNARG for relation $\mathcal{R}$ in the Maurer Bilinear GGM ($\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$) with a random oracle with a deterministic verifier that makes non-adaptive ROM $\mathbb{G}^*_{\{1,2,T\}} \to \{0,1\} \times \mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{G}_T$. Then there exists a single element SNARG for relation $\mathcal{R}$ in the Maurer Bilinear GGM with a random oracle with a deterministic verifier where for every crs in the image of the setup algorithm the polynomial f describing any verifier ROM query has $\mathsf{IsIndependent}_{\mathsf{crs}}(f) = 1$. Completeness and soundness are preserved, and all other parameters are polynomially related.*

Remark 6.8. Notice, the random oracle may take multiple group elements as input and outputs a bit and one group element from each group. This setting covers all possible ways of calling a random oracle. If an algorithm wants to hash bits, it can first encode them into group elements and then query the ROM with those group elements. With the proper domain separation, this does not cause any complications.

Proof. Given a SNARG $(\mathcal{S}'^{\mathcal{G}}, \mathcal{P}'^{\mathcal{G},\mathcal{O}}, \mathcal{V}'^{\mathcal{G},\mathcal{O}})$ in the Maurer Bilinear GGM with a random oracle. Let $T_{\mathcal{V}}$ be a bound on the running time of $\mathcal{V}$ (which also bounds its query complexity) and n be an upper bound on the number of group elements that are used as input to the random oracle in a single query (observe that both parameters are polynomial in the instance size and security parameter). We design a SNARG $(\mathcal{S}, \mathcal{P}, \mathcal{V})$ where all verifier ROM queries are proof-dependent:

- $\mathcal{S}^{\mathcal{G}}(1^\lambda)$:
  1. Sample $\mathsf{crs}' \leftarrow \mathcal{S}'^{\mathcal{G}}(1^\lambda)$.
  2. For $i \in [T_{\mathcal{V}}]$ sample $\mathsf{r}_i \leftarrow \{0,1\}$ uniformly at random.
  3. For $i \in [T_{\mathcal{V}}]$, $j \in \{1,2,T\}$ sample $\mathsf{s}_{i,j} \leftarrow \mathbb{Z}_p$ uniformly at random.
  4. Output $\mathsf{crs} := (\mathsf{crs}', (\mathsf{r}_i)_{i\in[T_{\mathcal{V}}]}, ([\mathsf{s}_{i,j}]_j)_{i\in[T_{\mathcal{V}}], j\in\{1,2,T\}})$

- $\mathcal{P}^{\mathcal{G},\mathcal{O}}(\mathsf{crs}, \mathbb{x}, \mathbb{w})$:
  1. Let $[\mathsf{crs}_1]_1$ (resp. $[\mathsf{crs}_2]_2$ and $[\mathsf{crs}_T]_T$) be the group elements in crs' which are $\mathbb{G}_1$ elements (resp. $\mathbb{G}_2$ and $\mathbb{G}_T$).
  2. Simulate $\mathcal{V}'^{\mathcal{G},\mathcal{O}}(\mathsf{crs}, \mathbb{x}, [\Pi]_1 := [1]_1)$. For each query $i \in [T_{\mathcal{V}}]$ and group element in that query $j \in [n]$ there is a polynomial $f_{i,j}(\hat{\Pi}, \hat{\mathsf{crs}}')$ outputting a group element from either $\mathbb{G}_1$, $\mathbb{G}_2$, or $\mathbb{G}_T$.
  3. Initialize a lazily sampled random oracle $\mathcal{SO}$.
  4. For each $i \in [T_{\mathcal{V}}]$ if for all $j \in [n]$ it holds that $\mathsf{IsIndependent}_{\mathsf{crs}}(f_{i,j}(\hat{\Pi}, \hat{\mathsf{crs}}')) = 1$ then program $\mathcal{SO}$ to output $(\mathsf{r}_i, [\mathsf{s}_{i,1}]_1, [\mathsf{s}_{i,2}]_2, [\mathsf{s}_{i,T}]_T)$ on input the group element $f_{i,j}([\Pi]_1 := [1]_1, \mathsf{crs}')$.
  5. Output $[\Pi]_1 := \mathcal{P}'^{\mathcal{G},\mathcal{SO}}(\mathsf{crs}, \mathbb{x}, \mathbb{w})$ where for a query X not programmed into $\mathcal{SO}$ we respond with $\mathcal{O}(X)$.

- $\mathcal{V}^{\mathcal{G},\mathcal{O}}(\mathsf{crs}, \mathbb{x}, [\Pi]_1)$:
  1. Let $[\mathsf{crs}_1]_1$ (resp. $[\mathsf{crs}_2]_2$ and $[\mathsf{crs}_T]_T$) be the group elements in crs' which are $\mathbb{G}_1$ elements (resp. $\mathbb{G}_2$ and $\mathbb{G}_T$).
  2. Simulate $\mathcal{V}'^{\mathcal{G},\mathcal{O}}(\mathsf{crs}, \mathbb{x}, [\Pi]_1 := [1]_1)$. For each query $i \in [T_{\mathcal{V}}]$ and group element in that query $j \in [n]$ there is a polynomial $f_{i,j}(\hat{\Pi}, \hat{\mathsf{crs}}')$ outputting a group element from either $\mathbb{G}_1$, $\mathbb{G}_2$, or $\mathbb{G}_T$.
  3. For each $i \in [T_{\mathcal{V}}]$ if for all $j \in [n]$ it holds that $\mathsf{IsIndependent}_{\mathsf{crs}}(f_{i,j}(\hat{\Pi}, \hat{\mathsf{crs}}')) = 1$ then program $\mathcal{SO}$ to output $(\mathsf{r}_i, [\mathsf{s}_{i,1}]_1, [\mathsf{s}_{i,2}]_2, [\mathsf{s}_{i,T}]_T)$ on input the group element $f_{i,j}([\Pi]_1 := [1]_1, \mathsf{crs}')$.
  4. Output $b := \mathcal{V}'^{\mathcal{G},\mathcal{SO}}(\mathsf{crs}, \mathbb{x}, \mathbb{w})$ where for a query $[X]_*$ not programmed into $\mathcal{SO}$ we respond with $\mathcal{O}([X]_*)$.

First, note that since $T_{\mathcal{V}}$ is polynomial, the CRS length and all other efficiency parameters remain polynomial (the proof length remains identical).

Recall that $\mathcal{V}'$ is non-adaptive with respect to the random oracle queries, meaning that the executions $\mathcal{V}'^{\mathcal{G},\mathcal{O}}(\mathsf{crs}, \mathbb{x}, [1]_1)$ and $\mathcal{V}'^{\mathcal{G},\mathcal{SO}}(\mathsf{crs}, \mathbb{x}, [\Pi]_1)$ yield the same polynomials $f_{i,j}$ that produce the queries to the random oracle. Because $\mathcal{V}'$ is deterministic and $\mathsf{IsIndependent}$ is efficiently computable

$\mathcal{P}$ and $\mathcal{V}$ will agree on which responses to retrieve from crs. Further, $(\mathsf{r}_i, [\mathsf{s}_{i,1}]_1, [\mathsf{s}_{i,2}]_2, [\mathsf{s}_{i,T}]_T)$ follow the same distribution as if drawn from a random oracle. Therefore, this transformation preserves completeness.

Soundness of $(\mathcal{S}', \mathcal{P}', \mathcal{V}')$ follows from the fact any adversary $\mathcal{A}'$ against its non-adaptive security can transformed into an attack against $(\mathcal{S}, \mathcal{P}, \mathcal{V})$ by sampling $(\mathsf{r}_i, [\mathsf{s}_{i,1}]_1, [\mathsf{s}_{i,2}]_2, [\mathsf{s}_{i,T}]_T)$ from the random oracle at which the verifier is going to make proof independent queries. □

6.1.4 Removing proof-dependent queries

In the next step, we remove all the queries that depend on the proof. We use the fact that any query that both prover and verifier have in common uniquely defines the proof because the query is a degree 1 polynomial in the proof.

Lemma 6.9. *Let $(\mathcal{S}'^{\mathcal{G}}, \mathcal{P}'^{\mathcal{G},\mathcal{O}}, \mathcal{V}'^{\mathcal{G},\mathcal{O}})$ be a single element SNARG for a relation $\mathcal{R}$ in the Maurer Bilinear GGM ($\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$) with a random oracle, completeness error α, soundness error β, and non-adaptive ROM $\mathbb{G}^*_{\{1,2,T\}} \to \{0,1\} \times \mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{G}_T$ queries by the deterministic verifier where for every crs in the image of the setup algorithm the polynomial f describing any verifier ROM query has $\mathsf{IsIndependent}_{\mathsf{crs}}(f) = 1$. Then, there exists a single element SNARG for $\mathcal{R}$ $(\mathcal{S}^{\mathcal{G}}, \mathcal{P}^{\mathcal{G}}, \mathcal{V}^{\mathcal{G}})$ in the Maurer GGM (without a random oracle), with completeness error $\alpha + 1 - 1/\mathrm{poly}(\lambda)$, soundness error β, and randomzied verifier.*

Proof. Given a SNARG $(\mathcal{S}'^{\mathcal{G}}, \mathcal{P}'^{\mathcal{G},\mathcal{O}}, \mathcal{V}'^{\mathcal{G},\mathcal{O}})$ in the Maurer GGM with ROM we prove that the following protocol in the Maurer GGM without ROM is also a SNARG.

- $\mathcal{S}^{\mathcal{G}}(1^\lambda)$:
  1. Output $\mathsf{crs} \leftarrow \mathcal{S}'^{\mathcal{G}}(1^\lambda)$.
- $\mathcal{P}^{\mathcal{G}}(\mathsf{crs}, \mathbb{x}, \mathbb{w})$:
  1. Lazily sample a random oracle $\mathcal{SO}_{\mathcal{P}}$.
  2. Output $[\Pi]_1 \leftarrow \mathcal{P}'^{\mathcal{G},\mathcal{SO}_{\mathcal{P}}}(\mathsf{crs}, \mathbb{x}, \mathbb{w})$.
- $\mathcal{V}^{\mathcal{G}}(\mathsf{crs}, \mathbb{x}, [\Pi]_1)$:
  1. Lazily sample a random oracle $\mathcal{SO}_{\mathcal{V}}$.
  2. If $\mathcal{V}'^{\mathcal{G},\mathcal{SO}_{\mathcal{V}}}(\mathsf{crs}, \mathbb{x}, [\Pi]_1) = 1$ then output 1.
  3. Otherwise, output 0.

The fact that soundness is preserved follows immediately from the verifier having the exact same distribution as in $(\mathcal{S}', \mathcal{P}', \mathcal{V}')$. Further, any adversary can not predict the response to a simulated random oralce query the verifier does because they are randomly sampled on the spot. We thus turn to proving completeness:

Claim 6.10. *$(\mathcal{S}, \mathcal{P}, \mathcal{V})$ has completeness error at most $\alpha + 1 - 1/\mathrm{poly}(\lambda)$.*

Proof. Fix an instance $\mathbb{x}$ and a witness $\mathbb{w}$. We prove that

$$\begin{aligned}
c &= \Pr_{\mathsf{r}_{\mathcal{V}},\mathsf{r}_{\mathcal{P}}}\left[\mathcal{V}^{\mathcal{G}}(\mathsf{crs},\mathbb{x},[\Pi]_1;\mathsf{r}_{\mathcal{V}})=1 \,\middle|\, \begin{array}{l}\mathsf{crs}\leftarrow\mathcal{S}^{\mathcal{G}}(1^\lambda),\\ [\Pi]_1\leftarrow\mathcal{P}^{\mathcal{G}}(\mathsf{crs},\mathbb{x},\mathbb{w};\mathsf{r}_{\mathcal{P}})\end{array}\right]\\
&= \Pr_{\mathcal{SO}_{\mathcal{V}},\mathcal{SO}_{\mathcal{P}}}\left[b=1 \,\middle|\, \begin{array}{l}\mathsf{crs}\leftarrow\mathcal{S}'^{\mathcal{G}}(1^\lambda),\\ [\Pi]_1\leftarrow\mathcal{P}'^{\mathcal{G},\mathcal{SO}_{\mathcal{P}}}(\mathsf{crs},\mathbb{x},\mathbb{w})\\ b\leftarrow\mathcal{V}'^{\mathcal{G},\mathcal{SO}_{\mathcal{V}}}(\mathsf{crs},\mathbb{x},[\Pi]_1)\end{array}\right]\\
&\geq 1/\mathrm{poly}(\lambda)
\end{aligned}$$

We define p_{cq} to be the probability of prover and verifier having a common query.

$$\begin{aligned}
p_{\mathsf{cq}} &= \Pr_{\mathcal{SO}_{\mathcal{V}},\mathcal{SO}_{\mathcal{P}}}\left[[\mathcal{X}_{\mathcal{P}}]_*\cap[\mathcal{X}_{\mathcal{V}}]_*\neq\emptyset \,\middle|\, \begin{array}{l}\mathsf{crs}\leftarrow\mathcal{S}'^{\mathcal{G}}(1^\lambda),\\ [\Pi]_1\leftarrow\mathcal{P}'^{\mathcal{G},\mathcal{SO}_{\mathcal{P}}}(\mathsf{crs},\mathbb{x},\mathbb{w})\text{ with }\mathcal{SO}_{\mathcal{P}}\text{-queries }[\mathcal{X}_{\mathcal{P}}]_*.\\ \mathcal{V}'^{\mathcal{G},\mathcal{SO}_{\mathcal{V}}}(\mathsf{crs},\mathbb{x},[\Pi]_1)\text{ with }\mathcal{SO}_{\mathcal{V}}\text{-queries }[\mathcal{X}_{\mathcal{V}}]_*.\end{array}\right]\\
&= \Pr_{\mathcal{SO}}\left[[\mathcal{X}_{\mathcal{P}}]_*\cap[\mathcal{X}_{\mathcal{V}}]_*\neq\emptyset \,\middle|\, \begin{array}{l}\mathsf{crs}\leftarrow\mathcal{S}'^{\mathcal{G}}(1^\lambda),\\ [\Pi]_1\leftarrow\mathcal{P}'^{\mathcal{G},\mathcal{SO}}(\mathsf{crs},\mathbb{x},\mathbb{w})\text{ with }\mathcal{SO}\text{-queries }[\mathcal{X}_{\mathcal{P}}]_*.\\ \mathcal{V}'^{\mathcal{G},\mathcal{SO}}(\mathsf{crs},\mathbb{x},[\Pi]_1)\text{ with }\mathcal{SO}\text{-queries }[\mathcal{X}_{\mathcal{V}}]_*.\end{array}\right]
\end{aligned}$$

This follows from the verifier only making non-adaptive queries, which means that the queries $[\mathcal{X}_{\mathcal{P}}]_*$ and $[\mathcal{X}_{\mathcal{V}}]_*$ only depend on $\mathcal{SO}_{\mathcal{P}}$.

First, we look at the case that it is probable that prover and verifier do not share queries. Specifically, if $p_{\mathsf{cq}} < 1/2$ we get that

$$\begin{aligned}
&\Pr_{\mathcal{SO}_{\mathcal{V}},\mathcal{SO}_{\mathcal{P}}}\left[b=1 \,\middle|\, \begin{array}{l}\mathsf{crs}\leftarrow\mathcal{S}'^{\mathcal{G}}(1^\lambda),\\ [\Pi]_1\leftarrow\mathcal{P}'^{\mathcal{G},\mathcal{SO}_{\mathcal{P}}}(\mathsf{crs},\mathbb{x},\mathbb{w})\\ b\leftarrow\mathcal{V}'^{\mathcal{G},\mathcal{SO}_{\mathcal{V}}}(\mathsf{crs},\mathbb{x},[\Pi]_1)\end{array}\right]\\
&\geq \Pr_{\mathcal{SO}_{\mathcal{V}},\mathcal{SO}_{\mathcal{P}}}\left[\begin{array}{l}b=1,\\ [\mathcal{X}_{\mathcal{P}}]_*\cap[\mathcal{X}_{\mathcal{V}}]_*=\emptyset\end{array} \,\middle|\, \begin{array}{l}\mathsf{crs}\leftarrow\mathcal{S}'^{\mathcal{G}}(1^\lambda),\\ [\Pi]_1\leftarrow\mathcal{P}'^{\mathcal{G},\mathcal{SO}_{\mathcal{P}}}(\mathsf{crs},\mathbb{x},\mathbb{w})\text{ with }\mathcal{SO}_{\mathcal{P}}\text{-queries }[\mathcal{X}_{\mathcal{P}}]_*.\\ b\leftarrow\mathcal{V}'^{\mathcal{G},\mathcal{SO}_{\mathcal{V}}}(\mathsf{crs},\mathbb{x},[\Pi]_1)\text{ with }\mathcal{SO}_{\mathcal{V}}\text{-queries }[\mathcal{X}_{\mathcal{V}}]_*.\end{array}\right]\\
&= \Pr_{\mathcal{SO}_{\mathcal{V}},\mathcal{SO}_{\mathcal{P}}}\left[b=1 \,\middle|\, \begin{array}{l}\mathsf{crs}\leftarrow\mathcal{S}'^{\mathcal{G}}(1^\lambda),\\ [\Pi]_1\leftarrow\mathcal{P}'^{\mathcal{G},\mathcal{SO}_{\mathcal{P}}}(\mathsf{crs},\mathbb{x},\mathbb{w})\text{ with }\mathcal{SO}_{\mathcal{P}}\text{-queries }[\mathcal{X}_{\mathcal{P}}]_*.\\ b\leftarrow\mathcal{V}'^{\mathcal{G},\mathcal{SO}_{\mathcal{V}}}(\mathsf{crs},\mathbb{x},[\Pi]_1)\text{ with }\mathcal{SO}_{\mathcal{V}}\text{-queries }[\mathcal{X}_{\mathcal{V}}]_*.\\ [\mathcal{X}_{\mathcal{P}}]_*\cap[\mathcal{X}_{\mathcal{V}}]_*=\emptyset\end{array}\right]\cdot(1-p_{\mathsf{cq}})\\
&\geq \Pr_{\mathcal{SO}}\left[b=1 \,\middle|\, \begin{array}{l}\mathsf{crs}\leftarrow\mathcal{S}'^{\mathcal{G}}(1^\lambda),\\ [\Pi]_1\leftarrow\mathcal{P}'^{\mathcal{G},\mathcal{SO}}(\mathsf{crs},\mathbb{x},\mathbb{w})\text{ with }\mathcal{SO}\text{-queries }[\mathcal{X}_{\mathcal{P}}]_*.\\ b\leftarrow\mathcal{V}'^{\mathcal{G},\mathcal{SO}}(\mathsf{crs},\mathbb{x},[\Pi]_1)\text{ with }\mathcal{SO}\text{-queries }[\mathcal{X}_{\mathcal{V}}]_*.\\ [\mathcal{X}_{\mathcal{P}}]_*\cap[\mathcal{X}_{\mathcal{V}}]_*=\emptyset\end{array}\right]\cdot(1-p_{\mathsf{cq}}) \qquad (1)\\
&= \Pr_{\mathcal{SO}}\left[\begin{array}{l}b\neq 0,\\ [\mathcal{X}_{\mathcal{P}}]_*\cap[\mathcal{X}_{\mathcal{V}}]_*=\emptyset\end{array} \,\middle|\, \begin{array}{l}\mathsf{crs}\leftarrow\mathcal{S}'^{\mathcal{G}}(1^\lambda),\\ [\Pi]_1\leftarrow\mathcal{P}'^{\mathcal{G},\mathcal{SO}}(\mathsf{crs},\mathbb{x},\mathbb{w})\text{ with }\mathcal{SO}\text{-queries }[\mathcal{X}_{\mathcal{P}}]_*.\\ b\leftarrow\mathcal{V}'^{\mathcal{G},\mathcal{SO}}(\mathsf{crs},\mathbb{x},[\Pi]_1)\text{ with }\mathcal{SO}\text{-queries }[\mathcal{X}_{\mathcal{V}}]_*.\end{array}\right]\\
&\geq 1-\alpha-p_{\mathsf{cq}} \qquad (2)\\
&> 1/4 > 1/\mathrm{poly}(\lambda)-\alpha
\end{aligned}$$

Equality 1 follows from the fact that if prover and verifier queries are distinct then the $\mathcal{SO}_\mathcal{P}$ and $\mathcal{SO}_\mathcal{V}$ offer a consistent view of a random oracle. Inequality 2 follows by union bound over completeness of $(\mathcal{S}', \mathcal{P}', \mathcal{V}')$ and the event that $[\mathcal{X}_\mathcal{P}]_* \cap [\mathcal{X}_\mathcal{V}]_* = \emptyset$.

For the rest of the proof we consider the case that $p_{\mathsf{cq}} \geq 1/2$. With $T_\mathcal{P}$ (respective $T_\mathcal{V}$) being the number of ROM queries of the prover (respective verifier) there exist an $i^* \in [T_\mathcal{P}]$ and $j^* \in [T_\mathcal{V}]$ such that

$$
\begin{aligned}
& p_{i^*,j^*} && (3)\\
&= \Pr_{\mathcal{SO}_\mathcal{P},\mathcal{SO}_\mathcal{V}}\left[\begin{array}{c|l} [X_{\mathcal{P},i^*}]_* = [X_{\mathcal{V},j^*}]_*, & \mathsf{crs} \leftarrow \mathcal{S}'^{\mathcal{G}}(1^\lambda), \\ \forall_{i<i^*}, j \in [T_\mathcal{V}] : [X_{\mathcal{P},i}]_* \neq [X_{\mathcal{V},j}]_* & [\Pi]_1 \leftarrow \mathcal{P}'^{\mathcal{G},\mathcal{SO}_\mathcal{P}}(\mathsf{crs}, \mathbb{x}, \mathbb{w}) \\ & \quad \text{where } [X_{\mathcal{P},i}]_* \text{ is the } i\text{-th } \mathcal{SO}_\mathcal{P}\text{-query.} \\ & b \leftarrow \mathcal{V}'^{\mathcal{G},\mathcal{SO}_\mathcal{V}}(\mathsf{crs}, \mathbb{x}, [\Pi]_1) \\ & \quad \text{where } [X_{\mathcal{V},j}]_* \text{ is the } j\text{-th } \mathcal{SO}_\mathcal{V}\text{-query.} \end{array}\right] \\
&= \Pr_{\mathcal{SO}}\left[\begin{array}{c|l} [X_{\mathcal{P},i^*}]_* = [X_{\mathcal{V},j^*}]_*, & \mathsf{crs} \leftarrow \mathcal{S}'^{\mathcal{G}}(1^\lambda), \\ \forall_{i<i^*,j\in[T_\mathcal{V}]} : [X_{\mathcal{P},i}]_* \neq [X_{\mathcal{V},j}]_* & [\Pi]_1 \leftarrow \mathcal{P}'^{\mathcal{G},\mathcal{SO}}(\mathsf{crs}, \mathbb{x}, \mathbb{w}) \\ & \quad \text{where } [X_{\mathcal{P},i}]_* \text{ is the } i\text{-th } \mathcal{SO}\text{-query.} \\ & b \leftarrow \mathcal{V}'^{\mathcal{G},\mathcal{SO}}(\mathsf{crs}, \mathbb{x}, [\Pi]_1) \\ & \quad \text{where } [X_{\mathcal{V},j}]_* \text{ is the } j\text{-th } \mathcal{SO}\text{-query.} \end{array}\right] && (4)\\
&\geq 1/2T_\mathcal{P}T_\mathcal{V}. && (5)
\end{aligned}
$$

Equality 4 follows from the verifier only making non-adaptive queries, which means that the queries $X_{\mathcal{P},i}$ and $X_{\mathcal{V},j}$ only depend on $\mathcal{SO}_\mathcal{P}$. Inequality 5 follows from an averaging argument and $\mathcal{X}_\mathcal{P} \cap \mathcal{X}_\mathcal{V}$ being equivalent to the statement $\exists i^*, j^* : X_{\mathcal{P},i^*} = X_{\mathcal{V},j^*}$ and $\forall_{i<i^*}, j \in [T_\mathcal{V}] : X_{\mathcal{P},i} \neq X_{\mathcal{V},j}$.

Let n be the upper bound on how many elements a ROM query by the verifier takes as input. Each of query j of the $T_\mathcal{V}$ queries is defined by n polynomials over crs and Π. We call these polynomials $(f_{j,k}(\hat{\Pi}, \hat{\mathsf{crs}}))_{j\in[T_\mathcal{V}],k\in[n]}$. For each $j \in [T_\mathcal{V}]$ and $k \in [n]$ with $\mathsf{IsIndependent}(f_{j,k}(\hat{\Pi}, \hat{\mathsf{crs}})) = 0$ we let $f^{-1}_{j,k,\mathsf{crs}}$ be the function derived by Claim 6.6 (i.e., such that for every crs it holds that $f^{-1}_{j,k,\mathsf{crs}}(f_{j,k}(\hat{\Pi}, \mathsf{crs})) = \hat{\Pi}$).

We use these functions in the following analysis of the accepting probability. With the rules of probability we get

$$\Pr_{\mathcal{SO}_\mathcal{V},\mathcal{SO}_\mathcal{P}}\left[b=1 \;\middle|\; \begin{array}{l} \mathsf{crs} \leftarrow \mathcal{S}'^{\mathcal{G}}(1^\lambda), \\ [\Pi]_1 \leftarrow \mathcal{P}'^{\mathcal{G},\mathcal{SO}_\mathcal{P}}(\mathsf{crs},\mathbb{x},\mathbb{w}), \\ b \leftarrow \mathcal{V}'^{\mathcal{G},\mathcal{SO}_\mathcal{V}}(\mathsf{crs},\mathbb{x},[\Pi]_1) \end{array}\right]$$

$$\geq \Pr_{\mathcal{SO}_\mathcal{V},\mathcal{SO}_\mathcal{P}}\left[\begin{array}{l} b=1, \\ [X_{\mathcal{P},i^*}]_* = [X_{\mathcal{V},j^*}]_*, \\ \forall_{i<i^*,j\in[T_\mathcal{V}]} : [X_{\mathcal{P},i}]_* \neq [X_\mathcal{V}]_* \end{array} \;\middle|\; \begin{array}{l} \mathsf{crs} \leftarrow \mathcal{S}'^{\mathcal{G}}(1^\lambda), \\ [\Pi]_1 \leftarrow \mathcal{P}'^{\mathcal{G},\mathcal{SO}_\mathcal{P}}(\mathsf{crs},\mathbb{x},\mathbb{w}) \\ \quad \text{where } [X_{\mathcal{P},i}]_* \text{ is the } i\text{-th } \mathcal{SO}_\mathcal{P}\text{-query}, \\ b \leftarrow \mathcal{V}'^{\mathcal{G},\mathcal{SO}_\mathcal{V}}(\mathsf{crs},\mathbb{x},[\Pi]_1) \\ \quad \text{where } [X_{\mathcal{V},j}]_* \text{ is the } j\text{-th } \mathcal{SO}_\mathcal{V}\text{-query} \end{array}\right]$$

$$= \Pr_{\mathcal{SO}_\mathcal{V},\mathcal{SO}_\mathcal{P}}\left[b=1 \;\middle|\; \begin{array}{l} \mathsf{crs} \leftarrow \mathcal{S}'^{\mathcal{G}}(1^\lambda), \\ [\Pi]_1 \leftarrow \mathcal{P}'^{\mathcal{G},\mathcal{SO}_\mathcal{P}}(\mathsf{crs},\mathbb{x},\mathbb{w}) \\ \quad \text{where } [X_{\mathcal{P},i}]_* \text{ is the } i\text{-th } \mathcal{SO}_\mathcal{P}\text{-query}, \\ b \leftarrow \mathcal{V}'^{\mathcal{G},\mathcal{SO}_\mathcal{V}}(\mathsf{crs},\mathbb{x},[\Pi]_1) \\ \quad \text{where } [X_{\mathcal{V},j}]_* \text{ is the } j\text{-th } \mathcal{SO}_\mathcal{V}\text{-query}, \\ [X_{\mathcal{P},i^*}]_* = [X_{\mathcal{V},j^*}]_*, \\ \forall_{i<i^*,j\in[T_\mathcal{V}]} : [X_{\mathcal{P},i}]_* \neq [X_{\mathcal{V},j}]_* \end{array}\right] \cdot (1-p_{i^*,j^*})$$

We introduce some notation. By $\mathsf{st} \leftarrow \mathcal{P}'^{\mathcal{G},\mathcal{SO}_\mathcal{P}[0,i^*)}(\mathsf{crs},\mathbb{x},\mathbb{w})$ we mean run $\mathcal{P}'^{\mathcal{G},\mathcal{SO}_\mathcal{P}}(\mathsf{crs},\mathbb{x},\mathbb{w})$ until its i^*-th $\mathcal{SO}_\mathcal{P}$-query and let st be the state of computation it is in before receiving the response to the i^*-th $\mathcal{SO}_\mathcal{P}$-query. Similarly, we define $\mathcal{P}'^{\mathcal{G},\mathcal{SO}_\mathcal{P}[i^*,T_\mathcal{P}]}(\mathsf{crs},\mathbb{x},\mathbb{w},\mathsf{st})$ is the continuation of the above mentioned execution until the end.

For $\mathsf{st} \leftarrow \mathcal{P}'^{\mathcal{G},\mathcal{SO}_\mathcal{P}[0,i^*)}(\mathsf{crs},\mathbb{x},\mathbb{w})$ we have $\mathcal{P}'^{\mathcal{G},\mathcal{SO}_\mathcal{P}[i^*,T_\mathcal{P}]}(\mathsf{crs},\mathbb{x},\mathbb{w},\mathsf{st}) = \Pi = f^{-1}_{j^*,k^*,\mathsf{crs}}(X_{\mathcal{V},j^*}[k^*])$ and $X_{\mathcal{V},j^*} = X_{\mathcal{P},i^*}$. Therefore, we can also compute Π from $f^{-1}_{j^*,\mathsf{crs}}(X_{\mathcal{P},i^*}[k^*])$.

$$= \Pr_{\mathcal{SO}_\mathcal{V},\mathcal{SO}_\mathcal{P}}\left[b=1 \;\middle|\; \begin{array}{l} \mathsf{crs} \leftarrow \mathcal{S}'^{\mathcal{G}}(1^\lambda), \\ \text{Run } \mathcal{P}'^{\mathcal{G},\mathcal{SO}_\mathcal{P}[0,i^*)}(\mathsf{crs},\mathbb{x},\mathbb{w}) \text{ where } [X_{\mathcal{P},i}]_* \text{ is the } i\text{-th } \mathcal{SO}_\mathcal{P}\text{-query}, \\ \text{Let } k^* \text{ be the smallest index s.t. } \mathsf{IsIndependent}(f_{j^*,k^*}) = 1, \\ \Pi \leftarrow f^{-1}_{j^*,k^*,\mathsf{crs}}(X_{\mathcal{P},i^*}[k^*]), \\ b \leftarrow \mathcal{V}'^{\mathcal{G},\mathcal{SO}_\mathcal{V}}(\mathsf{crs},\mathbb{x},[\Pi]_1) \text{ where } [X_{\mathcal{V},j}]_* \text{ is the } j\text{-th } \mathcal{SO}_\mathcal{V}\text{-query}, \\ [X_{\mathcal{P},i^*}]_* = [X_{\mathcal{V},j^*}]_*, \\ \forall_{i<i^*,j\in[T_\mathcal{V}]} : [X_{\mathcal{P},i}]_* \neq [X_{\mathcal{V},j}]_* \end{array}\right] \cdot (1-p_{i^*,j^*})$$

Now, we have that the queries on $\mathcal{SO}_\mathcal{V}$ and $\mathcal{SO}_\mathcal{P}$ are entirely distinct. Therefore, if $\mathcal{V}'$ uses $\mathcal{SO}_\mathcal{P}$ the probabilities are identical.

$$= \Pr_{\mathcal{SO}_\mathcal{P}}\left[b=1 \;\middle|\; \begin{array}{l} \mathsf{crs} \leftarrow \mathcal{S}'^{\mathcal{G}}(1^\lambda), \\ \text{Run } \mathsf{st} \leftarrow \mathcal{P}'^{\mathcal{G},\mathcal{SO}_\mathcal{P}[0,i^*)}(\mathsf{crs},\mathbb{x},\mathbb{w}) \text{ where } [X_{\mathcal{P},i}]_* \text{ is the } i\text{-th } \mathcal{SO}_\mathcal{P}\text{-query}, \\ \text{Let } k^* \text{ be the smallest index s.t. } \mathsf{IsIndependent}(f_{j^*,k^*}) = 1, \\ \Pi \leftarrow f^{-1}_{j^*,k^*,\mathsf{crs}}(X_{\mathcal{P},i^*}[k^*]), \\ b \leftarrow \mathcal{V}'^{\mathcal{G},\mathcal{SO}_\mathcal{P}}(\mathsf{crs},\mathbb{x},[\Pi]_1) \text{ where } [X_{\mathcal{V},j}]_* \text{ is the } j\text{-th } \mathcal{SO}_\mathcal{P}\text{-query}, \\ [X_{\mathcal{P},i^*}]_* = [X_{\mathcal{V},j^*}]_*, \\ \forall_{i<i^*,j\in[T_\mathcal{V}]} : [X_{\mathcal{P},i}]_* \neq [X_{\mathcal{V},j}]_* \end{array}\right] \cdot (1-p_{i^*,j^*})$$

We undo a previous step and compute Π using $\mathcal{P}'^{\mathcal{G},\mathcal{SO}_\mathcal{P}[i^*,T_\mathcal{P}]}(\mathsf{crs},\mathbb{x},\mathbb{w},\mathsf{st})$. We can do this because we have that $[X_{\mathcal{V},j^*}]_* = [X_{\mathcal{P},i^*}]_*$.

$$= \Pr_{\mathcal{SO}}\left[b=1 \;\middle|\; \begin{array}{l} \mathsf{crs} \leftarrow \mathcal{S}'^{\mathcal{G}}(1^\lambda), \\ [\Pi]_1 \leftarrow \mathcal{P}'^{\mathcal{G},\mathcal{SO}}(\mathsf{crs}, \mathbb{x}, \mathbb{w}) \text{ where } [X_{\mathcal{P},i}]_* \text{ is the } i\text{-th } \mathcal{SO}\text{-query}, \\ b \leftarrow \mathcal{V}'^{\mathcal{G},\mathcal{SO}}(\mathsf{crs}, \mathbb{x}, [\Pi]_1) \text{ where } [X_{\mathcal{V},j}]_* \text{ is the } j\text{-th } \mathcal{SO}\text{-query}, \\ [X_{\mathcal{P},i^*}]_* = [X_{\mathcal{V},j^*}]_*, \\ \forall_{i<i^*, j\in[T_\mathcal{V}]} : [X_{\mathcal{P},i}]_* \neq [X_{\mathcal{V},j}]_* \end{array} \right] \cdot (1 - p_{i^*,j^*})$$

We use equality 4 to combine the probabilities.

$$\begin{aligned} &= \Pr_{\mathcal{SO}}\left[\begin{array}{l} b \neq 0, \\ [X_{\mathcal{P},i^*}]_* = [X_{\mathcal{V},j^*}]_*, \\ \forall_{i<i^*, j\in[T_\mathcal{V}]} : [X_{\mathcal{P},i}]_* \neq [X_\mathcal{V}]_* \end{array} \;\middle|\; \begin{array}{l} \mathsf{crs} \leftarrow \mathcal{S}'^{\mathcal{G}}(1^\lambda), \\ [\Pi]_1 \leftarrow \mathcal{P}'^{\mathcal{G},\mathcal{SO}}(\mathsf{crs}, \mathbb{x}, \mathbb{w}) \\ \quad \text{where } [X_{\mathcal{P},i}]_* \text{ is the } i\text{-th } \mathcal{SO}\text{-query.} \\ b \leftarrow \mathcal{V}'^{\mathcal{G},\mathcal{SO}}(\mathsf{crs}, \mathbb{x}, [\Pi]_1) \\ \quad \text{where } [X_{\mathcal{V},j}]_* \text{ is the } j\text{-th } \mathcal{SO}\text{-query.} \end{array} \right] \\ &\geq 1 - \alpha - (1 - 1/2T_\mathcal{P}T_\mathcal{V}) \\ &= 1/\mathrm{poly}(\lambda) - \alpha \end{aligned}$$

These follow by union bound over completeness of $(\mathcal{S}', \mathcal{P}', \mathcal{V}')$ and the event that $[X_{\mathcal{P},i^*}]_* = [X_{\mathcal{V},j^*}]_* \wedge \forall_{i<i^*, j\in[T_\mathcal{V}]} : [X_{\mathcal{P},i}]_* \neq [X_\mathcal{V}]_*$. □

□

6.2 Ruling out one group element SNARGs in the pure GGM

In this section, we prove that if one-way functions exist, a one-group element SNARG in Maurer's generic group model does not exist. The proof is a refinement of Groth's lower bound [Gro16]. He proved that if one-way functions exist, then there does not exist a SNARG where the proof consists of only $\mathbb{G}_1$ group elements, and the verifier has a particular structure. We focus on the setting where the proof is just one group element. This change in focus allows us to cover arbitrary verifiers in Maurer's generic group model.

Lemma 6.11. *If one-way functions exist (in the plain model) then there is no SNARG for* NP *in the Maurer Bilinear GGM with argument size one group element, completeness error* $\alpha = 1 - 1/\mathrm{poly}(\lambda)$*, and soundness error* $\beta = \mathsf{negl}(\lambda)$.

Proof. We prove the statement by contraposition. Assume $(\mathcal{S}, \mathcal{P}, \mathcal{V})$ is a SNARG in the Maurer Bilinear GGM as in the lemma statement for the NP relation $\mathcal{R} = \{(\mathsf{G}(s), s)\}$ where $\mathsf{G}\colon \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$ is a PRG. Let L be the language derived from $\mathcal{R}$.

In this proof we will assume that the proof is in $\mathbb{G}_1$, however proving the lemma for $\mathbb{G}_2$ and $\mathbb{G}_T$ is analagous, the only changes accounting to the slightly different structure of $\mathbb{G}_2$ and $\mathbb{G}_T$ elements. Since the proof is a $\mathbb{G}_1$ element, the prover will output some group element $[\Pi]_1$ that is computed as some function $f(\hat{\mathsf{crs}}_1) = \langle \mathbf{\Pi}, \hat{\mathsf{crs}}_1 \rangle$ where $\mathbf{\Pi}$ is a vector of $\mathbb{Z}_p$ elements. This means each verifier pairing equation check can be written as

$$\begin{aligned} &(d_i \cdot [\Pi]_1) \cdot \langle \mathbf{a}'_i, [\mathsf{crs}_2]_2 \rangle + \langle \mathbf{b}_i, [\mathsf{crs}_1]_1 \otimes [\mathsf{crs}_2]_2 \rangle + \langle \mathbf{c}_i, [\mathsf{crs}_T]_T \rangle = [0]_T \\ \Leftrightarrow &[\Pi]_1 \cdot \langle \mathbf{a}_i, [\mathsf{crs}_2]_2 \rangle + \langle \mathbf{b}_i, [\mathsf{crs}_1]_1 \otimes [\mathsf{crs}_2]_2 \rangle + \langle \mathbf{c}_i, [\mathsf{crs}_T]_T \rangle = [0]_T \end{aligned}$$

for some $\mathbb{Z}_p$ vectors $\mathbf{a}_i$, $\mathbf{a}'_i$, $\mathbf{b}_i$, $\mathbf{c}_i$, and $\mathbb{Z}_p$ elements d_i.

Using this proof system we show how to break the security game of the PRG: i.e., we construct a distinguisher $\mathcal{A}$ for the distributions

$$\mathcal{D}_{\mathsf{G}} := \left\{ \mathbb{x} \middle| \begin{array}{c} s \leftarrow \{0,1\}^\lambda \\ \mathbb{x} \leftarrow \mathsf{G}(s) \end{array} \right\}, \quad \mathcal{D}_U := \left\{ \mathbb{x} \middle| \ \mathbb{x} \leftarrow \{0,1\}^{2\lambda} \ \right\}.$$

Note that $\mathcal{D}_{\mathsf{G}}$ always outputs $\mathbb{x} \in L$ and that this happens only with negligible probability for $\mathcal{D}_U$. We describe first an algorithm $\mathsf{SolutionFinder}$, and then use it to specify the distinguisher $\mathcal{A}$:

- $\mathsf{SolutionFinder}^{\mathcal{G}}(\mathsf{crs}, \mathbf{a}, \mathbf{b}, \mathbf{c})$:

```
1. Let $[\mathsf{crs}_1]_1$ (resp. $[\mathsf{crs}_2]_2$ and $[\mathsf{crs}_T]_T$) be the group elements in $\mathsf{crs}$ which are $\mathbb{G}_1$ elements (resp. $\mathbb{G}_2$ and $\mathbb{G}_T$).
2. Let $\mathsf{counter} \leftarrow 0$.
3. Let $\mathcal{B} \leftarrow \emptyset$.
4. While $\mathsf{counter} < \lambda/(1-\alpha)$:
   (a) Sample $s \leftarrow \{0,1\}^\lambda$ uniformly at random.
   (b) Run $[\Pi]_1 \leftarrow \mathcal{P}^{\mathcal{G}}(\mathsf{crs}, \mathbb{x} := \mathsf{G}(s), \mathbb{w} := s)$ where $\mathbf{\Pi}$ is the $\mathbb{Z}_p$-vector the prover uses to compute $[\Pi]_1 \leftarrow \langle \mathbf{\Pi}, [\mathsf{crs}_1]_1 \rangle$.
   (c) Let $\mathcal{I} \leftarrow \emptyset$.
   (d) Run $\mathcal{V}^{\mathcal{G}}(\mathsf{crs}, \mathbb{x} := \mathsf{G}(s), [\Pi]_1)$ where for $i \in [T_{\mathcal{V}}]$ we have $\mathbf{a}'_i$, $\mathbf{b}'_i$, and $\mathbf{c}'_i$ are the $\mathbb{Z}_p$ vectors such that the $i$-th query of the verifier checks the equation
       $$[\Pi]_1 \cdot \langle \mathbf{a}'_i, [\mathsf{crs}_2]_2 \rangle + \langle \mathbf{b}'_i, [\mathsf{crs}_1]_1 \otimes [\mathsf{crs}_2]_2 \rangle + \langle \mathbf{c}'_i, [\mathsf{crs}_T]_T \rangle \stackrel{?}{=} [0]_T.$$
       If the equations holds let $\mathcal{I} \leftarrow \mathcal{I} \cup \{i\}$.
   (e) For $i \in \mathcal{I}$:
       i. Let
          $$\mathbf{v} \leftarrow \begin{pmatrix} \mathbf{\Pi} \otimes \mathbf{a}'_i \\ \mathbf{b}'_i \\ \mathbf{c}'_i \end{pmatrix}$$
       ii. If $\mathbf{v} \notin \mathsf{span}(\mathcal{B})$ let $\mathcal{B} \leftarrow \mathcal{B} \cup \{\mathbf{v}\}$ and $\mathsf{counter} \leftarrow 0$.
   (f) Let $\mathsf{counter} \leftarrow \mathsf{counter} + 1$.
   (g) If there exists a $\mathbf{\Pi}$ such that $\begin{pmatrix} \mathbf{\Pi} \otimes \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \end{pmatrix} \in \mathsf{span}(\mathcal{B})$ then output such a uniformly random $\mathbf{\Pi}$ otherwise output $\perp$.
```

- $\mathcal{A}(\mathbb{x}^*)$:

```
1. Simulate a generic group under the name of $\mathcal{G}$.
2. Sample $\mathsf{crs} \leftarrow \mathcal{S}^{\mathcal{G}}(1^\lambda)$.
3. Simulate $b^* \leftarrow \mathcal{V}^{\mathcal{G}}(\mathbb{x}^*, [\Pi]_1 := [1])$ on the $i$-th pairing equation check respond as follows:
   (a) Let the $i$-th pairing equation check be
       $$[\Pi]_1 \cdot \langle \mathbf{a}_i, [\mathsf{crs}_2]_2 \rangle + \langle \mathbf{b}_i, [\mathsf{crs}_1]_1 \otimes [\mathsf{crs}_2]_2 \rangle + \langle \mathbf{c}_i, [\mathsf{crs}_T]_T \rangle \stackrel{?}{=} [0]_T.$$
```

(b) If $\langle \mathbf{a}_i, [\mathsf{crs}_2]_2 \rangle = [0]_2$ respond with 1 if $\langle \mathbf{b}_i, [\mathsf{crs}_1]_1 \otimes [\mathsf{crs}_2]_2 \rangle + \langle \mathbf{c}_i, [\mathsf{crs}_T]_T \rangle = [0]_T$ and 0 otherwise.
(c) Let $\mathbf{\Pi}^* \leftarrow \mathsf{SolutionFinder}^{\mathcal{G}}(\mathsf{crs}, (\mathbf{a}_i, \mathbf{b}_i, \mathbf{c}_i))$.
(d) If $\mathcal{V}^{\mathcal{G}}(\mathbb{x}, \langle \mathbf{\Pi}, [\mathsf{crs}_1]_1 \rangle) = 1$ then $\mathcal{A}$ outputs 1. Otherwise respond to the query with 0.

4. Output 0.

First, we argue that the adversary terminates in polynomial time. Because the dimension of $\begin{pmatrix} \mathbf{\Pi} \otimes \mathbf{a}_i \\ \mathbf{b}_i \\ \mathbf{c}_i \end{pmatrix}$ is polynomial in the security parameter, $\mathsf{counter}$ can only be reset to 0 a polynomial number of times and the loop in $\mathsf{SolutionFinder}$ is run $2\lambda/(1-\alpha) = \text{poly}(\lambda)$ many times.

For each i we define a linear space $S := \{(\mathbf{\Pi} \otimes \mathbf{a}) \mid \mathbf{\Pi} \in \mathbb{F}^{|\mathsf{crs}_1|}\}$ and an affine space

$$B := \left\{ x \;\middle|\; \begin{pmatrix} x \\ \mathbf{b} \\ \mathbf{c} \end{pmatrix} \in \mathsf{span}(\mathcal{B}) \right\}.$$

A $\mathbf{\Pi}$ meets all the conditions if and only if $\mathbf{\Pi} \in S \cap B$. Therefore it is sufficient to compute said intersection and sample from it.

We analyse how the adversary's output behaves on samples from the PRG versus uniformly random samples:

Claim 6.12. *If $\mathbb{x} \notin L$ the adversary $\mathcal{A}$ outputs 0 with probability $1 - \mathsf{negl}(\lambda)$.*

Proof. We define a new adversary, this time for the SNARG:

$\mathcal{A}_S^{\mathcal{G}}(\mathsf{crs}, \mathbb{x})$:

1. Simulate $b^* \leftarrow \mathcal{V}^{\mathcal{G}}(\mathbb{x}, [\Pi]_1 := [1])$ on the i-th pairing equation check respond as follows:
   (a) Let the i-th pairing equation check be

$$[\Pi]_1 \cdot \langle \mathbf{a}_i, [\mathsf{crs}_2]_2 \rangle + \langle \mathbf{b}_i, [\mathsf{crs}_1]_1 \otimes [\mathsf{crs}_2]_2 \rangle + \langle \mathbf{c}_i, [\mathsf{crs}_T]_T \rangle \overset{?}{=} [0]_T.$$

   (b) If $\langle \mathbf{a}_i, [\mathsf{crs}_2]_2 \rangle = [0]_2$ respond with 1 if $\langle \mathbf{b}_i, [\mathsf{crs}_1]_1 \otimes [\mathsf{crs}_2]_2 \rangle + \langle \mathbf{c}_i, [\mathsf{crs}_T]_T \rangle = [0]_T$ and 0 otherwise.
   (c) Let $\mathbf{\Pi}^* \leftarrow \mathsf{SolutionFinder}^{\mathcal{G}}(\mathsf{crs}, (\mathbf{a}_i, \mathbf{b}_i, \mathbf{c}_i))$.
   (d) If $\mathcal{V}^{\mathcal{G}}(\mathbb{x}, \langle \mathbf{\Pi}, [\mathsf{crs}_1]_1 \rangle) = 1$ then $\mathcal{A}_S$ outputs $\langle \mathbf{\Pi}, [\mathsf{crs}_1]_1 \rangle$. Otherwise respond to the query with 0.
2. Output 0.

Observe that $\mathcal{A}_S$ is identical to $\mathcal{A}$ except that it doesn't sample crs and it has access to a real oracle $\mathcal{G}$ rather than simulating it by itself. However crs in the SNARG soundness experiment is sampled as in $\mathcal{A}$, and $\mathcal{G}$ is exactly the same as well. Thus, if $\mathcal{A}$ outputs 1 then $\mathcal{A}_S$ outputs a proof $[\Pi]_1$ such that $\mathcal{V}(\mathbb{x}, [\Pi]_1) = 1$. By soundness we get the probability that $\mathcal{A}_S$ outputs a proof $[\Pi]_1$ with $\mathcal{V}(\mathbb{x}, \langle \mathbf{\Pi}, [\mathsf{crs}_1]_1 \rangle) = 1$ is $\beta = \mathsf{negl}(\lambda)$. It follows the probability that $\mathcal{A}(\mathbb{x})$ outputs 1 is $\mathsf{negl}(\lambda)$. □

Claim 6.13. *For $\mathbb{x} \leftarrow \mathcal{D}_{\mathsf{G}}$, the following occurs with negligible probability (over the choice of $\mathbb{x}$,* crs *and the verifier randomness): given $\mathbb{x}$ and* crs*, $\mathcal{V}$ accepts all proofs $[\Pi]_1$ where every pairing equation checks*

$$[\Pi]_1 \cdot \langle \mathbf{a}_i, [\mathsf{crs}_2]_2 \rangle + \langle \mathbf{b}_i, [\mathsf{crs}_1]_1 \otimes [\mathsf{crs}_2]_2 \rangle + \langle \mathbf{c}_i, [\mathsf{crs}_T]_T \rangle \overset{?}{=} [0]_T$$

with $\langle \mathbf{a}_i, [\mathsf{crs}_2]_2 \rangle \neq [0]_2$ in $\mathcal{V}^{\mathcal{G}}(\mathsf{crs}, \mathbb{x}, [\Pi]_1)$ are answered by 0*.*

Proof. We begin by moving from instances $\mathbb{x}$ sampled from $\mathcal{D}_{\mathsf{G}}$ to ones sampled from $\mathcal{D}_U$. Consider the following algorithm :

- $\mathcal{A}_{\mathsf{G}}(\mathbb{x})$:
  1. Simulate a generic group under the name of $\mathcal{G}$.
  2. Sample $\mathsf{crs} \leftarrow \mathcal{S}^{\mathcal{G}}(1^\lambda)$.
  3. Simulate $b \leftarrow \mathcal{V}^{\mathcal{G}}(\mathbb{x}, [\Pi]_1 := [1])$ where on the i-th pairing equation check respond as follows:
     (a) Let the i-th pairing equation check be
     $$[\Pi]_1 \cdot \langle \mathbf{a}_i, [\mathsf{crs}_2]_2 \rangle + \langle \mathbf{b}_i, [\mathsf{crs}_1]_1 \otimes [\mathsf{crs}_2]_2 \rangle + \langle \mathbf{c}_i, [\mathsf{crs}_T]_T \rangle \overset{?}{=} [0]_T.$$
     (b) If $\langle \mathbf{a}_i, [\mathsf{crs}_2]_2 \rangle = [0]_2$ respond with 1 if $\langle \mathbf{b}_i, [\mathsf{crs}_1]_1 \otimes [\mathsf{crs}_2]_2 \rangle + \langle \mathbf{c}_i, [\mathsf{crs}_T]_T \rangle = [0]_T$ and 0 otherwise.
     (c) Otherwise respond to the query with 0.
  4. Output b.

As we are in the Maurer GGM, without loss of generality, the view of $\mathcal{V}$ of the proof $[\Pi]_1$ is only in the pairing equation checks where $\langle \mathbf{a}_i, [\mathsf{crs}_2]_2 \rangle \neq [0]_2$ (if it makes an equality check in $\mathbb{G}_1$ then we can translate it into one in $\mathbb{G}_T$). Any the verifier's view when given $[\Pi]_1$ for which all of the are answered by 0 is identical to its view in $\mathcal{A}_{\mathsf{G}}(\mathbb{x})$. Thus the probability that $\mathcal{V}$ accepts all claims where the pariring equations output 0 is equal to the probability of $\mathcal{A}_{\mathsf{G}}$ ouptutting 1 for $\mathbb{x} \leftarrow \mathcal{D}_{\mathsf{G}}$.

By indistinguishability between $\mathcal{D}_{\mathsf{G}}$ and $\mathcal{D}_U$ and the fact that $\mathcal{A}_{\mathsf{G}}$ is efficient, we have that $\mathcal{A}_{\mathsf{G}}(\mathbb{x}) = 1$ with the same probability up to a $\mathsf{negl}(\lambda)$ factor when $\mathbb{x} \leftarrow \mathcal{D}_{\mathsf{G}}$ as $\mathbb{x} \leftarrow \mathcal{D}_U$.

We now show that $\mathcal{A}_{\mathsf{G}}(\mathbb{x}) = 1$ with negligible probability when $\mathbb{x} \leftarrow \mathcal{D}_U$, thus completing the proof. First, note that when sampled from this distribution, $\mathbb{x} \notin L$ with overwhelming probability. Thus we henceforth consider some fixed $\mathbb{x} \notin L$. Suppose towards contradiction that $\mathcal{A}_{\mathsf{G}}(\mathbb{x})$ outputs 1 with noticeable probability $p \geq 1/\text{poly}(\lambda)$. We show that this contradicts soundness of the SNARG by defining an adversary:

- $\mathcal{A}_0^{\mathcal{G}}(\mathsf{crs}, \mathbb{x})$: For $i \in [T_{\mathcal{V}} + 1]$ if $\mathcal{V}^{\mathcal{G}}(\mathsf{crs}, \mathbb{x}, [\Pi]_1 := [i]_1) = 1$ output $[i]_1$.

We show that $\mathcal{A}_0$ will win the soundness experiment with probability p. Consider a crs such that $\mathcal{A}_{\mathsf{G}}$ outputs 1, meaning that $\mathcal{V}$ accepts a proof $[\Pi]_1$ where all the relevant pairing equation checks in $\mathcal{V}^{\mathcal{G}}(\mathsf{crs}, \mathbb{x}, [\Pi]_1)$ return 0.

Each pairing equation check is a degree 1 polynomial in the proof $[\Pi]_1$. Therefore, by polynomial identity lemma, for every check there exists at most one $[\Pi]_1$ such that the pairing equation check outputs 1. Because the number of pairing equation checks the verifier is $T_{\mathcal{V}}$ it can reject at most $T_{\mathcal{V}}$ distinct group elements. $\mathcal{A}_0$ tests $T_{\mathcal{V}} + 1$ distinct group elements and at most $T_{\mathcal{V}}$ elements can be rejected by the verifier, and whenever crs and the verifier randomness are chosen so that this happens, $\mathcal{A}_0$ will win. This happens with probability $p \geq 1/\text{poly}(\lambda)$ which contradicts the assumption that the protocol has negligible soundness error. □

Claim 6.14. *If $\mathbb{x} \leftarrow \mathcal{D}_{\mathsf{G}}$, then $\mathcal{A}(\mathbb{x})$ outputs 1 with probability $\geq 1/\mathrm{poly}(\lambda)$.*

Proof. Let $\mathsf{GoodCRS}$ be the event that crs is such that

$$\Pr_{(\mathbb{x}',\mathbb{w}')\leftarrow\mathcal{D}_{\mathsf{G}}}\left[\mathcal{V}^{\mathcal{G}}(\mathsf{crs},\mathbb{x}',[\Pi']_1)=1 \middle| [\Pi']_1 \leftarrow \mathcal{P}^{\mathcal{G}}(\mathsf{crs},\mathbb{x}',\mathbb{w}')\right] \geq 1-\alpha,$$

where by $(\mathbb{x}',\mathbb{w}') \leftarrow \mathcal{D}_{\mathsf{G}}$ we mean that $\mathbb{w}'$ is the preimage of the PRG. By completeness of the SNARG, $\Pr_{\mathsf{crs}}[\mathsf{GoodCRS}] \geq 1-\alpha$. For the rest of the proof we assume $\mathsf{GoodCRS}$ holds.

We define another event, $\mathsf{GoodInstance}$, representing when for the challenge $\mathbb{x}^*$ we have $\mathcal{V}(\mathsf{crs},\mathbb{x}^*,[\Pi]_1) = 1$ for $[\Pi]_1 \leftarrow \mathcal{P}^{\mathcal{G}}(\mathsf{crs},\mathbb{x}^*,\mathbb{w}^*)$. By definition, $\Pr[\mathsf{GoodInstance}|\mathsf{GoodCRS}] \geq 1-\alpha$.

Let $\mathbf{\Pi}^*$ be the vector that $\mathcal{P}^{\mathcal{G}}(\mathsf{crs},\mathbb{x}^*,\mathbb{w}^*)$ uses to compute $[\Pi]_1 \leftarrow \langle \mathbf{\Pi}^*, [\mathsf{crs}_1]_1\rangle$. For notational convenience, we abuse notation and write $([\Pi]_1,\mathbf{\Pi}^*) \leftarrow \mathcal{P}^{\mathcal{G}}(\mathsf{crs},\mathbb{x}^*,\mathbb{w}^*)$. By Claim 6.13 for $\mathbb{x} \leftarrow \mathcal{D}_{\mathsf{G}}$ with probability $1-\mathsf{negl}(\lambda)$ there exists a query i such that $\langle \mathbf{a}_i, [\mathsf{crs}_2]_2\rangle \neq 0$ and $[\Pi]_1 \cdot \langle \mathbf{a}_i, [\mathsf{crs}_2]_2\rangle + \langle \mathbf{b}_i, [\mathsf{crs}_1]_1 \otimes [\mathsf{crs}_2]_2\rangle + \langle \mathbf{c}_i, [\mathsf{crs}_T]_T\rangle = [0]_T$. Let i^* be the random variable representing the smallest of these indeces.

We now argue about the behavior of $\mathsf{SolutionFinder}$. It will eventually use a basis $\mathcal{B}$ to check whether

$$\begin{pmatrix} \mathbf{\Pi}^* \otimes \mathbf{a}_{i^*} \\ \mathbf{b}_{i^*} \\ \mathbf{c}_{i^*} \end{pmatrix} \in \mathsf{span}(\mathcal{B}).$$

Let $\mathcal{B}_j$ be $\mathcal{B}$ after adding the j-th vector to it. Before $\mathsf{SolutionFinder}$ uses this as its final basis it samples $2\lambda/(1-\alpha)$ many independent $\mathbb{x}$ and $\mathbb{w}$. If for a fixed j

$$\Pr_{(\mathbb{x},\mathbb{w})\leftarrow\mathcal{R}_L}\left[\begin{array}{l|l} \exists i. & \\ [\Pi]_1 \cdot \langle \mathbf{a}_i, [\mathsf{crs}_2]_2\rangle + \langle \mathbf{b}_i, [\mathsf{crs}_1]_1 \otimes [\mathsf{crs}_2]_2\rangle & ([\Pi]_1,\mathbf{\Pi}^*) \leftarrow \mathcal{P}^{\mathcal{G}}(\mathsf{crs},\mathbb{x},\mathbb{w}) \\ \quad + \langle \mathbf{c}_i, [\mathsf{crs}_T]_T\rangle = [0]_T, & \mathcal{V}^{\mathcal{G}}(\mathsf{crs},\mathbb{x},\mathbb{w}) \text{ makes} \\ \begin{pmatrix} \mathbf{\Pi}^* \otimes \mathbf{a}_i \\ \mathbf{b}_i \\ \mathbf{c}_i \end{pmatrix} \notin \mathcal{B}_j & \text{paring equality checks} \\ & (\mathbf{a}_i,\mathbf{b}_i,\mathbf{c}_i)_{i\in[\ell]} \end{array}\right]$$
$$> (1-\alpha)/2,$$

then $\mathsf{SolutionFinder}$ will use it in the final computation with probability $(1-(1-\alpha)/2)^{2\lambda/(1-\alpha)}$. Because in the $\mathsf{SolutionFinder}$ there are only polynomially many $\mathcal{B}_j$ we have that the final state of $\mathcal{B}$ has the property

$$\Pr_{(\mathbb{x},\mathbb{w})\leftarrow\mathcal{R}_L}\left[\begin{array}{l|l} \exists i. & \\ [\Pi]_1 \cdot \langle \mathbf{a}_i, [\mathsf{crs}_2]_2\rangle + \langle \mathbf{b}_i, [\mathsf{crs}_1]_1 \otimes [\mathsf{crs}_2]_2\rangle & ([\Pi]_1,\mathbf{\Pi}^*) \leftarrow \mathcal{P}^{\mathcal{G}}(\mathsf{crs},\mathbb{x},\mathbb{w}) \\ \quad + \langle \mathbf{c}_i, [\mathsf{crs}_T]_T\rangle = [0]_T, & \mathcal{V}^{\mathcal{G}}(\mathsf{crs},\mathbb{x},\mathbb{w}) \text{ makes} \\ \begin{pmatrix} \mathbf{\Pi}^* \otimes \mathbf{a}_i \\ \mathbf{b}_i \\ \mathbf{c}_i \end{pmatrix} \notin \mathcal{B} & \text{paring equality checks} \\ & (\mathbf{a}_i,\mathbf{b}_i,\mathbf{c}_i)_{i\in[\ell]} \end{array}\right]$$
$$> (1-\alpha)/2,$$

with probability $\leq \mathsf{negl}(\lambda)$. We call this event $\mathsf{BadBasis}$. For the rest of the proof we assume $\overline{\mathsf{BadBasis}}$.

In this case, $\mathsf{SolutionFinder}$ outputs some vector $\mathbf{\Pi}$ such that $\begin{pmatrix}\mathbf{\Pi}\otimes\mathbf{a}_{i^*}\\ \mathbf{b}_{i^*}\\ \mathbf{c}_{i^*}\end{pmatrix}\in\mathsf{span}(\mathcal{B})$ with probability $\geq 1-(1-\alpha)/2$. Because every vector $\mathbf{v}\in\mathcal{B}$ is such that $\left\langle \mathbf{v}, \begin{pmatrix}[\mathsf{crs}_1]_1\otimes[\mathsf{crs}_2]_2\\ [\mathsf{crs}_1]_1\otimes[\mathsf{crs}_2]_2\\ [\mathsf{crs}_T]_T\end{pmatrix}\right\rangle = [0]_T$ we get that

$$\begin{aligned}
\left\langle \begin{pmatrix}\mathbf{\Pi}\otimes\mathbf{a}_{i^*}\\ \mathbf{b}_{i^*}\\ \mathbf{c}_{i^*}\end{pmatrix}, \begin{pmatrix}[\mathsf{crs}_1]_1\otimes[\mathsf{crs}_2]_2\\ [\mathsf{crs}_1]_1\otimes[\mathsf{crs}_2]_2\\ [\mathsf{crs}_T]_T\end{pmatrix}\right\rangle &= [0]_T = \left\langle \begin{pmatrix}\mathbf{\Pi}^*\otimes\mathbf{a}_{i^*}\\ \mathbf{b}_{i^*}\\ \mathbf{c}_{i^*}\end{pmatrix}, \begin{pmatrix}[\mathsf{crs}_1]_1\otimes[\mathsf{crs}_2]_2\\ [\mathsf{crs}_1]_1\otimes[\mathsf{crs}_2]_2\\ [\mathsf{crs}_T]_T\end{pmatrix}\right\rangle \Leftrightarrow \\
\langle \mathbf{\Pi}\otimes\mathbf{a}_{i^*}, [\mathsf{crs}_1]_1\otimes[\mathsf{crs}_2]_2\rangle &= \langle \mathbf{\Pi}^*\otimes\mathbf{a}_{i^*}, [\mathsf{crs}_1]_1\otimes[\mathsf{crs}_2]_2\rangle \Leftrightarrow \\
\langle \mathbf{\Pi}, [\mathsf{crs}_1]_1\rangle\cdot\langle \mathbf{a}_{i^*}, [\mathsf{crs}_2]_2\rangle &= \langle \mathbf{\Pi}^*, [\mathsf{crs}_1]_1\rangle\cdot\langle \mathbf{a}_{i^*}, [\mathsf{crs}_2]_2\rangle \Leftrightarrow \\
\langle \mathbf{\Pi}, [\mathsf{crs}_1]_1\rangle &= \langle \mathbf{\Pi}^*, [\mathsf{crs}_1]_1\rangle \\
&= [\Pi]_1
\end{aligned}$$

Therefore, $\mathcal{A}$ outputs 1 with probability

$$\begin{aligned}
&\geq \Pr\left[\mathsf{GoodCRS}\wedge\overline{\mathsf{BadBasis}}\wedge\mathsf{GoodInstance}\wedge\begin{pmatrix}\mathbf{\Pi}\otimes\mathbf{a}_{i^*}\\ \mathbf{b}_{i^*}\\ \mathbf{c}_{i^*}\end{pmatrix}\in\mathsf{span}(\mathcal{B})\right] \\
&\geq \Pr\left[\overline{\mathsf{BadBasis}}\wedge\mathsf{GoodInstance}\wedge\begin{pmatrix}\mathbf{\Pi}\otimes\mathbf{a}_{i^*}\\ \mathbf{b}_{i^*}\\ \mathbf{c}_{i^*}\end{pmatrix}\in\mathsf{span}(\mathcal{B})\middle|\mathsf{GoodCRS}\right](1-\alpha) \\
&\geq \Pr\left[\mathsf{GoodInstance}\wedge\begin{pmatrix}\mathbf{\Pi}\otimes\mathbf{a}_{i^*}\\ \mathbf{b}_{i^*}\\ \mathbf{c}_{i^*}\end{pmatrix}\in\mathsf{span}(\mathcal{B})\middle|\begin{matrix}\mathsf{GoodCRS},\\ \overline{\mathsf{BadBasis}}\end{matrix}\right](1-\alpha)(1-\mathsf{negl}(\lambda)) \\
&\quad\geq (1-\alpha-(1-\alpha)/2)(1-\alpha)(1-\mathsf{negl}(\lambda)) \\
&\quad\geq ((1-\alpha)/2)(1-\alpha)(1-\mathsf{negl}(\lambda)) \\
&\quad\geq 1/\mathrm{poly}(\lambda).
\end{aligned}$$

□

Lemma 6.11 follows from Claim 6.14 and Claim 6.12. □

Acknowledgments

Gal Arnon is supported by the European Research Council (ERC) under the EU's Horizon 2020 research and innovation programme (Grant agreement No. 101019547), and Stellar Foundation Grant. This work was done in part while Gal Arnon was a research fellow at the Simons Institute for the Theory of Computing. This work was done in part while Jesko Dujmovic was at CISPA Helmholtz-Center for Information Security. Jesko Dujmovic was supported by the European Reasearch Union (ERC, LACONIC, 101041207). Eylon Yogev is supported by the Israel Science Foundation (Grant No. 2302/22), and European Research Union (ERC, CRYPTOPROOF, 101164375). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

References

[ACFY24] Gal Arnon, Alessandro Chiesa, Giacomo Fenzi, and Eylon Yogev. "STIR: Reed–Solomon Proximity Testing with Fewer Queries". In: *Proceedings of the 44th Annual International Cryptology Conference*. CRYPTO '24. 2024, pp. 380–413.

[ACFY25] Gal Arnon, Alessandro Chiesa, Giacomo Fenzi, and Eylon Yogev. "WHIR: Reed-Solomon Proximity Testing with Super-Fast Verification". In: *Advances in Cryptology - EUROCRYPT 2025 - 44th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Madrid, Spain, May 4-8, 2025, Proceedings, Part IV*. Ed. by Serge Fehr and Pierre-Alain Fouque. Vol. 15604. Lecture Notes in Computer Science. Springer, 2025, pp. 214–243.

[ACY23] Gal Arnon, Alessandro Chiesa, and Eylon Yogev. "IOPs with Inverse Polynomial Soundness Error". In: *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023, Santa Cruz, CA, USA, November 6-9, 2023*. IEEE, 2023, pp. 752–761.

[ADI25] Gal Arnon, Jesko Dujmovic, and Yuval Ishai. "Designated-verifier SNARGs with one group element". In: *Annual International Cryptology Conference*. Springer. 2025, pp. 192–224.

[AFLN24] Martin R. Albrecht, Giacomo Fenzi, Oleksandra Lapiha, and Ngoc Khanh Nguyen. "SLAP: Succinct Lattice-Based Polynomial Commitments from Standard Assumptions". In: *Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Ed. by Marc Joye and Gregor Leander. Vol. 14656. Lecture Notes in Computer Science. Springer, 2024, pp. 90–119.

[AHIV17] Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. "Ligero: Lightweight Sublinear Arguments Without a Trusted Setup". In: *Proceedings of the 24th ACM Conference on Computer and Communications Security*. CCS '17. 2017, pp. 2087–2104.

[ALMSS98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. "Proof verification and the hardness of approximation problems". In: *Journal of the ACM* 45.3 (1998). Preliminary version in FOCS '92., pp. 501–555.

[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.

[BBHR18] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. "Fast Reed–Solomon Interactive Oracle Proofs of Proximity". In: *Proceedings of the 45th International Colloquium on Automata, Languages and Programming*. ICALP '18. 2018, 14:1–14:17.

[BBHR19] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. "Scalable Zero Knowledge with No Trusted Setup". In: *Proceedings of the 39th Annual International Cryptology Conference*. CRYPTO '19. 2019, pp. 733–764.

[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.

[BCIOP13] Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. "Succinct Non-Interactive Arguments via Linear Interactive Proofs". In: *Proceedings of the 10th Theory of Cryptography Conference*. TCC '13. 2013, pp. 315–333.

[BCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. "Interactive Oracle Proofs". In: *Proceedings of the 14th Theory of Cryptography Conference*. TCC '16-B. 2016, pp. 31–60.

[BHIRW24] Nir Bitansky, Prahladh Harsha, Yuval Ishai, Ron D Rothblum, and David J Wu. "Dot-product proofs and their applications". In: *2024 IEEE 65th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. 2024, pp. 806–825.

[BIOW20] Ohad Barta, Yuval Ishai, Rafail Ostrovsky, and David J Wu. "On succinct arguments and witness encryption from groups". In: *Annual International Cryptology Conference*. Springer. 2020, pp. 776–806.

[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.

[BS23] Ward Beullens and Gregor Seiler. "LaBRADOR: Compact Proofs for R1CS from Module-SIS". In: *Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part V*. Ed. by Helena Handschuh and Anna Lysyanskaya. Vol. 14085. Lecture Notes in Computer Science. Springer, 2023, pp. 518–548.

[CGKY25] Alessandro Chiesa, Ziyi Guan, Christian Knabenhans, and Zihan Yu. "On the Fiat-Shamir Security of Succinct Arguments from Functional Commitments". In: *IACR Cryptol. ePrint Arch.* (2025), p. 902. URL: https://eprint.iacr.org/2025/902.

[CY24] Alessandro Chiesa and Eylon Yogev. *Building Cryptographic Proofs from Hash Functions*. 2024. URL: https://github.com/hash-based-snargs-book.

[DFGK14] George Danezis, Cedric Fournet, Jens Groth, and Markulf Kohlweiss. "Square Span Programs with Applications to Succinct NIZK Arguments". In: *Proceedings of the 20th International Conference on the Theory and Application of Cryptology and Information Security*. ASIACRYPT '14. 2014, pp. 532–550.

[DMS24] Michel Dellepere, Pratyush Mishra, and Alireza Shirzad. "Garuda and Pari: Faster and Smaller SNARKs via Equifficient Polynomial Commitments". In: *IACR Cryptol. ePrint Arch.* (2024), p. 1245.

[FS86] Amos Fiat and Adi Shamir. "How to prove yourself: practical solutions to identification and signature problems". In: *Proceedings of the 6th Annual International Cryptology Conference*. CRYPTO '86. 1986, pp. 186–194.

[GGPR13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. “Quadratic Span Programs and Succinct NIZKs without PCPs”. In: *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings.* Ed. by Thomas Johansson and Phong Q. Nguyen. Vol. 7881. Lecture Notes in Computer Science. Springer, 2013, pp. 626–645.

[GLSTW23] Alexander Golovnev, Jonathan Lee, Srinath T. V. Setty, Justin Thaler, and Riad S. Wahby. “Brakedown: Linear-time and field-agnostic SNARKs for R1CS”. In: *Proceedings of the 43rd Annual International Cryptology Conference.* CRYPTO ’23. 2023.

[GWC19] Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. *PLONK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge.* Cryptology ePrint Archive, Report 2019/953. 2019.

[Gro10] Jens Groth. “Short Pairing-Based Non-interactive Zero-Knowledge Arguments”. In: *Proceedings of the 16th International Conference on the Theory and Application of Cryptology and Information Security.* ASIACRYPT ’10. 2010, pp. 321–340.

[Gro16] Jens Groth. “On the Size of Pairing-Based Non-interactive Arguments”. In: *Proceedings of the 35th Annual International Conference on Theory and Applications of Cryptographic Techniques.* EUROCRYPT ’16. 2016, pp. 305–326.

[HNY22] Iftach Haitner, Daniel Nukrai, and Eylon Yogev. “Lower Bound on SNARGs in the Random Oracle Model”. In: *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part III.* Ed. by Yevgeniy Dodis and Thomas Shrimpton. Vol. 13509. Lecture Notes in Computer Science. Springer, 2022, pp. 97–127.

[IKO05] Yuval Ishai, Eyal Kushilevitz, and Rafail Ostrovsky. “Sufficient Conditions for Collision-Resistant Hashing”. In: *Proceedings of the 2nd Theory of Cryptography Conference.* TCC ’05. 2005, pp. 445–456.

[IKO07] Yuval Ishai, Eyal Kushilevitz, and Rafail Ostrovsky. “Efficient Arguments without Short PCPs”. In: *Proceedings of the Twenty-Second Annual IEEE Conference on Computational Complexity.* CCC ’07. 2007, pp. 278–291.

[JLLW23] Aayush Jain, Huijia Lin, Ji Luo, and Daniel Wichs. “The Pseudorandom Oracle Model and Ideal Obfuscation”. In: *Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part IV.* Ed. by Helena Handschuh and Anna Lysyanskaya. Vol. 14084. Lecture Notes in Computer Science. Springer, 2023, pp. 233–262.

[Kho02] Subhash Khot. “On the Power of Unique 2-Prover 1-Round Games”. In: *Proceedings of the 17th Annual IEEE Conference on Computational Complexity, Montréal, Québec, Canada, May 21-24, 2002.* IEEE Computer Society, 2002, p. 25.

[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.

[LM19] Russell WF Lai and Giulio Malavolta. “Subvector commitments with application to succinct arguments”. In: *Advances in Cryptology–CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2019, Proceedings, Part I 39.* Springer. 2019, pp. 530–560.

[Lip12] Helger Lipmaa. “Progression-Free Sets and Sublinear Pairing-Based Non-Interactive Zero-Knowledge Arguments”. In: *Proceedings of the 9th Theory of Cryptography Conference on Theory of Cryptography.* TCC ’12. 2012, pp. 169–189.

[Lip24] Helger Lipmaa. "Polymath: Groth16 Is Not the Limit". In: *Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2024, Proceedings, Part X*. Ed. by Leonid Reyzin and Douglas Stebila. Vol. 14929. Lecture Notes in Computer Science. Springer, 2024, pp. 170–206.

[Mau05] Ueli Maurer. "Abstract models of computation in cryptography". In: *Cryptography and Coding: 10th IMA International Conference, Cirencester, UK, December 19-21, 2005. Proceedings 10*. Springer. 2005, pp. 1–12.

[Mic00] Silvio Micali. "Computationally Sound Proofs". In: *SIAM Journal on Computing* 30.4 (2000). Preliminary version appeared in FOCS '94., pp. 1253–1298.

[Nec94] Vassiliy Ilyich Nechaev. "Complexity of a determinate algorithm for the discrete logarithm". In: *Mathematical Notes* 55 (2 1994), pp. 165–172.

[SCI14] SCIPR Lab. *libsnark: a C++ library for zkSNARK proofs*. 2014. URL: `https://github.com/scipr-lab/libsnark`.

[SSEKYZ24] Ron Steinfeld, Amin Sakzad, Muhammed F. Esgin, Veronika Kuchta, Mert Yassi, and Raymond K. Zhao. "LUNA: Quasi-Optimally Succinct Designated-Verifier Zero-Knowledge Arguments from Lattices". In: *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security, CCS 2024, Salt Lake City, UT, USA, October 14-18, 2024*. Ed. by Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie. ACM, 2024, pp. 3167–3181.

[ST06] Alex Samorodnitsky and Luca Trevisan. "Gowers uniformity, influence of variables, and PCPs". In: *Proceedings of the thirty-eighth annual ACM symposium on Theory of Computing*. 2006, pp. 11–20.

[STW23] Srinath Setty, Justin Thaler, and Riad Wahby. "Customizable constraint systems for succinct arguments". In: *IACR Cryptol. ePrint Arch.* (2023), p. 552.

[SW14] Amit Sahai and Brent Waters. "How to use indistinguishability obfuscation: deniable encryption, and more". In: *STOC*. 2014.

[Sho97] Victor Shoup. "Lower bounds for discrete logarithms and related problems". In: *Proceedings of the 16th International Conference on the Theory and Application of Cryptographic Techniques*. EUROCRYPT '97. 1997, pp. 256–266.

[WW24] Brent Waters and David J. Wu. "Adaptively-Sound Succinct Arguments for NP from Indistinguishability Obfuscation". In: *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024, Vancouver, BC, Canada, June 24-28, 2024*. Ed. by Bojan Mohar, Igor Shinkar, and Ryan O'Donnell. ACM, 2024, pp. 387–398.

[WW25] Brent Waters and David J. Wu. "A Pure Indistinguishability Obfuscation Approach to Adaptively-Sound SNARGs for sfNP". In: *Proceedings of the 45th Annual International Cryptology Conference*. Ed. by Yael Tauman Kalai and Seny F. Kamara. Vol. 16006. CRYPTO '25. Springer, 2025, pp. 292–326.

[WZ24] Brent Waters and Mark Zhandry. "Adaptive Security in SNARGs via iO and Lossy Functions". In: *Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2024, Proceedings, Part X*. Ed. by Leonid Reyzin and Douglas Stebila. Vol. 14929. Lecture Notes in Computer Science. 2024, pp. 72–104.

[ZCF24] Hadas Zeilberger, Binyi Chen, and Ben Fisch. "BaseFold: Efficient Field-Agnostic Polynomial Commitment Schemes from Foldable Codes". In: *Proceedings of the 44th Annual International Cryptology Conference*. Vol. 14929. CRYPTO '24. 2024, pp. 138–169.

[Zha22] Mark Zhandry. "To label, or not to label (in generic groups)". In: *Annual International Cryptology Conference*. Springer. 2022, pp. 66–96.